Computational Methods to Improve Satellite Attitude Determination and Control with a Focus on Autonomy, Generalizability, and Underactuation

by

Patrick McKeen

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

DOCTORATE OF SCIENCE IN SPACE SYSTEMS ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2025

© 2025 Patrick McKeen. This work is licensed under a CC BY-NC-ND 4.0 license.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Patrick McKeen
Department of Aeronautics and Astronautics
January 10, 2025

Certified by: Kerri Cahoy
Professor of Aeronautics and Astronautics, Thesis Supervisor

Accepted by: Jonathan P. How
R. C. Maclaurin Professor in Aeronautics and Astronautics
Chair, Graduate Program Committee

Computational Methods to Improve Satellite Attitude Determination and Control with a Focus on Autonomy, Generalizability, and Underactuation

by

Patrick McKeen

Submitted to the Department of Aeronautics and Astronautics
on January 10, 2025 in partial fulfillment of the requirements for the degree of

DOCTORATE OF SCIENCE IN SPACE SYSTEMS ENGINEERING

ABSTRACT

The attitude determination and control system (ADCS) onboard a satellite uses sensors to measure orientation and angular velocity, enabling the satellite to manage angular momentum, counteract disturbances, and point in the desired directions. Many historical ADCS approaches are designed for constant pointing goals, high accuracy sensors, powerful actuators, or larger, high-inertia satellites. Many modern satellites are small satellites (tens of kilograms or less), with lower-cost actuators and sensors, and may have more complicated attitude goals. This dissertation presents a variety of computational approaches to improve ADCS performance by leveraging detailed satellite dynamics modeling and estimation, disturbance inclusion, and trajectory planning–all optimized for efficient onboard computation suitable for small satellites. The proposed framework generalizes ADCS operations, allowing it to adapt automatically to different satellite types, mission requirements, and operational goals, reducing reliance on predefined ground-based commands. This framework can be used in place of standard control laws to make ADCS more autonomous and "hands-off," calculating its own slews and desaturation while meeting pointing goals, even in cases of underactuation or large disturbances. This generalized and autonomous framework is a contribution of this work, alongside each of its components, which can be individually used in their own right.

One key component of this work is a generalized state estimator that integrates a dynamic model of the spacecraft. This estimator demonstrates high accuracy across various satellite configurations, achieving angular error as low as 0.01° in low Earth orbit (LEO) with high-quality sensors (but no star trackers), compared to the typical 1° error of conventional methods. The estimator can account for biases, sensor errors, and external disturbances, ensuring robust performance (e.g., 0.1° error in LEO) even with lower-quality sensors (MEMS gyroscopes, plus magnetometers and sun sensors). This adaptability highlights the increased autonomy of the system, as it requires minimal human intervention to maintain high accuracy across diverse mission scenarios.

Another major contribution is the integration of disturbance modeling into control laws. By accounting for disturbances directly (either individually or as an all-in-one value tracked by the estimator), rather than through reactive measures like integral control, the proposed methods improve stability and performance, particularly for underactuated systems–improving pointing accuracy by up to 20 degrees. The developed control laws are adaptable to various

actuator configurations, disturbance environments, and pointing objectives. This flexibility extends to modifying pointing goals, such as aligning specific vectors rather than requiring a fully specified orientation, enhancing mission adaptability.

This work also implements a novel trajectory planning method that generates efficient pointing trajectories for both constant and time-varying goals. The method, based on the Augmented Lagrangian iterated-LQR (ALTRO) approach, creates sequential mission trajectories that optimize performance even under underactuation or disturbance conditions. The planned trajectories are followed by two types of robust closed-loop controllers, applicable across satellite architectures ranging from large weather satellites to 3U CubeSats. By enabling onboard trajectory planning and adaptive control adjustments, this method significantly reduces the need for ground-based planning and interventions, further advancing autonomous operation.

The combined framework of estimation, disturbance-aware control, and trajectory planning achieves significantly higher accuracy than traditional ADCS approaches. This enables the use of commercial off-the-shelf components in high-performance missions, overcoming the limitations of low-cost sensors and actuators. The proposed methods allow satellites to operate with weaker or fewer actuators, such as magnetic-only control, while still achieving precise pointing, thereby expanding the feasibility of more autonomous, robust, and cost-effective satellite operations.

Thesis supervisor: Kerri Cahoy
Title: Professor of Aeronautics and Astronautics

Funding

Various organizations funded my studies throughout my doctoral work. Some work was completed while supported by Northrop Grumman for work on the project "MIT Beaver-Cube II" and the US Air Force Artificial Intelligence Accelerator for "Automation in Space Domain Awareness." This material is also based upon work completed while supported by MIT Portugal: Portuguese Science and Technology Foundation for work on the project "AEROS Constellation—Development of a Nanosatellite Platform as a Precursor of a Future Constellation to Leverage the Space/Ocean Scientific and Economic Synergies" and MIT Lincoln Laboratory for work on the project "Time-Resolved Observations of Precipitation Structure and Storm Intensity with a Constellation of Smallsats (TROPICS)." This support is appreciated.

Acknowledgments

The work of this dissertation would not have been possible without the guidance, support, and encouragement of many individuals, to whom I owe my deepest gratitude.

First and foremost, I would like to thank my advisor, Professor Kerri Cahoy, whose vision, expertise, and support have guided me through this journey. Her ability to see potential and foster innovation set me on this path and has shaped my development as a researcher and engineer. I am immensely grateful for her mentorship and for always challenging me to think critically and aim higher.

I am also deeply appreciative of Professor Richard Linares, who has provided guidance throughout my doctoral work. I have learned a great deal from his feedback and thoughtful perspectives.

Like any doctoral program, my work is the product of collaboration and shared knowledge. Alex Rose played a significant role in the project that inspired this work, and their contributions, both in terms of insights and code, were crucial to its success. I am deeply thankful for their collaboration, friendship, and the foundation they helped lay for this research.

On a personal level, I am profoundly grateful to my platonic partner, Morgan, who has stood by me with patience and kindness throughout this adventure. Morgan's unwavering presence has provided much-needed stability and support, keeping me grounded through both triumphs and challenges. Similarly, my romantic partner, Max, has been a source of motivation and inspiration, pushing me forward even when the road felt impossibly long. Their encouragement and support has meant more than words can express.

To my family—Mom, Dad, and Anna—thank you for your unconditional love and steadfast belief in me. You have been my anchor throughout my life, and I would not have reached this point without your love and assistance. My extended family and friends-who-are-family have similarly been an invaluable source of strength for me.

Finally, I extend my gratitude to my colleagues, friends, and everyone who has offered their time, wisdom, and kindness along the way. This dissertation is as much a testament to your contributions as it is to my efforts. To all of you, thank you for being part of this journey.

Biographical Sketch

Patrick Calvin McKeen, born on March 25, 1994, in Winston-Salem, North Carolina, pursued his early education at the North Carolina School of Science and Mathematics. He later attended Harvey Mudd College, earning a Bachelor of Science in Engineering with a focus on controls and structures. During his undergraduate studies, McKeen's passion for aerospace was evident through internships at Blue Origin and Millennium Space Systems.

In 2017, McKeen joined the Massachusetts Institute of Technology (MIT) to pursue a master's degree under the mentorship of Professor Leia Stirling. His research centered on the modeling and tradespace exploration of spacesuit hip bearing assemblies, utilizing multi-degree-of-freedom range of motion analysis. This work aimed to enhance the design and functionality of spacesuit joints, contributing to improved astronaut mobility.

After earning his master's degree in 2019, McKeen continued at MIT as a doctoral candidate in the Space Telecommunications, Astronomy, and Radiation (STAR) Lab, led by Professor Kerri Cahoy. His contributions spanned various projects, including BeaverCube–a 3U CubeSat developed to capture visible and long-wave infrared images of Earth's oceans and cloud tops. He also played roles in the AEROS project, a CubeSat mission aimed at studying ocean health, and collaborated with BAE Systems on advanced aerospace technologies.

Beyond his research, McKeen served as a teaching assistant for multiple courses and as a mentor and team lead at MIT MakerWorkshop and ProtoWorks. Residing in Somerville, Massachusetts, with his partner Morgan and their dog Sojourner, he is an avid traveler, reader, skier, and DIY enthusiast.

Contents

Title Page **1**

Abstract **3**

Funding **5**

Acknowledgments **7**

Biographical Sketch **9**

List of Figures **15**

List of Tables **19**

1 Synopsis **21**
1.1 Summary . . . 21
1.2 Motivation . . . 22
1.3 Contributions in Brief . . . 23
1.4 System Model Overview . . . 24
1.4.1 Rotational Dynamics . . . 24
1.4.2 Sensors and Actuators . . . 24
1.4.3 Disturbances . . . 25
1.5 Case Overview . . . 25
1.6 Thesis Organization . . . 28

2 Background **31**
2.1 Introduction . . . 31
2.2 Previous Work . . . 33
2.2.1 Attitude Determination . . . 33
2.2.2 Attitude Control . . . 35
2.2.3 Magnetic Control . . . 40
2.2.4 Trajectory Planning . . . 44
2.2.5 Reduced Attitude or Single Axis Control . . . 46
2.2.6 Disturbance Modeling in Attitude Control . . . 47
2.2.7 Real-World Missions . . . 50
2.3 Current State of the Art . . . 51

2.3.1 Current State Summary . . . 51
2.3.2 Technical Gaps . . . 51
2.4 Specific Motivation . . . 53
2.5 Conclusion . . . 53

3 Approach and Contributions **57**
3.1 Introduction . . . 57
3.2 Areas of Work . . . 58
3.2.1 Improved Attitude State Estimation . . . 59
3.2.2 ADCS Operations and Goals . . . 60
3.2.3 Direct Countering of Disturbances . . . 60
3.2.4 Attitude Trajectory Planning . . . 60
3.2.5 Generalized ADCS . . . 61
3.3 Contributions . . . 61
3.3.1 Generalized Disturbance- and Dynamics-Aware Estimator . . . 61
3.3.2 Disturbance Modeling for Control Inclusion . . . 63
3.3.3 Trajectory Planning . . . 64
3.3.4 Improved Autonomy for ADCS . . . 65
3.3.5 ADCS Architecture for Weak or Underactuated Satellites . . . 66
3.4 Technical Background . . . 66
3.4.1 Quaternions . . . 66
3.4.2 Rotational Dynamics . . . 70
3.4.3 Error, Bias, and Noise . . . 70
3.4.4 Real-world Systems . . . 71
3.4.5 Kalman Filter Overview . . . 75
3.5 Conclusion . . . 81

4 Modeling and Estimation of Spacecraft Attitude **83**
4.1 Introduction . . . 83
4.2 System Definition . . . 87
4.2.1 Timing Questions . . . 88
4.3 Software Approach . . . 89
4.3.1 Satellite Class . . . 91
4.3.2 Sensors . . . 93
4.3.3 Actuators . . . 94
4.3.4 Disturbances . . . 96
4.3.5 Estimation . . . 97
4.4 Tests and Results . . . 108
4.4.1 Simulation Information . . . 109
4.4.2 Case A: TRMM Satellite with Initial Attitude and Bias Errors . . . 109
4.4.3 Case B: TRMM Satellite with Only Small Initial Bias Errors . . . 117
4.4.4 Case C: CubeSat with Initial Bias and Attitude Errors . . . 121
4.4.5 Case D: CubeSat with Only Small Initial Bias Errors . . . 125
4.4.6 Case E: Effect of Ignoring Actuator Bias or Disturbances . . . 129
4.4.7 Case F: CubeSat with Propulsion . . . 132

4.4.8 Case G: Many Variables . . . 134
4.5 Extensions . . . 146
4.6 Conclusion . . . 147

5 Generalizing ADCS Goals and Control Laws 149
5.1 Introduction . . . 149
5.2 Specifying Time-Varying Goals Succinctly . . . 150
5.3 Aligning Two Vectors . . . 151
5.3.1 Derivation . . . 153
5.3.2 Alternate Formulations . . . 156
5.3.3 Minimum Slew from Arbitrary Quaternion . . . 159
5.3.4 Slewing with "Keep-out" Zone . . . 162
5.3.5 Application Cases . . . 164
5.4 Modifying and Generalizing Control Laws . . . 167
5.4.1 Adjusted Command . . . 167
5.4.2 Adjusted Torque . . . 167
5.4.3 Momentum Desaturation . . . 168
5.4.4 Command from Torque . . . 168
5.4.5 Bounds . . . 169
5.4.6 Thoughts on Gains and Other Parameters . . . 170
5.4.7 Goal-Selection on a Set . . . 170
5.5 Conclusion . . . 170

6 Active Disturbance Rejection 173
6.1 Introduction . . . 173
6.2 Comparison Cases . . . 176
6.2.1 Proportional-Derivative (Idealized) Thruster Control . . . 176
6.2.2 Proportional-Derivative Magnetic Control . . . 176
6.2.3 Sliding Scale Magnetic Control . . . 177
6.3 All-in-One Disturbance Torque . . . 178
6.4 Modified Sliding Scale Control Law . . . 179
6.5 Tests and Results . . . 179
6.5.1 Disturbance Inclusion . . . 180
6.5.2 CubeSat Tests . . . 198
6.5.3 Sliding versus Modified Sliding . . . 211
6.6 Conclusion . . . 216

7 Trajectory Planning 219
7.1 Introduction . . . 219
7.2 ALTRO . . . 220
7.2.1 ALTRO Overview . . . 220
7.2.2 Satellite-Specific ALTRO Details and Modifications . . . 228
7.3 Planning as Part of ADCS . . . 234
7.3.1 Planner-based ADCS Architecture . . . 234
7.3.2 Real-world Implementation . . . 236

7.3.3 Convergence Questions . . . 238
7.4 Following a Planned Trajectory . . . 239
7.4.1 Time-Varying Linear Quadratic Regulator . . . 239
7.4.2 Model Predictive Control . . . 240
7.5 Tests and Results . . . 242
7.5.1 Long-Term Trajectory Planning for Underactuated Satellite with Multiple Goals . . . 242
7.5.2 Planning around a Disturbance . . . 246
7.5.3 Monte Carlo for Various Goals . . . 251
7.5.4 Sequential Trajectories . . . 265
7.6 Conclusion . . . 273

8 Summary and Future Work **277**
8.1 Result Summary . . . 277
8.2 Contributions Revisited . . . 278
8.2.1 Generalized Disturbance- and Dynamics-Aware Estimator . . . 278
8.2.2 Disturbance Modeling for Control Inclusion . . . 279
8.2.3 Trajectory Planning . . . 280
8.2.4 Improved Autonomy for ADCS . . . 280
8.2.5 ADCS Architecture for Weak or Underactuated Satellites . . . 281
8.3 Future Work . . . 282
8.3.1 Software and Usability Improvements . . . 282
8.3.2 Control Methodology Extensions . . . 283
8.3.3 Real-World Performance . . . 283
8.3.4 Advanced Concepts . . . 284
8.4 Closing . . . 285

References **287**

List of Figures

1.1 Magnetic Control Problem Breakdown, with Extensions for Generalizability and Component Standalone Value . . . 23

2.1 Single-Axis Control Diagram . . . 47
2.2 Rapid Retargeting Trajectory . . . 54
2.3 Magnetorquer Example (ISISPACE iMTQ) . . . 55
2.4 The MIT BeaverCube CubeSat . . . 55

3.1 Magnetic Control Problem Breakdown, with Extensions for Generalizability and Component Standalone Value . . . 59
3.2 Relationships between Technical Gaps, Areas of Work, and Contributions . . . 62

4.1 Satellite Class Structure Example Diagram . . . 91
4.2 TRMM Satellite Angular Error with Initial Attitude and Bias Errors . . . 113
4.3 Error Bounds (3σ) for Case A Estimators . . . 115
4.4 TRMM Satellite Angular Velocity Error with Initial Attitude and Bias Errors 116
4.5 TRMM Satellite Angular Error with Small Initial Bias Errors Only . . . 118
4.6 Error Bounds (3σ) for Case B Estimators . . . 119
4.7 TRMM Satellite Angular Velocity Error with Small Initial Bias Errors Only 120
4.8 CubeSat Angular Error with Initial Attitude and Bias Errors . . . 123
4.9 CubeSat Angular Velocity Error with Initial Attitude and Bias Errors . . . 124
4.10 Error Bounds (3σ) for Case C: CubeSat with Initial Attitude and Bias Errors 125
4.11 CubeSat Angular Error with Small Initial Bias Errors Only . . . 126
4.12 Error Bounds (3σ) for Case D: CubeSat with Small Initial Bias Errors Only 127
4.13 CubeSat Angular Velocity Error with Small Initial Bias Errors Only . . . 128
4.14 CubeSat Angular Error when Actuator Bias is Included or Ignored . . . 130
4.15 CubeSat Angular Error when Disturbances are Included or Ignored . . . 131
4.16 CubeSat Angular Error when Propulsion is Included or Ignored . . . 132
4.17 CubeSat Propulsion Torque (Estimated and Real) . . . 133
4.18 CubeSat Angular Error with Many Variables . . . 138
4.19 CubeSat Angular Velocity Error with Many Variables . . . 139
4.20 CubeSat Angular Momentum with Many Variables . . . 140
4.21 Propulsion Torque with Many Variables . . . 141
4.22 Residual Dipole with Many Variables . . . 142
4.23 Residual Dipole (perpendicular to $\mathbf{B}$ field) with Many Variables . . . 143

4.24 Sensor Bias with Many Variables . . . 144
4.25 Error Bounds (3σ) for Case G Estimation . . . 145

5.1 Visual Example of Quaternion Set Goals . . . 151
5.2 Keep-out Zone Constraint Plot . . . 166

6.1 Wie Test Case Angular Error . . . 181
6.2 Wie Test Case Angular Velocity Error by Axis . . . 182
6.3 Wie Test Case Control Effort by Axis . . . 183
6.4 Lovera Test Case Angular Error . . . 186
6.5 Lovera Test Case Control Effort by Axis . . . 187
6.6 Wisniewski Log Satellite Angular Error . . . 191
6.7 Wisniewski Test Case Angular Error . . . 192
6.8 Wisniewski Test Case Control Effort by Axis . . . 193
6.9 Modified Sliding Satellite Log-scale Angular Error . . . 195
6.10 Modified Sliding Satellite Angular Error . . . 196
6.11 Modified Sliding Satellite Control Effort by Axis . . . 197
6.12 Wisniewski on CubeSat Angular Error . . . 200
6.13 Wisniewski on CubeSat Control Effort by Axis . . . 201
6.14 Lovera on CubeSat Angular Error . . . 203
6.15 Lovera on CubeSat Control Effort by Axis . . . 204
6.16 Modified Sliding on CubeSat Angular Error . . . 207
6.17 Modified Sliding on CubeSat Angular Velocity Error by Axis . . . 208
6.18 Modified Sliding on CubeSat Control Effort by Axis . . . 209
6.19 Sliding versus Modified Sliding Angular Error, Disturbed with Control . . . 212
6.20 Sliding versus Modified Sliding Angular Error, Disturbed . . . 213
6.21 Sliding versus Modified Sliding Angular Error, CubeSat, Disturbed . . . 214
6.22 Sliding versus Modified Sliding Angular Error, CubeSat, Disturbed with Control 215

7.1 ALTRO explanation . . . 222
7.2 Example Cost from RW Angular Momentum . . . 232
7.3 ADCS Overall . . . 235
7.4 Planner Timing Diagram . . . 236
7.5 Long-Term Two-Goal CubeSat Trajectory Visualization . . . 243
7.6 Long-Term Two-Goal CubeSat Trajectory Details . . . 245
7.7 Pointing Error of CubeSat Countering Disturbance by Spinning . . . 248
7.8 Angular Velocity of CubeSat Countering Disturbance by Spinning . . . 249
7.9 Command of CubeSat Countering Disturbance by Spinning . . . 250
7.10 Slew Trajectories for a MTQ-only satellite. . . . 253
7.11 Slew Trajectories for a 3MTQ+1RW satellite. . . . 253
7.12 Final RW Angular Momentum for a 3MTQ+1RW satellite in 180° slew . . . 254
7.13 Slew Convergence Examples for a MTQ-only satellite. . . . 255
7.14 Convergence Examples for a 3MTQ+1RW satellite in 180° Slew. . . . 255
7.15 Reduced-Attitude Slew Trajectories for a MTQ-only satellite. . . . 257
7.16 Reduced-Attitude Slew Trajectories for a 3MTQ+1RW satellite. . . . 258

7.17 Reduced-Attitude Slew Convergence Examples for a MTQ-only satellite. . . 258
7.18 Reduced-Attitude Slew Convergence Examples for a 3MTQ+1RW satellite. . 259
7.19 Final RW Angular Momentum for a 3MTQ+1RW satellite in Reducued-Attitude Slew 259
7.20 Multi-Target Trajectories for a MTQ-only satellite. 261
7.21 Multi-Target Trajectories for a 3MTQ+1RW satellite. 262
7.22 Multi-Target Convergence Examples for a MTQ-Only satellite 263
7.23 Multi-Target Convergence Examples for a 3MTQ+1RW satellite 263
7.24 Quaternion of a Sequentially-Planned Trajectory 268
7.25 Angular Velocity of a Sequentially-Planned Trajectory 269
7.26 Pointing Error of a Sequentially-Planned Trajectory 270
7.27 Command of a Sequentially-Planned Trajectory 271
7.28 Log-Scale Pointing Error of a Sequentially-Planned Trajectory 272

List of Tables

1.1 Summary of Satellite Test Cases 25

4.1 Summary of Test Cases in Chapter 4 85
4.2 Satellite Properties and Test Parameters for Case A 110
4.3 Satellite Properties and Test Parameters for Case B 117
4.4 Satellite Properties and Test Parameters for Case C 121
4.5 Satellite Properties and Test Parameters for Case D 125
4.6 Satellite Properties and Test Parameters for Case G 134

5.1 Control Goal Type Summary 152
5.2 Quaternion Set Formulation Summary. 156
5.3 Summary of Possibilities for Quaternion in Set Closest to $\bar{p}$. 160

6.1 Summary of Disturbance Control Test Cases 174
6.2 Satellite Properties and Test Parameters for Wie Comparison 180
6.3 Satellite Properties and Test Parameters for Lovera Comparison 185
6.4 Satellite Properties and Test Parameters for Wisniewski Comparison 189
6.5 Satellite Properties and Test Parameters for Wisniewski CubeSat Test 198
6.6 Satellite Properties and Test Parameters for Lovera CubeSat Test 202
6.7 Satellite Properties and Test Parameters for Modified Sliding CubeSat Test . 205

7.1 Summary of Planning Test Cases 221
7.2 Satellite Properties and Test Parameters for Underactuated CubeSat Two-Goal Test 244
7.3 Satellite Properties and Test Parameters for Overwhelming Disturbance Test 246
7.4 Monte Carlo Planning Test Satellite Properties 251
7.5 Pointing Accuracy in Monte Carlo Simulations for the 180-Degree Slew Case. 252
7.6 Pointing Accuracy in Monte Carlo Simulations for the Reduced Attitude Goal Case. 256
7.7 Pointing Accuracy in Monte Carlo Simulations for the Multi-Target Slew Case. 261
7.8 Satellite Properties and Test Parameters for Sequential Planning Test 265

Chapter 1

Synopsis

1.1 Summary

The goal of this dissertation is to improve attitude determination and control (ADC) for small satellites. When satellites are underactuated, have weak actuators, have large disturbances, or have poor sensors, it is challenging to make them point as desired. Generally, the best solution is to improve the actuators and sensors. When possible, this is the author's recommendation. However, it is not always possible. This work investigates several methods to improve the performance of satellites with limited sensor and actuator capability. By improving the performance of such satellites, architectures such as constellations of proliferated low-cost satellites become a possibility for missions, which can still meet challenging requirements. A magnetically-controlled satellite, for example, is not currently a feasible choice for a LEO mission that requires single or sub-degree-level pointing to variable targets. This work shows that achieving single or sub-degree pointing with magnetically controlled satellites is possible and could be considered for future missions. The techniques explored in this work, and the framework created for it, are not only applicable to small satellites with difficult ADC scenarios. By applying the same techniques to a spacecraft with a better ADC system (ADCS), that spacecraft can also improve performance–it may be more accurate, capable of more goals per unit time, or more energy-efficient. The techniques developed in this work are not limited to CubeSats or satellites with poor ADCS.

This concept of generalizing and improving ADCS techniques to many spacecraft is a central goal throughout this dissertation. Many historical approaches to ADCS are mission- or spacecraft-specific, and cannot be easily generalized to satellites with other sensors, actuators, inertia properties, goals, orbits, or disturbance environments. As this dissertation was developed, the author wanted his work to be applicable to many different satellites and missions. The resulting ADC tools and simulation are valuable for debugging and other development tasks, as they allow for simpler testing and a shared framework. However, the tools are also valuable in their own right, expanding the capabilities of the field. With the advancements in miniaturization and mass-production of satellites, there are many more satellites with a wider variety of architectures and goals than ever before. ADCS approaches that can be generally applied have become more valuable and allow for effort to be spent on improving the systems, as opposed to customizing or re-developing the ADCS for each

satellite.

Generalizability goes hand-in-hand with another goal of this work–enabling autonomy. Even recently, many spacecraft are designed to maintain a specific orientation, and slews between states have been hand-planned on the ground. Throughout this dissertation, the author has worked toward a vision of ADCS where the operators only need to give the satellite a set of pointing goals and the associated times, along with some limits to avoid, such as high rotation rates or pointing the camera towards the sun. The spacecraft would then work on its own to achieve these goals, adapting to its own physical parameters, actuator limits, constraints, and any disturbances as needed. Ideally, this would allow for rapid retargeting and switching quickly between varying goals–allowing the satellite to achieve more goals per unit time.

Most code and data for this work can be found on GitHub at: https://github.com/patrickmckeen/PhD_Dissertation_Code.git.

1.2 Motivation

The attitude determination system described in this work was originally inspired by the ADCS needs for an MIT student CubeSat mission called BeaverCube. When the author joined this mission, the plan called for magnetic attitude control and onboard electric propulsion. Magnetic attitude control is weak (as it generates torque by pushing against the Earth's magnetic field) and is underactuated (because it cannot torque around the magnetic field line). This raised a concern about a disturbance torque from the propulsion system (potentially caused by either off-axis thrust or misalignment between the thrust vector and center of mass). While the torque would be small, it would be continuous for hours, and if any component of it was aligned with the local $\mathbf{B}$ field, it would cause rotation that could not be countered. Additionally, the satellite's sensors were not high-end and had strong potential for noise, making the satellite attitude hard to estimate with precision.

In the efforts to solve the problem specifically for this mission, the author discovered that most approaches to ADCS are mission- and goal-specific. This led to the desire to develop a more general and autonomous system, as further detailed throughout this dissertation. A more general and autonomous system is valuable for testing and debugging, as specific aspects of the mission, environment, and spacecraft can be changed or turned on and off to discover their effects with only small changes to the code. As the work progressed and the author also worked on other missions, he decided it was valuable to develop an ADCS framework applicable to all missions.

The framework as a whole is described in this dissertation, and is a sizable contribution. In addition, each of the components that were required to solve this magnetic control problem is also valuable on their own (and in combinations that do not necessarily comprise the full framework). These ADC components can be applied to spacecraft generally, without needing to assume specific scenarios such as the propulsion disturbance or magnetic-only control. These possibilities are shown in Figure 1.1.

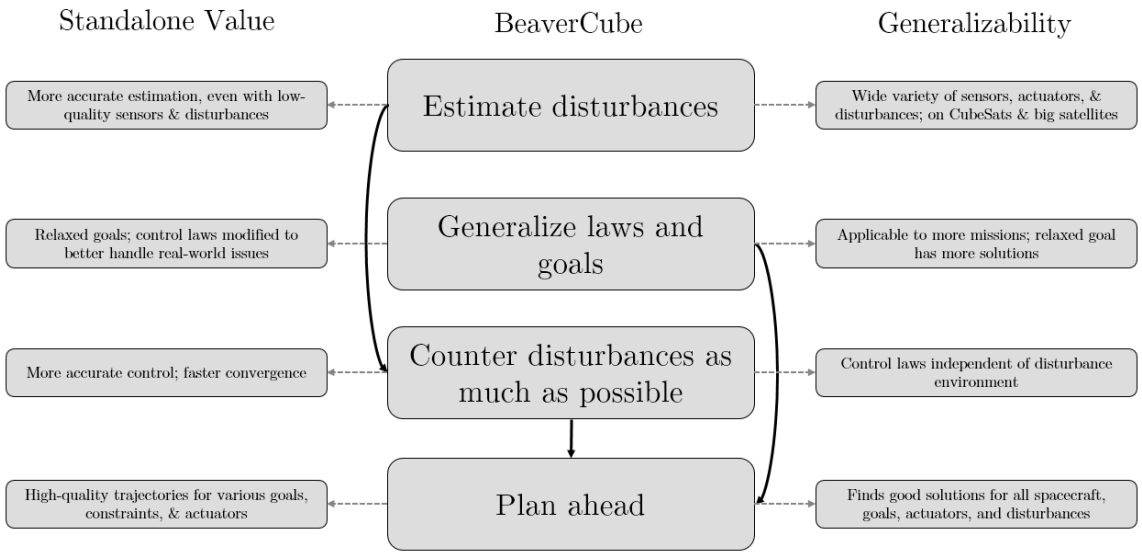


Figure 1.1: A rough overview of how the magnetic control problem break into components to be solved, along with how each of these components have value in its own right and could be kept generalizable. By keeping the solutions to these problems generalizable, they can be used on many other missions.

1.3 Contributions in Brief

This dissertation covers several areas of work that are general: improved state estimation, better and more autonomous ADCS operations and goals, disturbance modeling, trajectory planning, and a generalized approach. This is detailed further in Section 3.2. These elements combine in various ways to provide the contributions of this dissertation:

- a **Generalized Disturbance- and Dynamics-Aware Estimator** that incorporates system dynamics and individual sources of uncertainty or noise, while accounting for (and estimating) disturbances for a wide variety of satellites;
- a method for (and evidence of the advantages of) **Direct Countering of Disturbances** in spacecraft attitude control, including for underactuated systems and hard-to-estimate or time-varying disturbances;
- a **General Trajectory Planner** that can be used for various missions, goals, disturbance environments, and satellite architectures, as well as important real-world implementation details;
- a method for **Improved ADCS Autonomy** that leverages the other contributions to create a framework for ADCS that accepts various and time-varying (including rapidly changing) goals for a variety of satellites and adapts to relevant limits, constraints, and disturbances; and
- a **ADCS Architecture for Weak and Underactuated Satellites** that is general and autonomous by using the other contributions to improve pointing capability beyond

what might be expected for spacecraft with limited actuation and/or poor sensors.

The contributions are covered in greater depth in Section 3.3.

1.4 System Model Overview

This work centers on improving spacecraft ADCS. This section gives a baseline overview of the system and its components. Further details can be found in Sections 3.4 and 4.2. The spacecraft is modeled as a rigid-body system governed by rotational dynamics, incorporating the effects of actuators, disturbances, and time-dependent environmental influences. The state vector $\mathbf{x}$ represents orientation, angular velocity, actuator states, and relevant biases or uncertainties:

$$\mathbf{x}_{\text{base}} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\omega} \\ \mathbf{h} \end{bmatrix}, \quad \mathbf{x}_{\text{full}} = \begin{bmatrix} \mathbf{x}_{\text{base}} \\ \boldsymbol{\beta}_{\text{act}} \\ \boldsymbol{\beta}_{\text{sens}} \\ \mathbf{p}_{\text{dist}} \end{bmatrix}, \tag{1.1}$$

where $\mathbf{q}$ is the orientation quaternion, $\boldsymbol{\omega}$ is the angular velocity in the body frame, $\mathbf{h}$ represents stored angular momentum from actuators, $\boldsymbol{\beta}_{\text{act}}$ and $\boldsymbol{\beta}_{\text{sens}}$ are actuator and sensor biases, and $\mathbf{p}_{\text{dist}}$ encapsulates parameters for disturbances.

1.4.1 Rotational Dynamics

The spacecraft's motion follows Euler's equations, extended to include stored momentum, actuator contributions, and general disturbances:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{1.2}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\left(\boldsymbol{\tau}_{\text{act}} + \boldsymbol{\tau}_{\text{dist}} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega})\right). \tag{1.3}$$

Here, $\mathbf{J}$ is the spacecraft inertia tensor, $\boldsymbol{\tau}_{\text{act}}$ represents the commanded actuator torques, and $\boldsymbol{\tau}_{\text{dist}}$ accounts for disturbances. Note that Equation 1.3 is accurate only for a satellite without stored momentum. A more accurate description can be found in Equation 3.20.

1.4.2 Sensors and Actuators

Sensors provide noisy and potentially biased measurements of orientation, angular velocity, and other state-related quantities. These measurements are modeled as:

$$\mathbf{s} = \mathbf{z}(\mathbf{x}_{\text{base}}, t) + \boldsymbol{\beta}_{\text{sens}} + \boldsymbol{\nu}_{\text{sens}}, \tag{1.4}$$

where $\mathbf{s}$ is the sensor output, $\mathbf{z}(\mathbf{x}_{\text{base}}, t)$ is the true measurement, $\boldsymbol{\beta}_{\text{sens}}$ is sensor bias, and $\boldsymbol{\nu}_{\text{sens}}$ is random noise. More information on bias versus noise can be found in Section 3.4.3. The "true" value depends on the base system state (such as orientation and angular velocity), plus time and environmental vectors $\mathbf{v}_{\text{env}}$. For example, magnetometers output a reading based

on the spacecraft orientation $\mathbf{q}$ and the local magnetic field $\mathbf{B}(t)$. The specific functions for $\mathbf{z}$ can be found in Section 4.3.2.

Actuators generate control torques with potential biases and nonlinear behavior. The actuation torque is the sum of each of the actuators, indexed here by i:

$$\boldsymbol{\tau}_{\text{act}} = \sum_i f_{\text{act},i}(\mathbf{u}_i + \boldsymbol{\beta}_{\text{act},i} + \boldsymbol{\nu}_{\text{act},i}, \mathbf{x}_{\text{base}}, t, \mathbf{v}_{\text{env}}), \tag{1.5}$$

where $\mathbf{u}_i$ is the commanded control effort, $\boldsymbol{\beta}_{\text{act},i}$ is actuator bias, and $\boldsymbol{\nu}_{\text{act},i}$ is random noise. More information on bias versus noise can be found in Section 3.4.3. The function $f_{\text{act},i}$ encapsulates the actuator response, which depends on the net control effort $(\mathbf{u}_i + \boldsymbol{\beta}_{\text{act},i} + \boldsymbol{\nu}_{\text{act},i})$, plus the system state (most likely orientation), time, and environmental factors. The specific functions for $f_{\text{act},i}$ can be found in Section 4.3.3.

1.4.3 Disturbances

Disturbances depend on orbital conditions, such as the position vector $\mathbf{r}(t)$ and velocity vector $\mathbf{v}(t)$, as well as environmental variables like atmospheric density $\rho(t)$ (which itself depends on $\mathbf{r}(t)$ and other values like the solar cycle), local magnetic field $\mathbf{B}(t)$ (also dependent on $\mathbf{r}(t)$), and solar flux $W(t)$. Some disturbances are time-varying and depend on the state parameters $\mathbf{p}_{\text{dist}}$. An example of such is the spacecraft residual dipole. Further information can be found in Section 4.3.4.

1.5 Case Overview

Throughout this dissertation, various (simulated) spacecraft are considered, with different tests applied to them. They are detailed in their respective chapters but summarized in Table 1.1.

Table 1.1: Summary of Satellite Test Cases

Satellite	**Satellite Information**	**Section**	**Test**	**Purpose**
TRMM	3000 kg, circular 800 km orbit, 35° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.2	Baseline Estimation Test	Test estimation approach against Crassidis results in published test case.

Continued on next page

Table 1.1 – *Continued from previous page*

Satellite	**Satellite Information**	**Section**	**Test**	**Purpose**
TRMM	3000 kg, circular 800 km orbit, 35° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.3	Baseline Estimation Test 2	Test estimation approach against Crassidis results in published test case.
CubeSat	4 kg, circular 600 km orbit, 45° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.4	Cubesat Estimation Test	Test estimation approach against Crassidis results for Cubesat.
CubeSat	4 kg, circular 600 km orbit, 45° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.5	Cubesat Estimation Test 2	Test estimation approach against Crassidis results for Cubesat.
CubeSat	4 kg, circular 600 km orbit, 45° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.6	Test without Tracking Bias and Disturbance	Test effect of excluding bias and disturbance when present.
CubeSat	4 kg, circular 600 km orbit, 45° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer, added propulsion	4.4.7	Propulsion Estimation Test	Test ability to track propulsion disturbance.
6U CubeSat[a]	12 kg, circular 600 km orbit, 45° inclination, 3x each: coarse sun sensors, gyroscopes, magnetometer	4.4.8	Many-effect Test	Demonstrate the ability of the estimator to estimate and follow a wide variety of parameters.

Continued on next page

Table 1.1 – *Continued from previous page*

Satellite	**Satellite Information**	**Section**	**Test**	**Purpose**
Space Shuttle	Huge, 3x ACS thruster sets	6.5.1.a	Wie Control Law Test	Compare effect of disturbance inclusion on Wie control law in published test case.
Unknown	SSO orbit 450 km, midsize satellite, 3x MTQ	6.5.1.b	Lovera Control Law Test	Compare effect of disturbance inclusion on Lovera control law in published test case.
Ørsted	3x MTQ	6.5.1.c	Wisniewski Control Law Test	Compare effect of disturbance inclusion on Wisniewski control law in published test case.
Ørsted	3x MTQ	6.5.1.d	Wisniewski vs. Modification Control Law Test	Compare effect of modification on Wisniewski control law in published test case.
CubeSat	3x MTQ	6.5.2.a	Wisniewski Control Law Test	Compare effect of disturbance inclusion on Wisniewski control law for a CubeSat.
CubeSat	3x MTQ	6.5.2.b	Lovera CubeSat Test	Compare effect of modification on Lovera control law for CubeSat.

Continued on next page

Table 1.1 – *Continued from previous page*

Satellite	**Satellite Information**	**Section**	**Test**	**Purpose**
CubeSat	3x MTQ	6.5.2.c	Wisniewski vs. Modification CubeSat Test	Compare effect of modification on Wisniewski control law for CubeSat.
CubeSat	3x MTQ	7.5.1	Long-Term MTQ Planning Test with Multiple Goals	Determine possiblity to plan over a long time horizon and multiple goals.
CubeSat	3MTQ+1RW	7.5.2	Test with a Large Disturbance	Determine planner solution to dealing with a large disturbance.
CubeSat	3MTQ+1RW	7.5.3	Monte Carlo Tests over Various Starting Positions	Determine planner convergence for various goals and various starting positions.
CubeSat	3MTQ	7.5.4	Sequential Trajectories	Demonstrate on-line sequential planning for successive trajectories.

[a] Physical properties (but not sensors and actuators) based off ASTERIA [1].

1.6 Thesis Organization

Thesis Overview

1. **Synopsis:** Introduces the motivation and primary contributions of the dissertation, focusing on improving satellite ADCS through computational methods to enhance autonomy, generalizability, and performance in underactuated scenarios.

2. **Background:** Reviews the history and current state of satellite ADCS. Discusses

relevant literature, technical gaps in existing methods, and challenges in ADCS for small satellites.

3. **Approach and Contributions:** Describes the areas of work of this dissertation, along with the contributions which resulted from them. Describes various important technical background information.

4. **Modeling and Estimation of Spacecraft Attitude:** Details the system model and estimation methods, focusing on modularity, dynamics inclusion, and unscented Kalman filtering approaches. Demonstrates the effectiveness of the estimator through case studies.

5. **Generalizing ADCS Goals and Control Laws:** Explores techniques to adapt ADCS control laws for various satellite architectures and goals. Derives formulae for reduced-attitude control, which simplifies control requirements by focusing on key vectors rather than full orientation.

6. **Active Disturbance Rejection:** Discusses methods for identifying and counteracting external disturbances affecting satellite orientation. Shows improvements in accuracy by incorporating disturbances into control strategies.

7. **Trajectory Planning:** Presents a novel planning framework using the ALTRO algorithm to generate time-varying pointing goals. Demonstrates applications for sequential trajectory planning in underactuated systems.

8. **Summary and Future Work:** Summarizes key findings and contributions. Proposes future research directions, including further development of control methodologies and real-world applications of the framework.

Chapter 2

Background

2.1 Introduction

The first American satellite was called Explorer 1. It was an overall success, operating for roughly four months as planned and returning valuable data, including the first detection of the Van Allen radiation belt [2]. One of its biggest contributions, however, came from a failure. Within one orbit (90 minutes) after deployment, Explorer 1 stopped pointing as intended and was spinning in an unplanned manner. This was due to energy dissipation from its whip antennas and provided the first update to the theory of rigid body dynamics in 200 years [3]. The case of Explorer 1 highlights the long history of spacecraft attitude control. [1] From those earliest days, engineers and scientists have worked to control the rotational motion of their spacecraft in various ways.

For many spacecraft, their utility depends on controlling their orientation with respect to external entities. If a spacecraft spins too fast, it may exceed the structural and harnessing design limits of the vehicle or its payload, including, in the case of human spaceflight, the safety of the crew [2]. Damage or destruction may result from such excessive rates. The CONOPS (concept of operations) of many missions also requires a certain level of attitude control. Spacecraft may need to keep their solar panels pointed within a specific angle to ensure they remain power-positive. Similarly, imagers may need to maintain precise pointing at a target for specific durations to avoid smearing, while other sensors, thrusters, and radios must be aligned correctly to accomplish mission objectives. Spacecraft may need to hold a stable attitude (inertially still or with a specific rotation, such as pointing nadir) or perform slews between different orientations to meet their mission goals, often while contending with internal or environmental disturbances.

The pointing requirements of different spacecraft vary widely. For instance, JWST has an open-loop pointing accuracy of 0.1 arcseconds (3×10^{-5} degrees) [4], whereas the MinXSS 3U CubeSat achieves a pointing accuracy of 8 arcseconds (2×10^{-3} degrees). Some smaller satellites, particularly 1U CubeSats, may lack any pointing capability beyond basic rate-based detumbling.

[1] In aerospace applications, "attitude" refers to a body's rotational orientation, especially with respect to the universal and inertial frame.

[2] See Gemini 8.

Managing pointing and other rotational behavior is generally covered by the satellite attitude determination and control system (ADCS). There are two largely-separable but interdependent halves: the attitude determination system (ADS) and the attitude control system (ACS). The ACS is used to control the spacecraft's orientation, angular velocity or momentum, and other relevant parameters with respect to the world frame or other bodies. The ADS provides information regarding the spacecraft's rotational behavior that the ACS can use.

In all but the simplest spacecraft, such as spin-stabilized or free-rotating satellites, the ADS relies on various sensing hardware, including gyroscopes, magnetometers, and star trackers [5]. However, sensors do not provide perfect or complete information. Instead, attitude estimation algorithms process sensor data to derive state information. Common algorithms include the TRIAD vector estimation method, the Extended Kalman Filter (EKF), and the Unscented Kalman Filter (UKF) [6].

The ACS uses this estimated state information to control the spacecraft's orientation and angular velocity. Commonly used actuators include reaction wheels (RWs), magnetorquers (MTQs), control moment gyroscopes (CMGs), and thrusters. The ACS employs control laws or algorithms to command the necessary actuation based on the spacecraft's state information, sensor readings, and mission goals.

Due to the high cost of failure, many spacecraft rely on relatively simple attitude control methods. These include passive techniques (Section 2.2.2.a), spin-based passive methods (Section 2.2.2.b), and basic active control methods (Section 2.2.2.e). Such approaches prioritize simplicity and provable stability or convergence, features which are vital for satellites which are expensive or unique.

However, traditional attitude control methods have significant drawbacks. The systems have very limited autonomy, to the point that many satellites perform attitude changes based on dynamics calculated on the ground and then have commands transmitted to the spacecraft. This lack of autonomy limits flexibility and responsiveness, preventing accurate and rapid transitions between pointing goals. Additionally, many control systems converge slowly, taking multiple orbits or tens of hours to reach the desired state, further limiting flexibility and the possibility of using varying goals. These limitations are particularly pronounced in systems with weak or underactuated controls, such as those relying on magnetic actuation (Section 2.3.2).

The limitations of historic attitude control also present an opportunity. Many different limitations of satellite pointing and attitude control can be addressed simultaneously. By using more autonomous systems that plan an attitude trajectory, and by accounting for the specific goals of and disturbances on that trajectory, performance can be improved on many different metrics. These improvements include:

- the ability to manage different pointing goals and switch between them quickly,
- greater autonomy and robustness to disturbance,
- improved accuracy, and
- reduced actuation effort or the ability to achieve control with smaller actuators.

Such advancements could enable the effective use of underactuated systems, like magnetic control, in scenarios that would typically require full three-axis control. They could also reduce or eliminate the need for frequent momentum desaturation maneuvers. These improvements are especially relevant in the growing domains of small satellites and mega-constellations, where autonomy, fast-changing goals, flexibility, and cost-effective control solutions are critical.

The remainder of this chapter will review prior research on satellite ADCS (Section 2.2), highlighting the areas particularly relevant to this work (Section 2.3), the gaps that this work attempts to fill (Section 2.3.2), and the motivation behind doing so (Section 2.4). Chapter 3 outlines the methodology for addressing these gaps, detailing the contributions and associated technical background. Chapters 4 through 7 provide an in-depth exploration of the proposed methods and results, covering system modeling, estimation, goal definition, disturbance handling, and trajectory planning within the ADCS framework. Chapter 8 revisits the results and considers future work and extensions.

2.2 Previous Work

There is a wide variety of methods for both attitude determination and control. We will briefly cover a select subset of options for determination systems and then describe current options for attitude control in somewhat more depth. For linear systems, the separation theorem means these can be designed and studied separately. While this may not be provable in a mathematical sense for nonlinear systems (which these rotation-based systems are), considering them separately has typically worked for ADCS [7].

2.2.1 Attitude Determination

To actively control spacecraft attitude, the state that is being controlled must be determined, accounting for potential biases, errors, and other effects. For an attitude determination system (ADS), the hardware can include gyroscopes, magnetometers, sun and Earth sensors, star trackers and cameras, and radio beacons [5]. The information from these sensors can be converted into estimated attitude through a number of algorithms (some also including predicted dynamics, disturbances, or control information), including the TRIAD vector estimation algorithm, the extended Kalman filter (EKF), and the unscented Kalman filter (UKF) [6], [8]. There are many approaches to this, and the possibilities can fill a textbook [7], [9].

Early approaches rely on solving Wahba's problem, which involves finding the orientation that minimizes the error between the real-world and observed directions of two or more unaligned vectors (locations of stars, direction towards the sun, radio beacons, or a magnetic field are all possible) [10]. Various approaches to solving this have been tried, with novel methods being developed into the late 20th century and beyond [10]–[12]. If sensors are sufficiently accurate, such as star trackers, this is one of the best options to use.

However, static approaches to solving Wahba's problem neglect information about system kinematics. If an orientation is known at some time, and the rotation rate since that time can be estimated, the new orientation should also be able to be determined [8]. Such sequential methods rely on past estimates and take into account the kinematics, allowing gyroscopes

and other sensors (and potentially, as explored in this work, dynamics with control efforts and disturbances) to be included in the estimation. This is generally achieved using Kalman filters.

Kalman filters allow the incorporation of a wide variety of information, including system kinematics and dynamics, and can be used when sensors are asynchronous or missing information (such as coarse sun sensors that do not provide a full sun vector). They have a long history in spacecraft ADCS [8], [13]. Most Kalman filters for spacecraft are Extended Kalman Filters (EKFs) that represent the nonlinear dynamics and sensor equations by linearizing about the best estimate. In the last several decades, however, the Unscented Kalman Filter (UKF) has been developed [14]–[16]. This allows for full representation of nonlinear dynamics, which has great advantage for the nonlinearities inherent to attitude estimation. For this reason, UKFs have been considered for various missions [17]–[19]. There is also the square root UKF (SRUKF), a matrix-square-root-based implementation of the UKF, which avoids certain numerical issues [20].

One specific challenge in Kalman filtering is how to represent the system orientation. This can be done with quaternions, but as described in Section 3.4.1, there is the issue of linearization and the extra dimension. A number of representation options exist, with various advantages and disadvantages [21], [22]. This intersects with the difficulty of integrating quaternions over time. There are a number of approaches to numerical integration on SO(3), the mathematical group of all three-dimensional rotations [23]. Which method is best depends on the expected rotational rate and the importance of accuracy versus computational complexity. One mathematically correct approach is to use an error-state filter, working in the exponential map of the quaternion [24].

Key details in Kalman filtering are the estimates of sensor, state, and process uncertainty, which affect how the algorithm weights different pieces of information based on how accurate it expects each to be. It is crucial, therefore, to understand the noise sources for the sensors and signals being used in the estimator [25]–[28].

Many more complex approaches to ADS have been theorized, including robust filtering for detecting measurement faults [29]–[31] , active control to improve parameter estimation [32], full GPS inclusion [33], and sparse-matrix approaches [34]. Even these advanced approaches, however, are typically tied to the sensing hardware and physical behavior of a specific mission. Among the notable methods that are more generalizable, the Attitude Parameter-Calibration Kalman Filter (APKF) by Yoon et al. integrates rotational dynamics to achieve precise parameter estimation without relying on gyroscopes [35]. By exclusively using star tracker measurements, the APKF mitigates the noise challenges associated with gyroscopes but limits its applicability to missions with access to high-accuracy sensors.

System identification (SysID) and adaptive control represent alternative methodologies to traditional filter-based estimation frameworks. Instead of separating estimation and control, these techniques directly incorporate the estimation results to modify their control methods. SysID emphasizes the development and refinement of dynamic models from observational data, often requiring batch processing or iterative refinements that may not be practical in real-time operational settings. Adaptive control, in contrast, focuses on dynamically updating control strategies to compensate for changes in system parameters or unmodeled dynamics, ensuring stability and performance [36], [37]. While these methods are well-suited to applications where system dynamics are poorly understood or change rapidly, they differ

from the focus of this work, which assumes a known dynamic model and prioritizes real-time state estimation.[3] This distinction underscores the complementary nature of these approaches, with this framework addressing operational scenarios that demand robust estimation under significant resource constraints rather than model refinement or adaptive control responses.

2.2.2 Attitude Control

2.2.2.a Passive Attitude Control

On the control side of an ADCS, there is a wide variety of techniques dating back decades. The most basic control systems (other than no ACS at all) are passive control methods. These systems rely on natural phenomena to bring the spacecraft into a desired orientation and then maintain it [6], [38]. For gravity gradient control, a satellite with one axis much longer than the other two will tend towards orientations where that long axis is aligned with the vector towards or away from the body it is orbiting. Passive magnetic control is also possible, where a permanent magnet (or other source of a constant magnetic field) aligns with the Earth's magnetic field in orbit [6], [38]. Other passive control methods are possible, including aerodynamic stabilization (which works like the fins on a model rocket) and solar stabilization [6]. Most of these methods, however, have drawbacks. They typically only constrain two axes of the spacecraft, leaving the other free. They are also not very strong and often require some additional component to ensure stability and reduce oscillations, such as a damper, momentum bias wheel, or additional active control [6], [38].

2.2.2.b Spin-Based Attitude Control

A slightly more involved control method is spin-based control. A pure-spin spacecraft spins around an axis with enough angular velocity to induce gyroscopic stiffness [6], [38]. Due to the conservation of angular momentum, a spinning body will resist disturbing torques around the axes perpendicular to the angular momentum vector [5], [38]. The spacecraft geometry and selection of the spin axis are vital [38]. A rigid body with any level of damping will naturally tend towards rotation about its largest moment of inertia, or largest principal axis [38]. This requires the spacecraft to be designed like a disk or short cylinder to ensure that it is stable. Dampers are often used to reduce nutation and encourage the desired spin axis [6]. A further design consideration for a spinner spacecraft is that all items onboard, such as instruments or radios, will also be spinning at a potentially high rate. This can make the use of things like cameras and directional antennas difficult. Lastly, due to the desired gyroscopic stiffness, these satellites can be difficult to point or adjust their orientation [38].

To ameliorate some of the issues with a spinner spacecraft, a dual-spin satellite or a momentum wheel might be used [5]. In a dual-spin satellite, some portion of the satellite spins and provides gyroscopic stability, while a non-spinning portion (connected by a bearing and slipring) holds components which should not spin, such as instruments or directional radios. A satellite with a momentum wheel (often called a momentum bias wheel) has an internal flywheel spinning at a constant speed to provide the gyroscopic stability caused by

[3] It could be argued that the incorporation of the state estimation data, such as biases and disturbances, is a simple form of adaptive control.

angular momentum, allowing the rest of the satellite to have a greater set of geometries [6], [38].

2.2.2.c Active Control Overview

An active control system uses sensors to detect changes in the system and commands a response using actuators or "plants." For spacecraft attitude control, the sensors are typically those used in the attitude determination system, or the entire estimated state output by the ADS. Such an approach can be used to control all three axes of a spacecraft or to augment the previously-mentioned passive or spin-based control, to adjust for small disturbances, change the pointing, or damp undesired modes. Some options are more common for each use case, but there are a variety of plants that are possible.

While there is variation in the ADCS components, the system can typically be described (treating the spacecraft as a rigid body) by

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{2.1}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\left(\boldsymbol{\tau}_{\text{act}} + \boldsymbol{\tau}_{\text{dist}} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega})\right), \tag{2.2}$$

where $\boldsymbol{\omega}$ is the angular velocity of the spacecraft relative to the world frame (expressed in the body frame), $\mathbf{q}$ is the quaternion (see Section 3.4.1) representing the orientation of the spacecraft relative to the body frame (Hamilton local-to-global convention), $\mathbf{J}$ is the spacecraft inertia tensor, $\boldsymbol{\tau}_{\text{act}}$ represents the actuator torques, and $\boldsymbol{\tau}_{\text{dist}}$ accounts for disturbances[5], [24], [38], [39]. Note that Equation 2.2 is accurate only for a satellite without stored momentum. A more accurate description can be found in Equation 3.20.

The actuator torque $\boldsymbol{\tau}_{\text{act}}$ depends on which actuators and how many are used–this is described further in Chapters 3 and 4. Generally, however, the torque is commanded by the spacecraft (perhaps with error) based on the desired orientation $\mathbf{q}_g$, the current orientation $\mathbf{q}$, the desired angular velocity $\boldsymbol{\omega}_g$, and the current angular velocity $\boldsymbol{\omega}$. Other parameters such as biases, disturbances, and environmental information (like the local magnetic field if the actuators are magnetorquers) may also be included. The method of specifying torques based on the desired state and current state is called the control law. It can range from simple functions like B-dot to very sophisticated planners and nonlinear controllers. Several examples of this are discussed throughout this dissertation, notably in Section 2.2.2.e. For the control law, the current state (orientation, angular velocity, and perhaps more) is not exactly known, however. Instead, the spacecraft estimates these values using data from sensors or an ADS as described in Section 2.2.1. The disturbance torques $\boldsymbol{\tau}_{\text{dist}}$ will typically be a combination of torques caused by gravity gradient, drag, solar radiation pressure (SRP), satellite inherent dipole, and propulsion. These and several others are discussed in Section 2.2.6, as well as the potential to use these disturbances as part of the control (an idea which plays a significant role in this work).

2.2.2.d Active Control Hardware

Momentum systems

One common option is to use one or more reaction wheels (RW), an internally-mounted flywheel where the angular velocity can be controlled. By adjusting the angular momentum of the wheel (increasing or reducing its speed), the angular momentum of the spacecraft along that axis is adjusted by an identical and opposite amount [6]. These wheels allow for very fine control (down to 0.0001 degree) and, depending on the size of the wheel and spacecraft, can allow for comparatively fast slews [6], [38]. A related option is to use a mechanism called a control moment gyroscope (CMG), which consists of a constant-velocity wheel (like a momentum wheel) mounted on one or more gimbals perpendicular to the axis of spin [5]. By rotating the wheel, the spacecraft can torque against the wheel's gyroscopic stability and adjust its own angular momentum. This can produce large torques and thus fast slews, and have rather high (sub-degree, down to 0.001 degrees) accuracy [38]. RWs and CMGs (and momentum wheels) have some shared issues. They are also rather large and heavy (typically used on larger spacecraft) and have moving parts that can degrade and fail even when well-built [5], [38], [40]. Additionally, as they have a spinning component, they can introduce vibrations to the spacecraft, which can be disruptive to sensitive payloads. RWs and CMGs also specifically suffer from an issue called "momentum saturation." If the systems must counter a torque that has a non-zero average in the spacecraft body frame, one or more wheels will continue to speed up in the same direction and eventually "saturate" or reach a maximum angular velocity (for example, 8000 RPM [41]) which the mechanism can handle [6]. This can also occur based on repeated attitude adjustment maneuvers. At this point, the momentum stored in the wheels is maxed out along one axis and cannot be controlled in that direction. The spacecraft then must "desaturate" by actuating the wheels in the opposite direction to bring the stored momentum away from the saturation direction [38]. If done without an external torque to counter this momentum adjustment, the satellite will begin to spin due to conservation of angular momentum. Typically, the system is desaturated (sometimes called momentum dumping) by creating a scenario where, for the spacecraft to maintain an orientation, the wheel must reduce its saturated momentum. This can be done against natural torques, such as those induced by aerodynamic forces or solar radiation pressure, or by using a non-momentum based control system (usually thrusters or magnetorquers) that generate torque in the opposite direction.

Thrusters

Thrusters can also be used to control the attitude of a spacecraft. Attitude control thrusters are, in principle, identical to those used to adjust an orbit or reach space [38]. By expelling exhaust in one direction, they create an opposing force due to conservation of linear momentum [38]. The magnitude of this force is based on the mass flow rate of the exhaust and its velocity. When used for attitude control, the magnitude of the resulting torque is the product of this force and the length of the moment arm where it is applied [38]. If no orbit adjustment is desired, two thrusters mounted in opposite directions (and with opposing moment arms) must be fired simultaneously so that the net force is zero. Chemical thrusters can provide very high torque and thus quick slew rates [38]. However, they are typically less accurate (0.1-5 degrees [38]) and often cannot be controlled except for on or off [5]. Additionally, the manufacturing errors of their mounting and alignment (whether the

thrusters are for attitude control or propulsion) relative to the center of mass can be a source of uncertainty and disturbance [38]. They also require fuel, which means they have a limited lifetime [5]. The system to hold and distribute fuel to various thrusters on the spacecraft is also a consideration, as it affects the spacecraft design significantly, due to the potential volatility and toxicity of the fuels, the complexity of running fuel lines through the spacecraft, slosh, and the size and weight of the system itself. However, compared with magnetorquers, they work at all altitudes and even in interplanetary or interstellar space [5].

Magnetorquers

Magnetorquers are another option for attitude control. They are a primary focus of this thesis, based on the mission described in Section 2.4. Magnetorquers consist of electromagnets which are actuated either on and off or to various magnetic field strengths [5]. These electromagnets can take various forms, such as a coil with only air (or a vacuum) within, or a coil wrapped around a soft magnetic material to amplify the field [6]. When actuated, these electromagnets create their own magnetic fields, which interact with the Earth's magnetic field and create a small torque attempting to align the two fields (like a compass) [6]. Specifically, if the magnetorquers create a magnetic field equivalent to a magnetic dipole with magnetic moment $\mathbf{m}$, they will create a torque $\boldsymbol{\tau}$ defined by

$$\boldsymbol{\tau}_{\text{act}} = \mathbf{m} \times \mathbf{B}, \tag{2.3}$$

where $\mathbf{B}$ is the local magnetic B-field (flux density) from the Earth (or theoretically another body like Jupiter) and $\boldsymbol{\tau}_{\text{act}}$ is the control torque in Equation 2.2 [5].

Magnetorquers have no moving parts and do not require fuel, so they are highly reliable and have effectively unlimited life [5]. They are not sensitive to the location of the center of mass [5]. Furthermore, they adjust angular momentum directly (as opposed to storing it), so they do not saturate like reaction wheels. They are comparatively low cost, size, weight, and power, and do not produce structural vibrations. These advantages are a motivating factor behind their increased use, especially among small satellites.

However, they have significant drawbacks. The two most significant disadvantages are their strength and underactuation. The maximum torque they can produce is the product of the generated magnetic moment and the local B-field strength. The Earth's magnetic field is relatively weak, meaning most generated torques are rather small or require high current (and thus power) [5], [38]. For a magnetorquer with a small maximum magnetic moment (such as 0.2 Am^2), the maximum torque may be in the tens of micro-Newton-meters. Bigger magnetorquers are possible, but the torques remain small compared to reaction wheels, thrusters, and CMGs, meaning slow slews (especially for larger spacecraft) and potential difficulty countering larger disturbances. The underactuation is another significant concern. Torque cannot be produced that is parallel to the local magnetic flux density $\mathbf{B}$, and thus the magnetorquers cannot torque around the local field lines [5]. This results in two degrees of control for a system with three degrees of freedom, leaving it underactuated [6].

The challenges of working with a weak and underactuated system are some of the problems that this dissertation explores. Complexities of magnetism are another disadvantage of magnetorquers, as the satellite may have its own magnetic dipole (which can cause a

varying disturbance as it is scrambled by the magnetorquer actuation), the rods in the magnetorquers may have magnetic dipoles even when the system is turned off, and the structure of the satellite can deform the field generated by the magnetorquers. Further, the magnetic field produced by the magnetorquers can create eddy currents inside the satellite and induce other magnetic effects which can interfere with or damage sensitive instruments and, in more extreme cases, the satellite avionics themselves [5]. Finally, the necessity of the magnetic field of the Earth (or another body) affects the use of this system. It is weaker at higher altitudes and cannot be used around the moon, in deep space, or in other locations with negligible magnetic fields [5]. The planned orbit for the spacecraft must be considered not only for these reasons but also to understand the impact of how the local magnetic field will change throughout the orbit (Earth polar orbits allow more control than equatorial, as the direction of the B-field varies) and the potential for the field to deviate from models.

Other Actuators

There are several other potential methods for active control of spacecraft attitude. Most are new or rarely used. One potential method is fluid dynamic actuators, where a fluid moving through a loop formed of tube that can operate similarly to a reaction wheel [6]. This has a number of advantages over a reaction wheel but is a novel technology with its own challenges. Another potential method of attitude control is mass movement. If a spacecraft has appendages, deployables, or internal actuated masses, their motion can be used to adjust the center of mass of the system as a whole and induce rotation [5], [6]. There is also the potential to use environmental torques for a more active control system, which is further explored in this thesis. The spacecraft could also actuate various components to use these torques more deliberately, such as movable aerodynamic control surfaces [6]. A similar system, using the adjustment of the reflective properties of static surfaces, could be used with solar radiation pressure.

2.2.2.e Active Control Algorithms

The actuators in section 2.2.2.d are simply tools. For successful active control, they must be used in an appropriate manner determined by the control system algorithm. These are the control laws for finding $\boldsymbol{\tau}_{\text{act}}$ in Equation 2.2. There are many approaches to control algorithms, especially for modern nonlinear control (relevant to rotational cases like this one). However, few are used for spacecraft attitude control. There may be others that are classified or proprietary, but there are three kinds of control algorithms that are commonly discussed for spacecraft ACS. Markley and Crassidis provide a good overview [7]. In their Chapter 7, they derive two control laws which sufficiently capture most of the attitude control algorithms used today. First, they consider the attitude regulation case, where the control law attempts to drive an orientation to match another (as is common, they use quaternions to represent orientation and are, for the example, driving the quaternion to the unit quaternion without loss of generality). They first derive a simple linear feedback control for external torques as

$$\boldsymbol{\tau}_{\text{act}} = -k_p \delta \mathbf{q}_v - k_d \boldsymbol{\omega}, \tag{2.4}$$

where $\delta\mathbf{q}_v$ is the vector component of $\bar{\delta q} = \bar{q}_g^{-1}\bar{q}$, the error in orientation quaternion versus the goal orientation quaternion, $\boldsymbol{\omega}$ is the angular velocity (the desired angular velocity is assumed to be zero), and k_p and k_d are gains chosen to control the system. Working from Wie and Barba, they prove the stability of this method with Lyapunov analysis [42]. Some of this work is further explored in several papers by Wie [43], [44]. Markley and Crassidis also provide several extensions, such as

$$\boldsymbol{\tau}_{\text{act}} = -k_p \text{sign}\left(\delta q_0\right)\delta\mathbf{q}_v - k_d\boldsymbol{\omega} \tag{2.5}$$

$$\boldsymbol{\tau}_{\text{act}} = -k_p\delta\mathbf{q}_v - k_d\left(1 \pm \delta\mathbf{q}_v \cdot \delta\mathbf{q}_v\right)\boldsymbol{\omega} \tag{2.6}$$

$$\boldsymbol{\tau}_{\text{act}} = -k_p\left(\delta q_0\right)\delta\mathbf{q}_v - k_d\left(1 \pm \delta\mathbf{q}_v \cdot \delta\mathbf{q}_v\right)\boldsymbol{\omega} \tag{2.7}$$

where δq_0 is the scalar component of the orientation quaternion error and $\cdot$ is the vector dot product. They also extend some of these laws to reaction wheels. All of these control laws are Lyapunov control laws that are similar or identical to proportional-derivative (PD) control laws. PD control laws and their cousin, proportional-integral-derivative (PID) control laws, are the only types of three-axis pointing control discussed in System Mission Engineering: The New SMAD [38]. The CubeSat Handbook covers them as well [6].

The second control law described by Markley and Crassidis is a sliding mode control law [7]. This is also the second type of three-axis pointing control discussed in The CubeSat Handbook [6]. Markley and Crassidis apply the sliding mode control to a tracking case, where the spacecraft is supposed to follow a time-varying trajectory (in terms of orientation). The derivation of this sliding mode is more intricate and not replicated here. PID/Lyapunov and sliding mode control laws (and variations on them) appear to be the standard set of pointing control laws, based on various entries and dedicated texts, as well as conversations and correspondence with several experts in academia and industry. The ones described here and in the sources require full three-axis control and are therefore not applicable to magnetorquers. There are some different methods for magnetorquers that have been theorized or flown. Those will be described further in subsection 2.2.3.

There is the potential for control laws involving a minimization of a linear quadratic regulator (LQR) or a planned trajectory. Markley and Crassidis touch on these by mentioning the possibility of using an H_∞ controller [7]. Other, more complicated, algorithms and control laws have been proposed [45], [46]. However, there appears to be little serious discussion of the usage of these more complicated methods for attitude control in common missions, in either the texts or the author's conversations with relevant experts. Some relevant published or theoretical examples are discussed in Section 2.2.4.

2.2.3 Magnetic Control

Much of this dissertation is focused on magnetic control. As discussed in Section 2.2.2.d, magnetic control through magnetorquers has many advantages but presents a specific set of challenges: the weak maximum torque and the underactuation due to the inability to torque around the magnetic field line. Due to their advantages, there have been many approaches to try to overcome these difficulties. Ovchinnikov and Roldugin present a very comprehensive survey of different approaches [47]. Many methods discussed here are specifically for attitude acquisition or regulation control, where a specific attitude is maintained. In many cases,

these are reasonably simple to modify for tracking, where the goal orientation is time-varying and the goal angular velocity is the rotation rate of the goal orientation. The remainder of this section describes some specific approaches to magnetic control, but some result papers on three-axis magnetic control of flown satellites should be mentioned. The satellites GOCE and PRISMA TANGO are two notable examples [48], [49].

B-dot

Among the simplest magnetic control algorithm (and the only one common across introductory texts) is the B-dot angular rate algorithm [5]–[7], [38]. Lovera presents a useful description and analysis of this long-used algorithm [50]. Ovchinnikov devotes an entire section of magnetic control to it [47]. The B-dot algorithm is defined by the expression (or one similar to it)

$$\mathbf{m} = -k\dot{\mathbf{b}} = -k\left({}_{\text{body}}R_{\text{ECI}} \frac{d\,\mathbf{b}_{\text{ECI}}}{dt} - \boldsymbol{\omega} \times \mathbf{b} \right), \tag{2.8}$$

where $\mathbf{m}$ is the magnetic moment control output in the body frame, k is some gain constant, $\mathbf{b}$ is the measurement of the Earth's magnetic field in the body frame, with $\dot{\mathbf{b}}$ as its time derivative, and $\mathbf{b}_{\text{ECI}}$ as the local magnetic field in the world or Earth-centered inertial (ECI) frame. The magnetic field $\mathbf{b}_{\text{ECI}}$ is varying as the satellite moves through it, but it typically varies quite slowly, so the $\boldsymbol{\omega} \times \mathbf{b}$ term dominates. Due to the function of the cross product and the magnetorquer actuation in Equation 2.3, this will result in loss of angular momentum over time (except when $\boldsymbol{\omega} || \mathbf{b}$, when it will have no effect, but this is transient in any real system). As the magnetic field varies throughout the orbit, the angular velocity will eventually be countered entirely (or until it matches ${}_{\text{body}}R_{\text{ECI}} \frac{d\,\mathbf{b}_{\text{ECI}}}{dt}$). Thus, in this instance, the underactuation is not a problem, though it is clearly slower. This algorithm is a common usage of magnetic control but does not achieve pointing, only succeeding in damping the angular rate. It is frequently used to detumble spacecraft as it can also be done with no attitude knowledge but just time-differencing magnetometer readings. It remains significant, and new ideas continue to try and gain marginal improvements on it through dynamics tricks, including the $\frac{d\,\mathbf{b}_{\text{ECI}}}{dt}$ component, using a matrix instead of scalar gain, and more [47], [50]–[52].

Environmental Stabilization

For pointing control algorithms, the underactuation issue becomes more significant. One set of approaches is to use a predictable disturbing environmental torque as an extra degree of control. A combination of local magnetic control with the effect of the gravity gradient torque was used to stabilize the Ørsted satellite to four specific equilibrium points [51]. This approach was taken further by Lovera and Astolfi, who used gravity gradient combined with a scheduling gain and the magnetic field averaged over the orbit to create a feedback control law that would eventually stabilize to an arbitrary orientation [53]. In fact, their approach is exponentially stable. Similar approaches are possible using aerodynamic drag and solar radiation pressure [39], [54]–[56]. These approaches are dependent on satellite geometry and, for many, the desired attitude. Several further examples are described by Ovchinnikov [57].

General Feedback Control

A more universal approach is to use feedback controllers, similar to the Lyapunov/PD controllers in Section 2.2.2.e, adapted for magnetic control. Lovera and Astolfi demonstrate that their simple approach is to use a law very similar to Equation 2.4, but then remove any component that is parallel to the magnetic field, essentially projecting it onto the plane perpendicular to the magnetic field, where there is full control [58]. In principle, this is similar to the B-dot control algorithm (in fact, the damping terms are equivalent), where the torque applied is projection of the ideal rate-countering torque onto the plane perpendicular to the B-field. This result is locally exponentially stable and is slightly generalized by the introduction of the tensor of inertia and a proof of global stability in a later work [58], [59]. Many approaches since then have worked to optimize and improve this result. One literal optimization of the method was to optimize the gains using the H_∞ method [60]. A feedback control approach was used as part of the control system of the PRISMA mission's TANGO satellite (though it was an approximation for a constant solution to an LQR solution) [48]. Similarly, a basic feedback controller was the basis of the attitude control for the GOCE satellite. A three-axis correction torque was generated, modified to counter predicted disturbance and gyroscopic torques, then projected onto the plane perpendicular to the magnetic field [49]. A Lyapunov/PD-control variant specialized for magnetorquers (working identically to the projection of the Lyapunov RW control law in The CubeSat Handbook [6] onto the plane perpendicular to the magnetic field) has also been developed. Its stability to disturbances, as well as mass and estimation errors, has been shown [61]. The examples mentioned here are just a sampling. Many other conceptually similar approaches are described by Ovchinnikov [57].

Sliding Mode Control

Applying the feedback control from Section 2.2.2.e to magnetic control typically involves taking a full three-axis control algorithm, finding the resultant torque, then removing the components which a magnetic system cannot generate. The other common approach, the sliding mode control, is more complicated but lends itself more readily to magnetic control, as it can more easily handle constraints. Sliding mode control is a robust control strategy that works by continuously adjusting the control input to keep the system on a specific trajectory called the "sliding surface." Instead of explicitly canceling out disturbances or modeling complex dynamics, Sliding mode control uses a switching control law to adaptively compensate for uncertainties and external influences. Once the system reaches the sliding surface, it remains on this surface (if full actuation is possible), even in the presence of disturbances, resulting in a resilient for nonlinear systems. The foundation of this approach for magnetic control comes from the late 1990s, when Wisniewski developed a version for the Øersted mission [62]. At around the same time, a different sliding mode controller was being developed by Wang and Shtessel [63]. A more recent approach takes this method and uses a higher-dimension "super-twisting" sliding mode algorithm to allow for the modes to be entirely continuous [64]. This approach was also used in the PRISMA TANGO satellite for successful on-orbit magnetic control [48], [65]. A method that claims to allow full magnetic

control via a sliding surface (but does not adjust for disturbances) has also been created [66]. Sliding mode control has been extended with fuzzy logic and adaptation [67]. Many other variants have been proposed as well [47], [68], [69].

Reduced Methods

Two approaches with a simple elegance are worth mentioning, particularly because they are relevant to later topics of this dissertation. The existing use of planned trajectories will be discussed in depth later in Section 2.2.4. One approach that is not a fully planned trajectory but is worth considering is to use two-step rotations. As any rotation can be composed of two perpendicular rotations (as they are on the $SO(3)$ group), a given rotation with a component along the B-field can be split into two rotations with no components on the B-field. Inamori, et al. has suggested this for use, even around the magnetic equator [70]. Another set of ideas to counter the underactuated nature of this problem is to reduce the degree of the target. As there are only two degrees of control, using a goal that is only defined in two degrees can simplify the problem. This can be done by setting the goal as aligning a body vector with a direction in the world frame, as opposed to specifying a full orientation. This approach, called reduced attitude or single-axis pointing, will be discussed in more detail in the coming sections. However, it is worth mentioning that there have been several approaches for this that are explicitly about magnetic control [71], [72]. Additionally, there are several approaches that worked to align a vector with the magnetic field or towards the sun [57], [73]–[75]. These approaches could be easily modified to work for an arbitrary axis.

LQR-based Approach

There have also been significant attempts to use optimization to find effective strategies for magnetic control in space. Using solutions to various Riccati equations or similar linear quadratic regulator problems is a time-tested method of finding solutions to various linear and nonlinear problems. For this purpose, they should also be grouped with attempts which use finite receding horizon control and infinite horizon control. In the case of magnetic attitude control, there have been a number of attempts, dating back over two decades. A finite horizon periodic controller was presented by Wisniewski as it was studied for the Øersted satellite [62], [76], [77]. This was further developed into an infinite horizon periodic controller with Markley [78]. Another receding horizon controller was developed that used a pseudospectral method that was significantly more computationally efficient than solving the Riccati equation [79]. A similarly-motivated approach found an efficient algorithm for solving a Riccatti equation with a periodic nature (which is applicable here due to periodic orbits) [80]. Other LQR approaches include a foundational approach by Psiaki which was more recently extended to include magnetorquer saturation [39], [81]. Finally, an LQR approach was also used successfully as part of the PRISMA TANGO mission [48], [65]. Optimization of magnetic control trajectories has also been attempted with the H_∞ method [82], [83]. Many other variations on optimization have been attempted [47].

Model Predictive Control

There have also been significant attempts to use optimization based on trajectory planning to find effective strategies for magnetic control in space. These concepts will be further described in Section 2.2.4, but those that are focused on magnetic control are worth discussing here. One common approach is the use of model predictive control (MPC). This has already been mentioned as an approach to a more efficient variant on B-dot [52]. It has also been applied to pointing control. An early attempt at magnetic trajectory planning used the RIOTS optimization program to generate an optimal magnetically controlled trajectory for a time-optimal slew, and then used single-step model predictive control (minimizing weighted deviations from the trajectory at the next step) to extend the generated trajectory to closed-loop control [84]. Using MPC for a whole trajectory segment has also been suggested, including consideration of the external disturbances [85]. An explicit MPC controller has also been devised which allows for MPC to be applied in a receding horizon manner, with most of the calculations done in advance. This approach compared very favorably in accuracy and power to a magnetic angular rate damper and the basic magnetic feedback controller as described previously in this section [86]. One last optimization method worth mentioning is the ALTRO (Augmented Lagrangian TRajectory Optimizer) trajectory optimization algorithm, which is connected to a large portion of this work. It has been applied once to magnetic control before, as it was tested for 180-degree slews [87].

2.2.4 Trajectory Planning

One possible method of attitude control for spacecraft centers on trajectory planning. Feedback control laws and sliding mode controls respond based on the current state and try to reach some goal. LQR and H_∞ approaches find control gains or laws that optimize some cost or performance, then apply them to a given state. Trajectory planning approaches instead generate a path through orientations not just for the next time step, but for some time period after. These methods have an obvious advantage in nonlinear and underactuated cases (like rotation and magnetic control), as they allow for control solutions that may not optimize in the immediate time step but, combined with the rest of the trajectory plan, perform better. Many of these methods can be quite computationally complex and may not be guaranteed to find a solution. They are more common in robotics and autonomous vehicles, but much rarer on spacecraft, especially for attitude control [88]–[92]. Typically, these control algorithms require a feedback controller or other general control law to reject disturbances and keep the system on the planned trajectory.

While many planned trajectory approaches involve complicated optimizations across trajectories, some use much simpler rules that meet the goals of a slew or other movement. For example, there is a simple trajectory that can be generated to slew a spacecraft from an arbitrary orientation so that its imager is pointing at the target, while maintaining communication and avoiding pointing the imager at a bright object (such as the sun) [93]. A more recent approach to a similar problem finds two separate, sequential rotations that will align a body vector with a direction in space, while avoiding a keep-out area and restricting torque to a plane (such as for magnetic control) [94]. A similar two-step method has been proposed to reach an arbitrary orientation, even for restricted magnetic control [70].

More advanced trajectory planning has been proposed using model predictive control (MPC), where a planned trajectory is modeled and optimized. This allows for better solutions

to problems with greater constraints and in highly nonlinear environments. Such an approach can also be modified to weight different aspects of the problem differently, allowing for solutions with different qualities, such as minimum-time, least-actuation-effort, or weighted functions balancing the endpoint values, completion time, and control effort and state values over the trajectory. Exploring such solutions has become more feasible and gained more attention with recent developments in robotics and computing, but the idea goes back more than 20 years, when it was successfully applied to the problem of slew rotations using thrusters restricted to two axes. This approach worked in function space by using a best-step Newton-type search algorithm [95].

In a nod to the computational complexity of such approaches, modifications have been proposed that move most of the computation offline, with only simple adjustments made during use. This method, termed explicit MPC (eMPC), was applied in a receding horizon manner, solving an optimal control problem for a future finite-length trajectory at each time step. It was used on magnetic control of a CubeSat. This approach works by linearizing the dynamics about a set point and using a quadratic cost function to approximate an LQR solution over time, with dynamics enforced by slack variables, and relying on a control gain matrix after some intermediate time horizon. This can be written as a convex optimization problem and then transformed into a multiparametric quadratic programming problem, which can be optimally solved through various means (taking advantage of Karush-Kuhn-Tucker conditions). This solving can be done offline. The solution that is found can then be used in real-time to find an optimal control gain and additive control vector for each time step in the finite trajectory. This approach compared very favorably in accuracy and power to a magnetic angular rate damper and the basic magnetic feedback controller as described previously in this section [86]. Using MPC for a whole trajectory segment has also been suggested, including consideration and internal modeling of the external disturbances. In this approach, the system was controlled via open-loop MPC. This is run at regular intervals based on an updated estimated state. By including disturbances in the planning, the idea is that open-loop will be sufficient [85].

Another early attempt at magnetic trajectory planning used MPC in an interesting manner. This approach used the RIOT (Recursive Integration Optimal Trajectory Solver) optimization program to generate an optimal magnetically controlled trajectory for a time-optimal slew, and then used single-step model predictive control (minimizing weighted deviations from the trajectory at the next step) to extend the generated trajectory to closed-loop control. This reduced the complexity required for MPC and increased the validity of linearization and other approximations due to the short time step considered [84]. There have been several recent proposals to use MPC in underactuated spacecraft, when reaction wheels have failed. Several have focused on either countering the disturbance by solar radiation pressure (SRP) or using that disturbance for three-axis control. One used model predictive control with a receding horizon to solve linear programming problems that served as the closed-loop control function for a greater planned trajectory that was supposed to prevent drift from SRP [96]. The use of MPC has also been explored to use that same SRP disturbance to fully control and stabilize an underactuated satellite [97]. This was taken even further when it was shown that an underactuated satellite with two reaction wheels could be fully stabilized and controlled using MPC [98]. This work appears to stem from earlier work on MPC and other algorithms that have been modified to apply to attitude control, indicating these problems likely reduce

to solvable quadratic programming problems [91].

These and other approaches have been successful. There have also been approaches from the field of robotics that have created trajectory-optimization algorithms capable of finding and generating orientation trajectories that match goals, even in the underactuated case [82], [84], [86]. A specific example highly relevant to this thesis is the ALTRO algorithm and its associated variants. ALTRO is a trajectory optimization method that uses an augmented Lagrangian to enforce constraints. The outer loop weights the cost of a trajectory based on the relevant cost functions and adds to the cost for violated constraints using a Lagrange multiplier and a penalty method. The inner loop works to optimize that augmented cost while the penalty and Lagrange multipliers remain set. When the inner loop has sufficiently optimized (or run through sufficient time steps), the outer loop will check for violations and adjust penalties and Lagrange multipliers, with the penalty increasing with each loop. The inner loop optimizes the trajectory under the given multipliers through an iterated LQR approach, finding the second-order expansion of the cost-to-go for each step in a backward pass, then working forward over time (with regularization, if needed) with the values of the expansion to correct the control term at each step. This is repeated until it converges [88], [99]. While not originally developed for attitude control, ALTRO has been extended for this purpose [89]. It has been expanded and updated for more general problems, especially with a conic formulation of constraints [22], [100]. It has also been applied to magnetic control for simple slews, using a time-varying back-calculated LQR for the closed-loop control portion [87].

2.2.5 Reduced Attitude or Single Axis Control

Many spacecraft with active attitude control attempt to maintain a specific three-dimensional (3D) orientation in space [7], [43], [53], [101], [102] However, in many cases, controlling a spacecraft to a specific 3D orientation is unnecessary. Consider a satellite with a nadir-pointing camera. If it points nadir, the goal is met, independent of rotation around the nadir vector. This is applicable to many spacecraft goals, such as pointing a camera at a target, an antenna at a ground station, or solar panels at the sun.

In such cases, a different approach can be taken, where the goal is not a specific orientation or quaternion, but an entire set, related by rotation around the target vector [46]. An example of the various orientations that all meet a requirement can be seen in Figure 2.1. Such approaches can be called "single-axis pointing" or said to rely on a "reduced attitude" formulation. When such such an approach is used, the system is simplified but needs to be thought of differently, with different formulas for kinematics, dynamics, and control [103]. Many familiar techniques, however, can be applied to such systems, such as PD controllers [104]. This can be extended further to control laws that are robust to various uncertainties [105]. This approach can also be combined with a keep-out zone to derive simple trajectory plans [93]. More complex approaches have also been used, such as higher-dimension approaches that simultaneously stabilize the full attitude (not just reduced) or creating control laws based around single-axis pointing with complex dynamic constraints [46], [106].

When working with underactuated systems, such as magnetically controlled satellites, there is an appealing symmetry to the reduced-attitude approach. The system has three dimensions, but the control only has two. If the goal can also be reduced to two, there is the

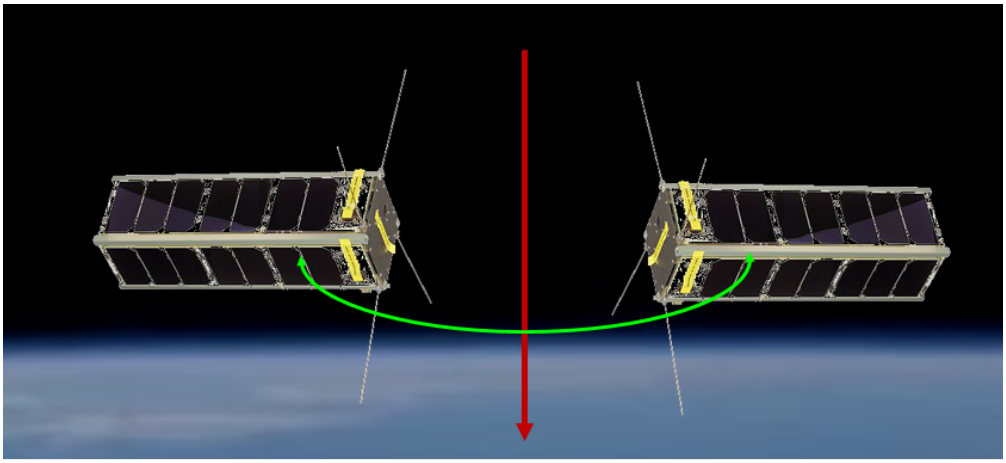


Figure 2.1: A diagram showing how a single-axis or reduced attitude control goal can permit various orientations while still meeting the pointing goal. Image component source: BeaverCube CDR CAD [107]

possibility of simplifying the problem or improving performance. This idea has been applied to a a general underactuated system for the purpose of pointing while avoiding a keep-out zone [94]. Other authors have applied it directly to magnetically-controlled spacecraft. Control laws for such a situation and a spherical spacecraft can be derived analytically [71], [72]. If the pointing direction in the inertial frame is aligned with the magnetic field vector, such an approach will work very quickly [73]. A common application of such an approach is to point the spacecraft's solar panels at the sun, as this is an operational safe mode that will keep the spacecraft power positive [57], [74], [75]. These also have the advantage of being able to run with fewer sensors (such as solar panels sensing the sun), allowing for simple control algorithms. However, the vector to the sun has no special significance in terms of attitude dynamics and could be replaced with an arbitrary vector combined with a sufficient and converged ADS.

2.2.6 Disturbance Modeling in Attitude Control

One of the key functions of satellite attitude control is disturbance rejection. Even if a trajectory planning method is used, disturbances will occur and nudge the spacecraft off the planned trajectory. These small deviations will grow over time due to other disturbances and because the nonlinear nature of rotational control will mean the planned actuation now has different effects than expected.

To many people, space seems empty, so the presence of disturbances may sound surprising. This is somewhat true, compared to being in an atmosphere (and is even more true for interplanetary space than it is for low-Earth orbit (LEO)). However, the (relative) lack of atmosphere also means there is very little damping or resistance, so even weak forces and torques can, over time, have significant effects. Some sources of disturbances can be internal from mass movement and other sources, such as the motion of appendages and deployables or a small imbalance in a reaction wheel [38]. These disturbances cannot be eliminated, but can be reduced through careful design and analysis, and they tend to be

cyclical (otherwise they would violate conservation of momentum). Internal systems can also cause an "external" disturbance torque, caused by energy or mass leaving the system. Thrusters are a common source of this, and can cause a disturbance if they are misaligned, have off-axis thrust, if the satellite center of mass is different than expected, or if a pair of thrusters for attitude control do not have matched thrust (or direction) [5], [38]. Similar, typically smaller, disturbances can come from leaks of propellant or another fluid, thermal radiation, or off-gassing from components. These may have larger effects on very precise and long duration missions or missions where attitude correction is supposed to occur at a slower rate. There are also purely external disturbances, though their magnitude and effect can vary based on the spacecraft orbit and location. Typically, there are four purely external disturbances which are discussed:[5], [38], [108]

- Aerodynamic drag effects can be significant if in lower Earth orbit or around other bodies with atmosphere. This torque is caused by a difference between the spacecraft center of mass and center of pressure.

  The magnitude can vary greatly, scaling linearly with the cross-sectional area of the satellite [108], decaying exponentially with altitude [109] (though it also depends on the solar cycle), and increasing quadratically with orbital velocity [108]. It is highly dependent on the relative distribution of spacecraft mass and surface area, and the satellite's alignment with its direction of motion. Example values for small satellites could be anywhere from 10^{-12} Nm to 10^{-6} Nm [38]. For a large satellite between 400 km and 1000 km, the magnitude could range from 10^{-7} Nm to 5×10^{-3} Nm [110].

- Magnetic torque can occur if the satellite is in an external magnetic field (such as the one around the Earth). The effect can be caused by currents in the spacecraft, by inherent magnetism in spacecraft components, or residual magnetism in magnetorquers.

  The magnitude decays according to an inverse cubic with altitude [109], and is inherently dependent on the spacecraft's own magnetism (which relates to construction, materials, and scale). The residual dipole on the spacecraft could be 0.1-20 Am^2 or even higher [38]. It is also dependent on the spacecraft's alignment with the local magnetic field, which can vary throughout an orbit. Example values for small satellites could be anywhere from 5×10^{-7} Nm to 2×10^{-5} Nm [38]. For a large satellite between 400 km and 1000 km, the magnitude could range from 10^{-4} Nm to 10^{-3} Nm [110].

- Gravity gradient torque is caused by the fact that gravity decreases with distance and will slowly torque orbiting spacecraft until their axis of minimum inertia is parallel to the local vertical [5].

  The magnitude can vary greatly, scaling linearly with the largest value of the spacecraft inertia matrix [111] and decays according to an inverse cubic with altitude [109]. If the angle between the satellite's major principal axis and the local vertical is θ, gravity gradient torque scales by $\sin(2\theta)$. Example values for small satellites could be anywhere from 3×10^{-7} Nm to 4×10^{-5} Nm [38]. For a large satellite between 400 km and 1000 km, with a misalignment of 1 degree between the spacecraft principal axis and the local vertical, the magnitude is about 10^{-3} Nm [110].

- Solar radiation pressure is caused by light from the sun being absorbed or reflected by spacecraft components, thus transferring momentum. (This is like a solar sail, but when thinking about attitude, it is less of a sail and more of a weathervane.) It can be thought of as working like aerodynamic effects, in the sense that each component of the spacecraft may absorb a different amount of momentum and cause a torque. However, for solar radiation, the surface properties of each component have a much greater effect, as they affect how light is reflected, absorbed, or scattered [108].

  The magnitude can vary greatly, scaling linearly with the cross-sectional area of the satellite [108]. It is generally not dependent on altitude [109]. It is highly dependent on the relative distribution of spacecraft mass and surface area, and the satellite's alignment with the sunward direction. Example values for small satellites could be anywhere from 10^{-6} Nm to 7×10^{-6} Nm [38]. For a large satellite between 400 km and 1000 km, the magnitude could be about 2×10^{-5} [110].

There are other sources of disturbances, such as the effect of solar wind and things like radiation pressure from Earthshine, but these four are the standard external disturbances to consider [5], [38], [108]. Impact by debris or micrometeroids could also be considered a disturbance, though this is typically not considered, as attitude control is not the primary issue if such an event occurs. "Pseudo-disturbances" are also worth mentioning, which do not impart a real torque on the system, but can be a source of deviation that active control must absorb and correct. These are typically caused by differences between the ideal or expected control and the real-world actuated control. Sources include angular misalignment of actuators, variations in external magnetic fields in the case of magnetorquers, error or noise in actuation, actuation dynamics (like reaction wheel speed control or back EMF), inertia matrix error, or estimation error.

Effective control laws should be able to handle disturbances, if the actuation torque is sufficiently stronger than that of the disturbance. Thus, many control laws (such as feedback and sliding mode methods) simply counter disturbances as they arise [7], [62], [63]. Other methods try to "learn" the disturbances over time (assuming they are periodic) and account for them in future orbits while refining their estimates [112].

However, many of the external disturbances can be well-estimated, if certain parameters are known [5], [108]. Gravity gradient torque can be derived from the inertia matrix combined with the spacecraft orientation and orbit; aerodynamic drag can be estimated from orbital velocity, spacecraft geometry, orientation, and atmospheric density; solar radiation pressure from solar flux with spacecraft geometry, orientation, and surface properties; and magnetic dipole with spacecraft magnetism and local magnetic field [5], [108]. (Of these, magnetism appears to be the most challenging, as the residual dipole of the satellite cannot be easily predicted and can vary over time. The others can be approximated from base principles.) By studying the magnitude, dynamics, and timing of these torques, spacecraft orientation stability can be understood. As early as 1970, there were considerations on whether there are stable equilibrium orientations for satellites under aerodynamic drag and gravity gradient (there are if drag is conservative) [56]. Understanding the stability of satellites under aerodynamic drag and solar radiation pressure is part of ongoing research [113], [114].

These disturbances can be further utilized in control algorithms. Their values can be estimated and included in control algorithms and the "disturbance rejection" control

component can be used to correct for disturbance model errors and unmodeled disturbances. If done correctly, this will reduce the magnitude of unpredictable effects that must be rejected and corrected, improving performance. One simple and common approach is to include the gravity gradient torque (simplest to model and often significant for LEO satellites) in the attitude dynamics model used to build the control algorithm [45], [53], [65], [80], [82]. Further disturbances can be included. In the PRISMA TANGO ADCS, the spacecraft residual dipole is included as an estimated state and predicted over time. When in sliding mode control, the TANGO system algorithm determines a three-axis torque, from which gravity gradient torque, estimated residual dipole torque, and gyroscopic effects are removed. This torque is then projected into the plane perpendicular to the magnetic field [65]. The GOCE mission uses a similar method, starting with a three-axis feedback control law, from which predicted disturbances and gyroscopic effects are removed, before it is converted to magnetic control [49]. This has been extended to many other methods. The pitch torque caused by an elliptical orbit can be modeled and included in an LQR problem, which can then be solved with a pseudospectral method [79]. A later analysis on GOCE mission models indicated that including drag and thruster disturbances in an MPC controller improved accuracy. Interestingly, a simple constant disturbance model did about as well as a more complicated harmonic model [85]. Including gravity gradient, aerodynamic drag, and residual magnetic disturbance torques in dynamics has shown improvement for H_∞-based magnetic control [82]. Solar radiation pressure disturbances, often neglected, have also been studied, and an MPC receding horizon method has been found that can directly counter this disturbance (and resulting drift) in both fully and underactuated cases, with one, two, or three reaction wheels [96].

Perhaps the most interesting possibility is not simply accounting for disturbance torques, but using them to improve magnetic control by addressing the underactuated nature of the problem. There have also been considerations of how the gravity gradient disturbance can be used to provide a stabilizing influence to magnetic control laws. In particular, this was part of the attitude control for the Ørsted satellite which used gravity gradient along with magnetic control to keep the satellite in up to four orientations and switch between them [51]. More recently, it has been shown that satellites with significant gravity gradient effect (those with a boom, especially) can be stabilized to an arbitrary orientation using magnetic control [53]. Other work has focused on a non-magnetic, but underactuated (only two reaction wheels) system, combined with solar radiation pressure. Both linear-quadratic control and MPC have been explored for this case [97], [114], [115].

2.2.7 Real-World Missions

As the literature shows, there are many innovative and interesting attitude control methods, techniques, and algorithms. However, only some have been flown. A mathematical algorithm is typically only one part of the larger ADCS, which also includes the determination system, command handling, and different modes of operation [38]. Combining algorithms and novel control ideas into such a system can be difficult and requires an understanding of how the controller behaves, can fail, and how it will interact with the rest of the satellite. There are many papers available detailing control systems for various satellites, and most highlight the customization required for the specific satellite and mission. [48], [65] [49]

One potential reason for these mission-specific applications is that the ADCS approaches can often be simpler. If they are built for a specific satellite, orbit, pointing goal (or set thereof), and mission, they can often be customized and kept quite simple–such as a PD control law. This certainly has advantages. Simpler approaches can use less power and computational effort. When the assumptions used to derive the simplified system are correct, these approaches are robust and verifiable. For example, Lyapunov stability can be shown for control laws but is trickier for trajectory optimization, especially if it is real-time. Another big motivator is risk reduction (often tied to flight heritage). It is seen as less risky, from a mission perspective, to spend time and money making a customized, simpler approach similar to methods that have been used successfully in space before than to try something new that might have better performance or deal better with unplanned issues. One goal of this work is to better describe methods that do not require as much customization but are more than theoretical–methods that are adaptable and generalizable and are prepared for real-world missions.

2.3 Current State of the Art

2.3.1 Current State Summary

The current state of the art in satellite ADCS includes many innovative approaches and interesting algorithms. However, many of these are products of a space industry which has long emphasized certainty over flexibility and robustness over cost. The ongoing changes in space technology and industry are moving towards smaller and cheaper spacecraft with more tolerance for failure and more widely varying goals. Wafer satellites could benefit from small-scale attitude control without moving parts (such as magnetic control), and satellite-based networks may require many small, cheap-to-build satellites–it may be advantageous for these mass-produced satellites to be adaptive to environmental issues, varying goals, and their own physical properties. Desire for cheaper mass-produced satellites may drive the industry towards magnetic control.

These trends, and others that currently exist or have yet to occur, will require new algorithms and control systems. These algorithms will need to improve satellite autonomy, perform faster adjustments and more accurate pointing with weaker and cheaper actuators, and better tolerate disturbances. Ideally, the systems will need to understand their dynamic attitude environment–including errors or biases on their own sensors and actuators as well the disturbances they are encountering, incorporate these effects into their control approach, and potentially plan ahead to maximize compliance with pointing goals under varying conditions.

2.3.2 Technical Gaps

The current state of satellite ADCS reveals four key technical gaps that this work aims to address. First, there is a lack of a **General ADCS Approach** that can adapt across different satellite types, missions, and actuation methods, with most existing systems being highly mission-specific. Second, **Effective Magnetic Control** is still a challenge, as many magnetic control algorithms struggle with robustness and convergence, especially when dealing with

variable pointing goals. Third, **ADCS Trajectory Planning** remains limited and is rarely used, with existing planners focused on specific goals or missions. Finally, there is a need for **Autonomous Rapid Retargeting**, as many systems require manual input to switch between pointing goals, reducing flexibility and responsiveness during missions. Addressing these gaps can significantly improve the autonomy, adaptability, and efficiency of future ADCS architectures.

2.3.2.a General ADCS Approach

As discussed in Section 2.2, there are many ADCS components that exist or have been theorized. However, many of these approaches–for both determination and control–are designed for specific satellites and specific mission goals. Approaches that are valid across spacecraft of different scale, actuation methods, and CONOPs are rare. An example is estimators. There is a gap as there is no easy approach to creating satellite ADS for a wide variety of satellites, including and/or tracking modeled or unmodeled disturbances, control efforts, and errors or biases in both sensors and actuators. However, there is a desire for a more general, non-customized, mission-agnostic approach that can include all of these for satellites with a wide range of physical properties, mission goals, and sensor and actuator quality. Such an approach has not yet been well-described or analyzed, and a straightforward method is needed to create such estimators for generic spacecraft.

2.3.2.b Effective Magnetic Control

Another gap area is effective (works without relying on outside stabilizing factors), fast (converges in an orbit, not hours or days), and accurate (meets pointing goals of several degrees or better) magnetic control that can tolerate different pointing or tracking goals and allows for switching between them. Such algorithms will be critical to improved performance in upcoming small and inexpensive satellites, where magnetorquers will have a great advantage due to their low cost and high reliability. There are many magnetic control algorithms that have been proposed or used, as shown in Section 2.2. However, many of these are tailored to a specific pointing goal or satellite. Others are robust but converge slowly and thus are not useful for quick slews or changing targets. Others still are fast, but less accurate or intolerant to disturbances.

2.3.2.c ADCS Trajectory Planning

A significant gap in the field of attitude control lies in the autonomous planning of time-varying orientation trajectories, rather than the more common slew maneuvers targeting a fixed endpoint. This distinction is critical for modern satellite operations, particularly for missions that require continuous adjustments to meet evolving pointing goals. Existing literature has primarily focused on formulating optimization problems for attitude trajectory planning but lacks detailed insights into practical implementation. Key missing components include the methodologies for solving these optimization problems onboard in real-time, the scheduling and frequency of planner executions, and the handling of convergence failures. Furthermore, most work does not address how planned trajectories integrate into a broader mission architecture, especially for resource-constrained small satellites. Another key part

of this gap is the lack of robust disturbance rejection and closed-loop control in trajectory-following frameworks. Trajectory planners often produce nominal paths but fail to account for real-world factors like persistent disturbances, leading to significant deviations from planned behavior and reduced mission performance. Addressing these gaps requires not only a focus on trajectory generation but also a comprehensive framework that considers onboard computation constraints, adaptive execution, and resilience to real-world operational challenges.

2.3.2.d Autonomous Rapid Retargeting

Most satellite ADCS are based on the idea of two modes: rarely slewing between goals or states, and then, more frequently, tracking/maintaining those goals. There is the opportunity for a more general approach that allows for many different and varying goals, enabling the spacecraft to quick;y re-targeting and changing attitudes on the fly, instead of prioritizing stability over long periods of time (even many orbits). An example of a varying goal trajectory is shown in Figure 2.2.

2.4 Specific Motivation

Many of the ideas in this dissertation originated from the MIT BeaverCube CubeSat (Figure 2.4). This was a magnetically-actuated 3U CubeSat that potentially experienced disturbances (specifically a torque of unknown magnitude from a propulsion system) of a magnitude too large for the magnetorquers (shown in Figure 2.3) to easily counter. Thus, the system was under-actuated, and relatively weak in the directions it did have control. In developing the attitude control system for this satellite, the author became aware of the current state of ADCS and the gaps described in Section 2.3.2. The satellite needed to estimate and understand the propulsion disturbance, potentially alongside other disturbances, as well as flexibly manage estimation despite errors in lower-quality sensors (Section 2.3.2.a). Testing and debugging particularly required the mission ADCS to be flexible as certain parameters and issues were turned on and off. Dealing with the propulsion disturbance and varying pointing goals required a more advanced magnetic attitude control system (Section 2.3.2.b). To address the issue of underactuation, trajectory planning was applied over constant and repeating windows as opposed to simple slews (Section 2.3.2.c). As the satellite needed to switch between various goals, a way to encode and admit varying goals that could change quickly was needed (Section 2.3.2.d). The efforts made for the BeaverCube CubeSat were generalizable in their own right. This is described further in Chapter 3, particularly in Section 3.3.

2.5 Conclusion

This chapter provides a foundation for the dissertation by exploring previous work, the current state, and challenges of satellite Attitude Determination and Control Systems (ADCS), while also articulating the specific motivations for this research. Section 2.2 describes the variety of approaches to ADCS, from early passive methods to contemporary systems. Specific focus was

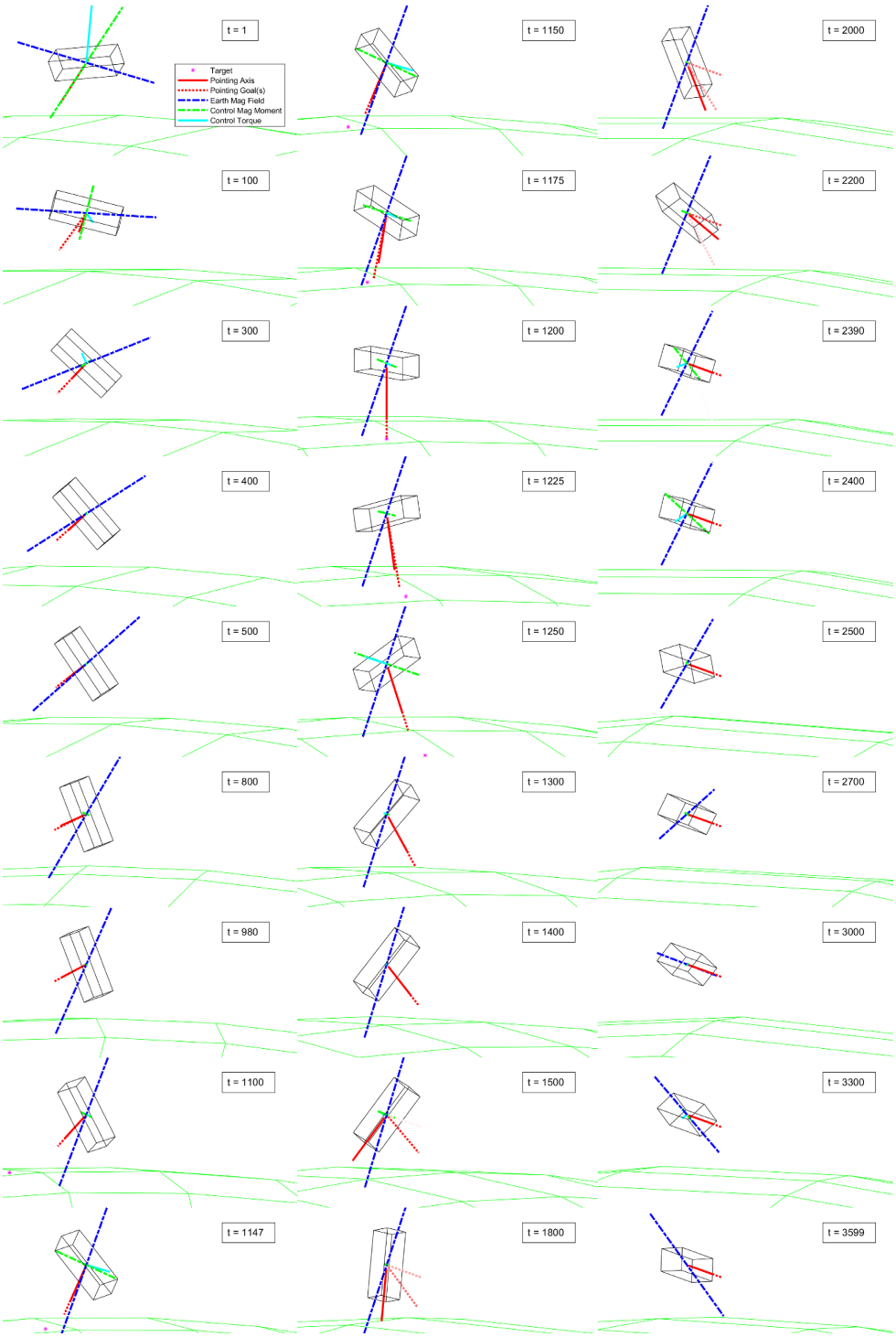


Figure 2.2: Graphic showing various time points of a CubeSat flying a trajectory where it switches between goals. At $t = 1$, the CubeSat is floating freely in an arbitrary orientation. It plans a trajectory for the next hour (3600 seconds) to meet two goals: tracking a target on the ground (small star) from in an 8-minute window centered on 1200 seconds (960s-1440s), and then pointing its propulsion unit anti-ram from 2400 seconds onwards.

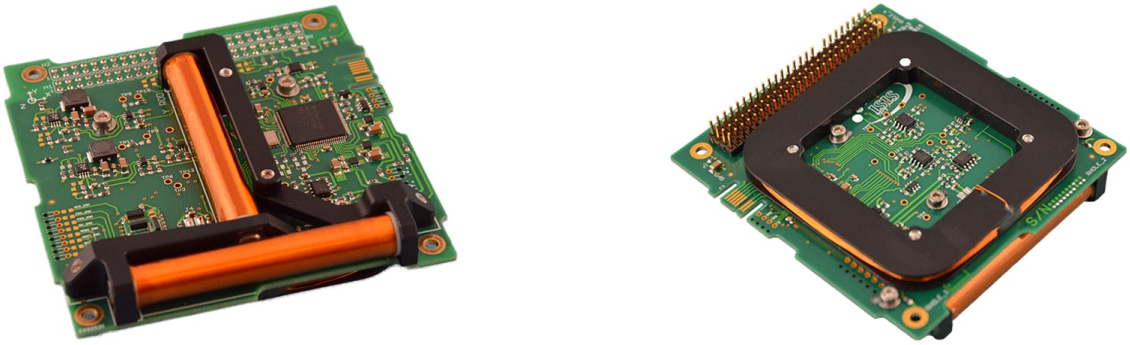


Figure 2.3: Example of magnetorquers, specifically the ISISPACE iMTQ board, which was used in the MIT BeaverCube mission that motivated aspects of this work (see Figure 2.4). The left image is the obverse of the board, showing two torque rods, while the right image is the board's reverse, with an air core torquer. The air core torquer is used to keep the board to a manageable size, as a third torque rod would have to be perpendicular to the others and make the envelope much larger. The rods and air core torquer each produce a magnetic field that is perpendicular to the other two (up to 0.2 Am^2 each), allowing the board to create a magnetic moment in any direction.

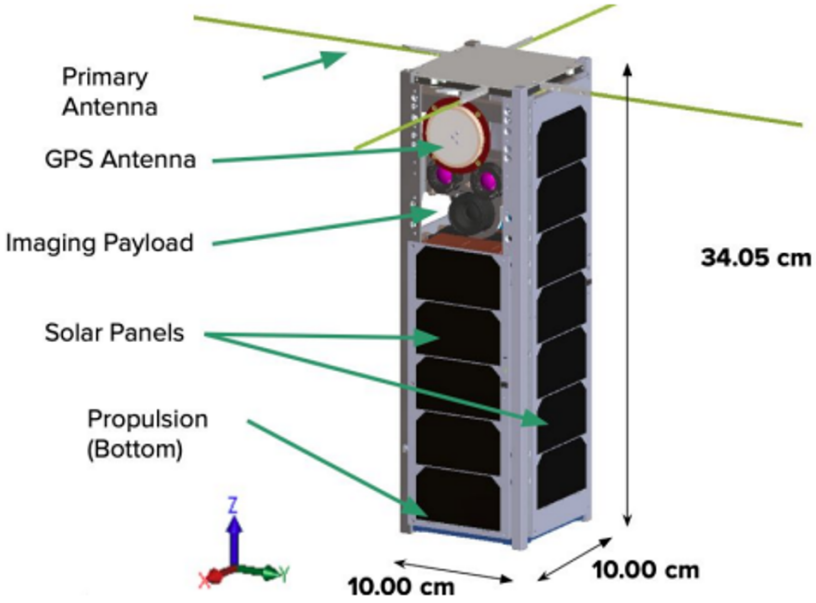


Figure 2.4: The MIT BeaverCube CubeSat, the flight mission that motivated aspects of this work. Image source: BeaverCube CDR CAD [107]

given to sub-topics germane to this dissertation, such as magnetic control, trajectory planning, and reduced attitude control. The discussion also included the technical capabilities and constraints of various modern systems, such as their reliance on predefined mission parameters and limited adaptability to diverse operational conditions. Section 2.3 briefly articulates the unique challenges posed by emerging mission requirements, such as underactuated satellite systems, resource-limited configurations, and the growing demand for autonomy in multi-mission scenarios. Some specific areas of needed improvement are labeled as technical gaps and described in Section 2.3.2, including the struggles of applying existing approaches to magnetic control, the lack of scalable and modular frameworks, and the need for methods using trajectory planning. These gaps are closely tied to the central focus of this dissertation, guiding the development of contributions that address the critical limitations of current systems. Finally, Section 2.4 describes MIT BeaverCube, a 3U CubeSat mission that led to much of the work in this dissertation. It shows how the ideas of this dissertation, and the specific contributions and results, can solve problems in a real-world mission.

By transitioning from a broad review of the field to a detailed exploration of technical gaps and specific motivations, this chapter establishes the context for the areas of work and contributions detailed in Chapter 3. The identified gaps underscore the importance of developing generalized ADCS approaches, effective magnetic control, and trajectory planning techniques, which are addressed in subsequent chapters. This transition sets the stage for the methodological advancements that follow, providing a clear link between the challenges of modern ADCS and the solutions explored in this dissertation.

Chapter 3

Approach and Contributions

3.1 Introduction

This project started as an attempt to develop an ADCS for a small satellite with many constraints (see Section 2.4). In the process, the author became convinced that there could be a more general and autonomous approach. The question of pointing a rigid-body satellite in space is almost a textbook physics and optimization problem. The dynamics are perfectly defined. The goals can be stated explicitly. There is uncertainty, but it can be quantified. There are disturbances, but they are generally understood and can be estimated. Unlike other autonomy problems, there is no chance of a sudden issue occurring that the system is expected to address (compared to, for example, a self-driving car braking for a pedestrian). The problem can be written in full mathematical detail, so it seems like there should be a solution. A solution where the spacecraft's own control system can process the desired goals, incorporate actuation limits and disturbance effects, and guide the system to meet the goals–without needing to pre-plan a slew or reaction wheel desaturation. This idea extends to a satellite rapidly changing between time-varying goals, retargeting all on its own to meet mission objectives as necessary.

This work does not claim to have the full or optimal solution to this problem. However, it presents a framework for solving such a problem for a wide range of satellites, disturbances, and goals. It is readily adaptable to platforms of different capabilities, whether they are high-performance or low-cost; the approach can be applied to either. Such flexibility is possible because it does not rely on specific physical parameters, sensing/actuation capabilities, or desired behavior.

By addressing the technical gaps identified in Section 2.3.2, this dissertation aims to improve autonomous ADCS for satellites, especially those that are small or underactuated, enabling more accurate pointing and more responsive systems. This improvement in capability has several impacts. By improving the expected performance, missions can be accomplished using weaker actuators or underactuated systems, such as magnetic control. Additionally, more responsive systems also allow for satellites that can autonomously switch between different pointing and tracking goals. Faster slews become feasible, with pointing algorithms that force the orientation to converge quickly to their goal. Finally, this approach enables control that is more robust to disturbances, even with weaker actuation. Combined, these

improvements allow for more tasks to be completed in a single mission, or for missions to be accomplished with smaller, simpler, and cheaper spacecraft. Such possibilities are synergistic with current movement in the aerospace field towards smaller and less-expensive nanosatellites, mass-produced satellites, and large constellations.

This chapter serves as the overview for this dissertation. The areas of work and the ideas behind it are covered in Section 3.2, and the contributions in Section 3.3. The remainder of the chapter (Section 3.4) covers the relevant technical concepts that will be used throughout this work.

3.2 Areas of Work

As described in Section 2.4, the original spark of this dissertation work comes from the MIT BeaverCube (Figure 2.4) CubeSat project. This was a magnetically-actuated 3U CubeSat that potentially had disturbances (specifically, a torque of unknown magnitude from a propulsion system) of a magnitude too large for the magnetorquers to easily counter. Thus, the system was both weakly and under-actuated. In developing the attitude control system for this satellite, trajectory planning was determined to have the best chance of meeting time-varying pointing goals, resulting in the work referenced in Section 3.2.4. To further simplify the pointing goals and improve the odds of good performance, single-axis pointing was implemented, as part of an improved approach to ADCS goals and the ADCS overall, as described in Section 3.2.2. Finally, to further improve accuracy under comparatively large disturbances, a plan for modeling these disturbances in the trajectory planner, post-planner control, and other ADCS components was developed (Section 3.2.3). In order to understand the disturbances, it was important to track them, motivating the estimator work described in Section 3.2.1. And while working on this system, it became clear how testing and debugging various components like estimators and control laws was easier when the approaches were generalizable to other satellites and missions. That work, mentioned in Section 3.2.5, makes the results broadly applicable and valuable across many missions. All of these ideas are also applicable to larger questions, and could serve many spacecraft ADCS on their own or in combination. These benefits are conceptually illustrated in Figure 3.1 and discussed further in their respective sections (Sections 3.2.1-3.2.5).

To achieve these goals, several areas required progress to address technical gaps. Specifically, the areas of work included:

- **Improved Attitude State Estimation**: Focuses on enhancing satellite attitude estimation by integrating dynamic models and handling biases, disturbances, and sensor noise effectively.
- **ADCS Operations and Goals**: Explores improving autonomy by allowing satellites to achieve different pointing goals without manual intervention, supporting a wide range of satellite missions.
- **Direct Countering of Disturbances**: Introduces methods for incorporating real-time disturbance estimations into control laws to enhance performance in underactuated systems.

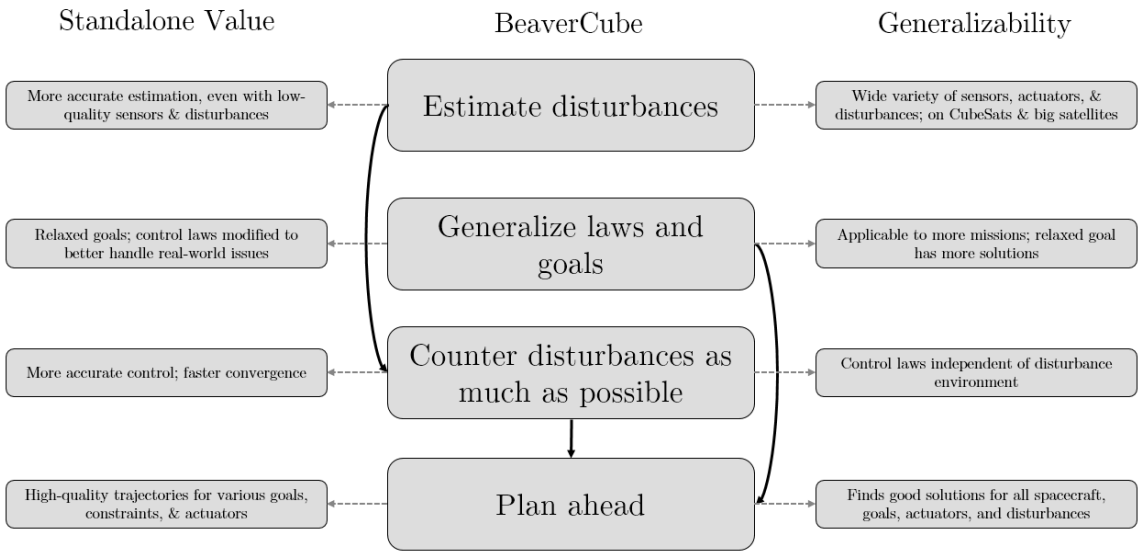


Figure 3.1: A rough overview of how the magnetic control problem broke into components to be solved, and how each of these components had value in their own right, and how could be generalized.

- **Attitude Trajectory Planning**: Utilizes planning methods such as ALTRO to efficiently achieve time-varying and constant pointing goals for satellites, and includes real-world implementation details.
- **Generalized ADCS**: Proposes a comprehensive ADCS framework that can be adapted across various satellite architectures, missions, and disturbance environments.

Combined, these objectives will support various tracking and pointing goals and enable the ability to change between them.

3.2.1 Improved Attitude State Estimation

A key part of any ADCS is the ADS–the system by which the spacecraft estimates and understands its own state, particularly its orientation. For the magnetic control problem, the disturbance also needs to be estimated so it could be countered when possible. With limited and noisy sensors, tracking a small disturbance is a challenge. The solution is to be very deliberate about the sources of uncertainty and how the system evolves, incorporating all information possible. This includes disturbances; separable noise on sensors, actuators, and dynamics; biases on actuators and sensors; and dynamics rules. For debugging and to test the effect of these (when correct and when wrong), a general estimator was built that allows the incorporation of a wide variety of options. This is detailed in Chapter 4 and specifically addresses part of the **General ADCS Approach** technical gap in Section 2.3.2.a. This general and more accurate estimator is applicable to most missions, not just small satellites.

3.2.2 ADCS Operations and Goals

Many approaches to ADCS are over-strict or require more constraints than needed. One clear example is setting the orientation goal to be a specific orientation rather than the reduced attitude problem of simply aligning two vectors (with some counter examples discussed in Section 2.2.5). As work progressed, it became apparent that a general approach could be valuable (e.g., the **General ADCS Approach** technical gap in Section 2.3.2.a). This was especially important when testing various control laws from literature and attempting to adjust the simulation and the control laws used to different spacecraft, disturbance environments, and goals. One relevant need was to describe system goals and limits in a minimal fashion, such as aligning two vectors, angular keep-out zones, velocity limits, etc. and relying on a fully specified orientation only when required. Additionally, it seemed valuable to allow the goals able to vary over time and be part of the satellite's regular operations, as opposed to making slews or reaction wheel desaturation a special separate event. This specification of goals can be incorporated into the spacecraft control approach which can also vary over time (using the B-dot control initially, for example). This is especially important for a system with trajectory planning and notably helpful in the underactuated magnetic control case, as it reduces the goal by one degree of freedom. It is, however, applicable to all missions by making the control opportunities more flexible. The efforts are mostly described in Chapter 5. Specifically, this area of work helps to address the **General ADCS Approach** and **Autonomous Rapid Retargeting** technical gaps in Sections 2.3.2.a and 2.3.2.d, respectively with some effect on the **ADCS Trajectory Planning** gap (Section 2.2.4).

3.2.3 Direct Countering of Disturbances

Many ADCS for satellites treat disturbances as noise, and counter them using control laws that are provably stable. In many cases, this is effective, robust, and simple. However, most disturbances are not random–some are fully describable, others vary with a small randomness over time. However, they can usually be better understood than random noise and thus can be countered with more precision when included directly. In cases where a disturbance is large or the system is underactuated (such as the magnetic control problem), direct incorporation and countering of the disturbance torques may be required to achieve the pointing goals. This is detailed in Chapter 6, specifically addressing the **General ADCS Approach** technical gap (Section 2.3.2.a), while also playing a role in the **Effective Magnetic Control** (Section 2.3.2.b) and **ADCS Trajectory Planning** (Section 2.3.2.c).

3.2.4 Attitude Trajectory Planning

Most satellites rely on relatively simple control laws (such as PID control), but there are cases where this is not enough. When there are large disturbances, actuators may not be strong enough to counter them directly or they might saturate. In an underactuated system, some motions or countering of disturbances are simply impossible. And if more rapid goal-switching is desired, as described in the **Autonomous Retargeting** technical gap in Section 2.3.2.d, more nuanced control may be required. To find a solution, the author investigated trajectory planning–creating control not just based on the current state,

but determining a "path" through the space of orientations that meets specified goals (see Section 3.2.2) as much as possible, including large disturbances where necessary, and respecting issues like actuator bounds and desaturation. This is especially useful to the magnetic control problem as the magnetic field varies throughout the orbit. By utilizing planning, the system is underactuated only in the moment, not over the whole trajectory. This work directly addresses the **ADCS Trajectory Planning** technical gap in Section 2.3.2.c, but also has impact on other techincal gaps, particularly **Effective Magnetic Control** (Section 2.3.2.b), **General ADCS Approach** (Section 2.3.2.a), **Autonomous Rapid Retargeting** (Section 2.3.2.d). This work on trajectory planning can be found primarily in Chapter 7.

3.2.5 Generalized ADCS

There are a wide variety of approaches to satellite ADCS. Many are not easily generalized, as they are based on specific actuators, spacecraft, or mission details. Throughout this work, the author endeavored to make each component developed as general as possible–with the idea that they can be applied to satellites with wildly different physical properties, pointing goals, sets or arrangements of actuators and sensors, and disturbances. By relying on general dynamic models and approaches that do not presuppose a scale, disturbance environment, sensor set, mission goal, or actuation method, the work of this dissertation should be applicable across many spacecraft and missions. As it is a general principle, the relevant work cannot be found in a specific place in this dissertation, but applies to work in every chapter.[1] This is a direct attempt to help address the **General ADCS Approach** technical gap in Section 2.3.2.a. By making each of the steps more general, it makes debugging and testing easier. For example, it is a useful test to replace the magnetic control with thrusters to see if the issue comes from it being under-actuated. The primary value in keeping things general, however, is that the results and methods produced are then applicable to a wide variety of missions, spacecraft, and problems. This is described in Figure 3.1.

3.3 Contributions

The efforts in the areas of work described in Section 3.2 produced five contributions. Some of these contributions come from a specific area of work, while others are the result of a combination of two or more. The contributions are described in this section, with a rough overview plus connections to the areas of work and technical gaps, shown in Figure 3.2.

3.3.1 Generalized Disturbance- and Dynamics-Aware Estimator

The creation, both mathematically and as software, of a generalized attitude estimation system that incorporates disturbances, dynamics, and control effort for a wide variety of spacecraft, sensors, actuators, and disturbances is a contribution. This is a direct effort towards addressing the need for a general approach to estimation, described in Section 2.3.2.a.

[1] For example: the generalizable nature of the estimator in Chapter 4 and the planner in Chapter 7, the various control laws–magnetic and thruster-based–in Chapter 6, and the approach to goals and operations in Chapter 5.

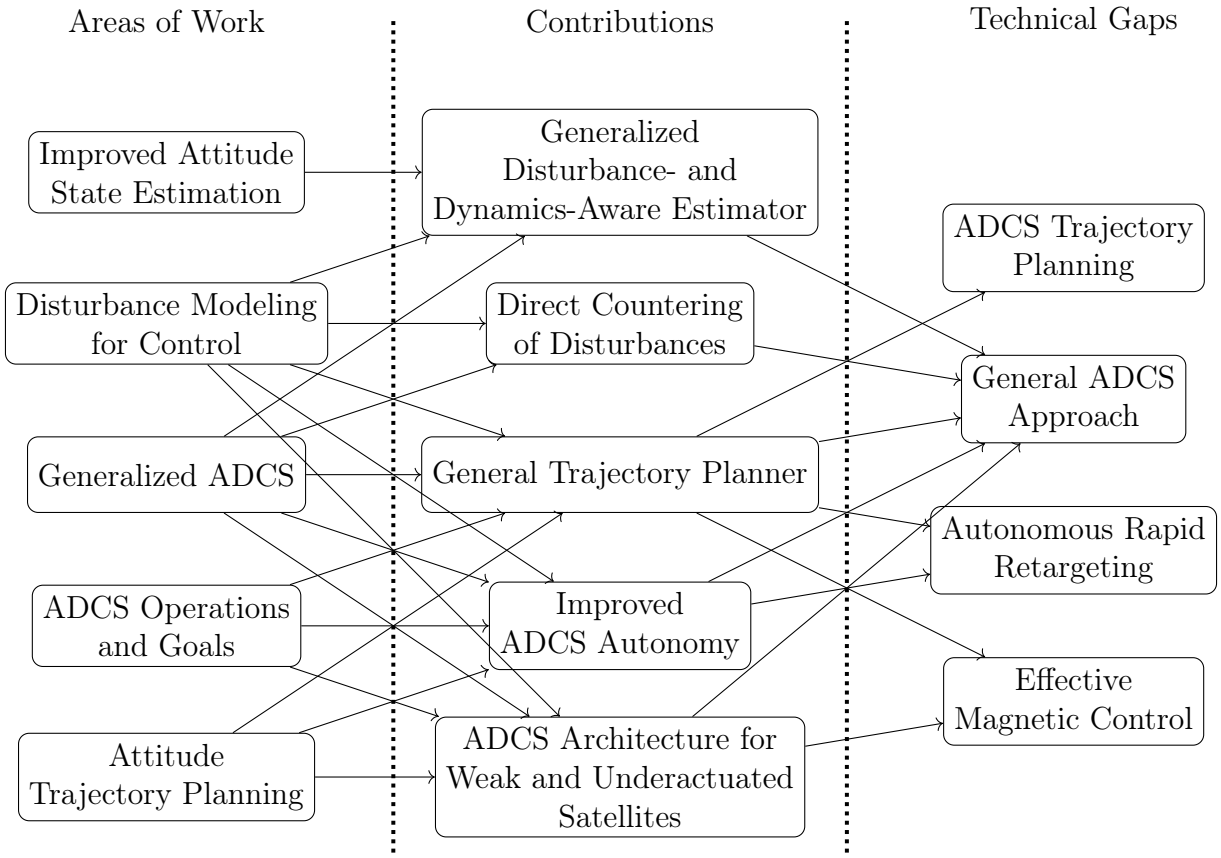


Figure 3.2: A rough overview of how the various contributions are related to the areas of work and technical gaps.

The relevant work can largely be found in Chapter 4 and comes from the **Generalized ADCS** and **Improved Attitude State Estimation** areas of work described in Sections 3.2.5 and 3.2.1. As described in Section 2.2, there has been a great deal of effort on satellite ADS; however, there have been few approaches that generalize across spacecraft of widely differing sizes, complexities, disturbance conditions, and costs, particularly in a way that is easy for designers to implement and use.

This contribution includes:

- the derivations, for modeling satellite state, including angular velocity, orientation represented as a quaternion, stored angular momentum, time-varying disturbing torques and magnetic moments, and sensor and actuator biases;
- methods to encompass various sources of uncertainty in satellite modeling, including sensor and actuator noise, integration error, and general process noise; and
- methods to easily create estimators (including different types of nonlinear Kalman filters) for systems with a variety of physical properties, actuators, sensors, and disturbances, based on the properties and noise behavior of different satellites.

This estimator has proven particularly useful for cases when strong disturbances threaten to overwhelm a satellite that has weak actuation or is underactuated. It is notably useful in the control of a magnetically-controlled spacecraft with body-based disturbances, as those disturbances can be estimated and incorporated into control laws, as described in the **Disturbance Modeling for Control Inclusion** contribution in Section 3.3.2. However, it is useful in many other areas as well, allowing the quick creation of an ADS for simple or small satellites (as well as large ones!), permitting ADS that can incorporate lower-quality sensors and actuators that have large noise or biases, and improving ADS accuracy and precision by taking advantage of all information such as the dynamics information, specific sources of uncertainty, and control effort.

3.3.2 Disturbance Modeling for Control Inclusion

The third contribution is a set of methods for including a variety of expected disturbances into satellite attitude control. This is a specific effort toward the **General ADCS Approach** technical gap in Section 2.3.2.a, with the idea of making disturbance inclusion easier to do in the general case, and not just for specific missions or control laws. The relevant work is largely in Chapter 6, and it comes from the **Direct Countering of Disturbances** and **Generalized ADCS** areas of work described in Sections 3.2.3 and 3.2.5.

This method of disturbance inclusion and the resulting algorithms improve pointing performance by reducing the magnitude and source of "noise" in the real-world system dynamics. This should allow for more accurate pointing, as well as the possibility for reduced actuation effort and the inclusion of trajectories that would not be feasible for otherwise underactuated systems.

This contribution includes:

- an analysis of how different disturbances can be effectively and efficiently modeled for use in real-time algorithms;

- a method of real-time estimation of time-varying disturbances (magnetic dipole and bias) and methods to include these in planning or correct them during control;
- methods to include analytical disturbances (propulsion and gravity gradient) in planning, estimation, and control;
- consideration of a potential "general" torque term calculated by an estimator (see the **Generalized Disturbance- and Dynamics-Aware Estimator** contribution in Section 3.3.1) for cases where disturbances cannot be precisely modeled; and
- inclusion of both modeled and all-in-one disturbances in a trajectory planning algorithm and other control approaches to improve pointing.

This is directly applicable to the original problem of a magnetic satellite with a significant body disturbance, but it is applicable at large as well. All satellites face disturbances, and the potential to counter them more directly (instead of treating them as noise) can improve pointing performance and lead to more efficient use of control efforts.

3.3.3 Trajectory Planning

The first contribution is the creation of an attitude trajectory planner and accompanying closed-loop controller for spacecraft with changing pointing goals. The planner includes various disturbances and that can be used for a variety of actuator arrangements, directly addressing the **ADCS Trajectory Planning** technical gap described in Section 2.3.2.c. The relevant work falls under the **Attitude Trajectory Planning** area of work described in Section 3.2.4 (with connections to **ADCS Operations and Goals** in Section 3.2.2 and **Generalized ADCS** in Section 3.2.5). Further information is detailed in Chapter 7. As described in Chapter 7, this approach allows an underactuated, magnetically-controlled CubeSat to meet tighter pointing goals, making this contribution applicable to the **Effective Magnetic Control** technical gap (Section 2.3.2.b) as well. By permitting frequently changing goals, this method also works to improve the **Autonomous Rapid Retargeting** technical gap (Section 2.3.2.d). Such a controller does not appear to have been used before on satellites, especially CubeSats or those relying on magnetic control. Implementation details were also not yet found in the literature.

This contribution includes:

- the trajectory planner details, including problem formulation and algorithmic solving approach, as well as the seed trajectory and constraint enforcement;
- the variability across satellites with different disturbances and actuator arrangements;
- the primary use of being always-on, continuously planning trajectories to meet different (potentially varying) goals, instead of being used for specific slews or other operations;
- the closed-loop control method; and
- the overarching control system that includes command interpretation, scheduling of planning, management of calculation time and exceptions, and that deals with convergence failures.

In addition to its novel nature, this approach allows trajectories to be flown with magnetic actuation that would otherwise be difficult or infeasible. It has value for fully-actuated spacecraft as well–it allows for reduced slew times, reduced actuation effort, and improved pointing accuracy. This advantage is especially true when combined with other work (in this dissertation as well as existing approaches), such as disturbance inclusion and single-axis pointing goals. Enabling such trajectories and accuracy also creates the opportunity for more novel or uncommon control architectures, such as a single reaction wheel with three magnetorquers or a system with a full set of reaction wheels that constantly desaturates, and for more advanced mission plans and CONOPS.

3.3.4 Improved Autonomy for ADCS

The **Trajectory Planning** and **Disturbance Modeling for Control Inclusion** contributions mentioned in Sections 3.3.3-3.3.2 have benefits on their own, but they can also be used together to create greater synergistic contributions. One such contribution is an ADCS approach that is more autonomous, general, and flexible across different satellite architectures, scales, and missions. This approach can take into account the spacecraft disturbance environment, actuation limitations, and specific goals to manage the satellite's attitude in a reasonable (though perhaps not provably optimal) way. This arises from effort towards the **Effective Magnetic Control** technical gap in Section 2.3.2.b and is certainly applicable to that case, as further described in the following contribution (Section 3.3.5). However, it is also useful for many other satellites and is a step towards a **Generalized ADCS Approach** (Section 2.3.2.a), with an additional focus on a variety of changing goals, making it applicable to the Rapid Retargeting Technical Gap as well (Section 2.3.2.d). The work for this contribution is spread across several chapters, as it takes parts from Chapter 4, 6, 5, and 7. These chapters each show part of the method, and some show results of the combination. The most detail can be found in Chapter 5. The work largely falls in the **ADCS Operations and Goals** and **Generalized ADCS** areas of work described in Sections 3.2.2 and 3.2.5.

This contribution includes:

- an ADCS framework that combines a generalized estimator and planner (or other control approaches) in a manner that includes disturbances and considers the ADCS goals;
- methods to create this framework for a variety of satellite architectures with various sensors and actuators of different limits and quality, and a variety of disturbances;
- incorporation of a variety of types of pointing goal, at varying times, with methods to include them in various control laws;
- incorporation of a variety of control laws (with the potential for many more) and specific schedules for using them; and
- software to combine these methods and operate onboard satellites.

This approach applies broadly across different satellite missions. For a simple, fully-actuated nadir-pointing satellite, it reduces to a standard ADCS configuration with enhanced robustness

against actuator failures, sensor issues, and unexpected disturbances, potentially improving pointing accuracy and reducing actuation demands. However, its true value lies in more complex missions. By supporting dynamic goals and rapid changes, this framework enables satellites to accommodate evolving mission requirements, maximizing operations within each orbit and throughout the mission duration. It allows for higher activity rates per unit time while increasing resilience to component limitations, thereby reducing the need for high-cost sensors and actuators without compromising control precision.

3.3.5 ADCS Architecture for Weak or Underactuated Satellites

The final contribution is also a "combined" contribution, as described in Section 3.3.4 and is exactly what is described in Section 2.4–an ADCS approach for satellites that are weak or underactuated and in a difficult disturbance environment. This contribution is a direct result of efforts on the **Effective Magnetic Control** Technical Gap in Section 2.3.2.b. In many cases, this contribution uses the methods laid out in this dissertation (across Chapters 4, 6, 5, and 7) to fully address that technical gap. However, due to efforts on making each component generalizable, the final product is also useful for many other satellites and is a step towards a **Generalized ADCS** (Section 2.3.2.a), especially for small satellites. While results can be found across Chapter 4-7, the clearest results, combining all of the parts to show the improved performance of limited systems, can be found in 7. The work largely falls in the areas of work described in Sections 3.2.2 and 3.2.5.

This contribution includes:

- the generalized code for different ADCS components that can be applied to various small or underactuated satellites; and
- results showing the possibility of novel actuator combinations, such as constant desaturation or the combination of one reaction wheel and three magnetorquers.

By increasing the accuracy and making weak or underactuated control (whether magnetic or not) feasible in general for small satellites (not tied to specific geometry or mission goals), this contribution can enable such methods to be used on future missions, allowing for satellites that are cheaper, more reliable, and capable of pointing with higher accuracy.

3.4 Technical Background

This section provides some context on various technical concepts that are used throughout the work. Additionally, in some of these areas, notations, conventions, and approaches differ from work to work. This section is to ensure the reader has the necessary context.

3.4.1 Quaternions

There are many different notations for quaternions, with varying levels of clarity in usage. Even the "standard" notation is confusing. A slightly different notation is described here that differentiates quaternions from vectors but allows for conversion (in terms of notation) between

quaternions and 4-vectors. A background on the quaternions and how they correspond to rotation is also given.

There are a number of ways to represent the orientation of a spacecraft, such as rotation matrices. This work will primarily use unit quaternions, due to their compact size, their lack of singularities and gimbal lock, and convenient mathematical properties.

Unfortunately, there are many different notations and conventions for quaternions. To minimize confusion, we will first briefly define quaternions and our notation and express the quaternion operations we will use (including how those operations are represented). For example, this work will use the diacritics to represent vectors and quaternions and whether they have unit length. Specifically, $\breve{q}$ indicates a general quaternion, $\bar{q}$ a unit quaternion, $\mathbf{q}$ a general vector, and $\hat{\mathbf{q}}$ a unit vector. Similarly, a set of elements in $[\star]$ represents a vector while a set of elements in $\{\star\}$ represents a quaternion.

3.4.1.a Quaternion Definition and Component Operations

The quaternion is a four-dimensional algebraic object that can be imagined as an extension of complex numbers. There are a variety of equivalent ways to represent and describe a quaternion $\breve{q}$, with different advantages. One that is commonly seen is $\breve{q} = q_0 + q_1 i + q_2 j + q_3 k$, where q_0, q_1, q_2, and q_3 are the four real scalar components. However, due to mathematical convenience, this work will focus on the description of a quaternion as having a scalar and a vector component, and written as

$$\breve{q} = q_0 + \mathbf{q}_v \tag{3.1}$$

where q_0 is the real scalar component and $\mathbf{q}_v$ is the real, three-dimensional vector component.

The symbols used, q_0 and $\mathbf{q}_v$, are also used as operators to "pull out" the scalar and vector components of a given quaternion, such as:

$$\breve{q} = a + \mathbf{u} \implies q_0 = a \tag{3.2}$$

and

$$\breve{q} = a + \mathbf{u} \implies \mathbf{q}_v = \mathbf{u}. \tag{3.3}$$

These operations allow us to pull out the scalar and vector components of a quaternion, but we will also need a way to indicate the creation of a quaternion using both a scalar and a vector. To do so, we will contain the scalar and vector in $\{\star\}$, such as

$$\begin{Bmatrix} q_0 \\ \mathbf{v} \end{Bmatrix} = q_0 + \mathbf{v}. \tag{3.4}$$

We will also use vector transpose notation on Equation 3.4, such as $\begin{Bmatrix} q_0 & \mathbf{v}^T \end{Bmatrix}^T = q_0 + \mathbf{v}$.

3.4.1.b Quaternion Operations

With these component operations defined, we can define the quaternion operations in the same manner as [116]. For simplicity and notational clarity, we repeat them here. The first is simple: quaternion addition works element-wise, so that [116]

$$\breve{p} + \breve{q} = \begin{Bmatrix} p_0 + q_0 \\ \mathbf{p}_v + \mathbf{q}_v \end{Bmatrix}. \tag{3.5}$$

Quaternion multiplication is more complex, defined as [116]

$$\breve{p}\breve{q} = \begin{Bmatrix} p_0 q_0 - \mathbf{p}_v \cdot \mathbf{q}_v \\ p_0 \mathbf{q}_v + q_0 \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{Bmatrix}, \tag{3.6}$$

where $\cdot$ represents the standard vector dot product and $\times$ is the standard three-element vector cross product.

While quaternions are not vectors, the vector dot product is a useful operation to borrow. We will define it as

$$\breve{p} \cdot \breve{q} = p_0 q_0 + \mathbf{p}_v \cdot \mathbf{q}_v \tag{3.7}$$

The magnitude, or length, of a quaternion is identical to that of a vector. Specifically, [116]

$$\left\| \breve{q} \right\|^2 = q_0^2 + \mathbf{q}_v \cdot \mathbf{q}_v = \breve{q} \cdot \breve{q} \quad \text{and} \quad \left\| \breve{q} \right\| = \sqrt{q_0^2 + \mathbf{q}_v \cdot \mathbf{q}_v}. \tag{3.8}$$

Finally among quaternion operations, there is the quaternion inverse, defined as [116]

$$\breve{q}^{-1} = \frac{1}{\left\| \breve{q} \right\|^2} \begin{Bmatrix} q_0 \\ -\mathbf{q}_v. \end{Bmatrix} \tag{3.9}$$

3.4.1.c Quaternions and Rotation

While we have defined the operations on the algebraic quaternion, their application to rotation and orientation remains undefined. Unit quaternions, where $\|\bar{q}\| = 1$, can be used to represent three-dimensional rotation and thus the relative orientation between frames. Note that for these unit quaternions, the inversion operation in Equation 3.9 simplifies to $\bar{q}^{-1} = \begin{Bmatrix} q_0 & -\mathbf{q}_v^T \end{Bmatrix}^T$.

Specifically, there is a reasonably simple formula to transform a vector by the rotation that the quaternion represents. For simplicity, we will use the $\circ$ operator to represent the rotation between frames, defined as [116]:

$$\mathbf{v}_{\text{rotated frame}} = \bar{q} \circ \mathbf{v} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\bar{q}^{-1} \begin{Bmatrix} 0 \\ \mathbf{v} \end{Bmatrix} \bar{q} \right], \tag{3.10}$$

where $[\breve{p}]$ indicates treating $\breve{p}$ as a four-element column vector, equal to $\begin{Bmatrix} p_0 & \mathbf{p}_v^T \end{Bmatrix}^T$. This definition of rotation is the equivalent to a rotation matrix A such that

$$\mathbf{v}_{\text{rotated frame}} = \bar{q} \circ \mathbf{v} = A(\bar{q})\,\mathbf{v}, \quad A(\bar{q}) = \left(q_0^2 - \left\|\mathbf{q}_v\right\|^2\right) I_{3\text{x}3} + 2\mathbf{q}_v \mathbf{q}_v^T - 2q_0 \mathbf{q}_v^{\times}, \tag{3.11}$$

where $I_{3\text{x}3}$ is the 3x3 identity matrix and $\mathbf{a}^{\times}$ is the skew-symmetric matrix of vector $\mathbf{a}$. Because $A(\bar{q})$ is a rotation matrix, $A^T(\bar{q}) = A^{-1}(\bar{q}) = A(\bar{q}^{-1})$.

It is worth remarking that unit quaternions double-cover the space of rotations. Specifically, $\bar{q}$ and $-\bar{q}$ represent equivalent rotations,[2] so that

$$\bar{q} \circ \mathbf{v} = (-\bar{q}) \circ \mathbf{v} \tag{3.12}$$

[2]This may seem very unintuitive, but it is a mathematical quirk. One way to think about it, using the axis-angle conversion in Equation 3.13, is that flipping the sign on both the scalar and the vector components is equivalent to flipping both the axis and angle of rotation, which would result in the same rotation.

where $-\bar{q} = \begin{Bmatrix} -q_0 & -\mathbf{q}_v^T \end{Bmatrix}^T$.

To convert a rotation quaternion to and from a more intuitive rotation representation, we have a formula generating a unit quaternion from an axis-angle rotation [117]. The unit quaternion representing a rotation of angle ϕ around an axis $\hat{\mathbf{k}}$ can be expressed as

$$\begin{Bmatrix} \cos\frac{\phi}{2} \\ \hat{\mathbf{k}}\sin\frac{\phi}{2} \end{Bmatrix}. \tag{3.13}$$

There are many ways to describe the distance between a pair of quaternions [118]. For this work, the difference between two unit quaternions, $\bar{p}$ and $\bar{q}$ will be expressed by the angle

$$\Delta(\bar{p}, \bar{q}) = \arccos\left(2(\bar{p}\cdot\bar{q})^2 - 1\right). \tag{3.14}$$

This formula can be derived from the metrics explored by [118].[3] The derivation of this metric is that the angle between $\bar{p}$ and $\bar{q}$ is the angle of rotation of $\bar{p}\bar{q}^{-1}$, the rotation quaternion that would make them align. Treating this quaternion as a single-axis rotation, we find from Equation 3.13 that the scalar component of $\bar{p}\bar{q}^{-1}$ is equal to $\cos\frac{\delta}{2}$, if δ is the angle between the orientations. Combining Equations 3.6 and 3.9, we see that we can rewrite this as

$$\cos\frac{\delta}{2} = \left(\bar{p}\bar{q}^{-1}\right)_0 = p_0 q_0 - \mathbf{p}_v \cdot (-\mathbf{q}_v) \tag{3.15}$$

which can be further simplified with Equation 3.7 to

$$\cos\frac{\delta}{2} = p_0 q_0 + \mathbf{p}_v \cdot \mathbf{q}_v = \bar{p}\cdot\bar{q}. \tag{3.16}$$

From this, we use the cosine double-angle identity (confining the difference angles to the interval $[-\pi, \pi]$) to create our distance metric in Equation 3.14.

Lastly, a note on convention. Unit quaternions represent a rotation. This includes a rotation between frames, so it can be used to describe a spacecraft's orientation relative to the global frame. When used in this manner, we use the formalism that the orientation quaternion is a rotation that would rotate the spacecraft body frame to match the world frame.[4] In other words, a vector expressed in the body frame, when rotated by the inverse of the orientation quaternion, would equal the expression of the same vector expressed in the world frame:

$$\mathbf{v}_{\text{world}} = \bar{q}^{-1} \circ \mathbf{v}_{\text{body}} \tag{3.17}$$

and

$$\mathbf{v}_{\text{body}} = \bar{q} \circ \mathbf{v}_{\text{world}}. \tag{3.18}$$

Other sources may use the opposite convention, though these formulae and approach will work for them as well.

[3] However, the author first found (and used) this metric for a previous quaternion-based project in a Mathematics Stack Exchange post by Jim Belk: https://math.stackexchange.com/questions/90081/quaternion-distance

[4] Orientation quaternions do not inherently describe a rotational motion but are instead a way of describing the relative orientation between frames. When we talk about "rotating vectors" between the body and the world, we are actually converting between vector expressions in different reference frames. See [24], Chapter 3 for more information. This work uses the Hamilton convention.

3.4.2 Rotational Dynamics

The crux of this work involves pointing a satellite. This section will briefly review the governing equations of rotational dynamics. A satellite with an inertia tensor J and body-frame angular velocity $\boldsymbol{\omega}$ can be modeled using Euler's equation subject to torque $\boldsymbol{\tau}$, as

$$\dot{\omega} = J^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times J\boldsymbol{\omega}), \tag{3.19}$$

where $\dot{v}$ indicates the time derivative (most of these values are time-varying–except J–but the (t) component of the functions has been left off for visual clarity, as will be done throughout this work). With reaction wheels, which are actuators with stored angular momentum described further in Section 3.4.4.b, the dynamics equation is

$$\dot{\boldsymbol{\omega}} = J_{\text{noRW}}^{-1}\left(-\boldsymbol{\omega} \times \left(J_{\text{noRW}}\omega + \sum_{i=1}^{N_{\text{RW}}} \hat{\alpha}_{\text{RW},i} h_i\right) + \boldsymbol{\tau}_{\text{control}} + \boldsymbol{\tau}_{\text{disturbance}}\right), \tag{3.20}$$

where $J_{\text{noRW}} = J - \sum_{i=1}^{N_{\text{RW}}} j_{\text{RW},i}\hat{\alpha}_{\text{RW},i}\hat{\alpha}_{\text{RW},i}^T$ represents the spacecraft inertia tensor ignoring all momentum storage. The ith reaction wheel i has axis $\hat{\alpha}_{\text{RW},i}$, stored momentum h_i, and rotational inertia $j_{\text{RW},i}$ about that axis ($\hat{\alpha}_{\text{RW},i}$). The stored momentum for the ith RW evolves according to

$$\dot{h}_i = -\tau_{\text{RW,i}} - j_{\text{RW,i}}\left(\hat{\alpha}_{\text{RW,i}}^T\dot{\boldsymbol{\omega}}\right). \tag{3.21}$$

3.4.3 Error, Bias, and Noise

A key aspect of working on ADCS, or any control system, is dealing with imperfection. Sensors have error sources: scale and misalignment, random noise, bias, and more. Actuators are imperfect as well, subject to fluctuations, noise, control sub-routines (such as motor speed controllers), and more. Other effects, such as some relevant dynamic or environmental parameters (including disturbances), evolve randomly over time. Throughout this work, a unified model is used to capture these sources of uncertainty.

For actuators and sensors, the error is split into two components: bias and noise. They can be configured to have either, both, or neither. If x is the "true" sensor value (or commanded actuation effort), the actual sensor reading (or actuation effort applied) is $x + \beta + \nu$, where β is the bias and ν is the noise.

In this work, noise ν involves high-frequency values that change between each sampling or command and is (usually) modeled as a zero-mean white-noise process. Mathematically, this is a random (Normal/Gaussian) distribution: $\nu = N(0, \sigma_{\text{noise}})$, where $N(\mu, \sigma)$ denotes the normal distribution with mean μ and standard deviation σ. The value of σ_{noise}, which scales the size of the noise, varies depending on the sensor or actuator in question (and its real-world value could have outside dependencies, such as temperature).

In comparison, bias β is a low-frequency value that changes slowly (or is even a constant). In theory, it can be tracked, a key part of the results in Chapter 4. In this dissertation, it is modeled as a discrete random walk approximating a Wiener process. This modeling method is commonly used for gyroscopes [18], [25], [27], [119]–[122], where the bias β is defined as the integral of a random noise process:

$$\dot{\boldsymbol{\beta}} = \boldsymbol{\nu}_{\text{drift}}(t) \quad \text{s.t.} \quad E[\boldsymbol{\nu}_{\text{drift}}(t_1)\boldsymbol{\nu}_{\text{drift}}^T(t_2)] = Q_{\text{drift}}\delta(t_1 - t_2), \tag{3.22}$$

where $\boldsymbol{\nu}_{\text{drift}}$ is a zero-mean white noise with an autocorrelation of Q_{drift}. The bias β is the integral of this random process noise, making it a Wiener process [123]. Separating noise sources into a high-frequency random white noise components and a low-frequency Wiener process (integral of a random white noise component) is a widely-used model of gyroscope noise [18], even though gyroscopes have other sources, colors, and types of noise such as quantization noise and flicker noise or bias instability [121], [122]. There is less consistency across the literature in how to model noise for other sensors (including sometimes including no noise such as in sun sensors), actuators, or the evolution of disturbances. Due to its generality, this approach has been used in this work for all other sensors and actuators when bias and noise are included.

A method similar to this bias model is used in this work for other randomly-varying parameters. It is most relevant for certain disturbances which might vary over time, such as the spacecraft residual dipole or a propulsion torque. The same model of a discretized Wiener process (equivalent to the integral of zero-mean white noise or a random walk) is used.

3.4.4 Real-world Systems

There are many kinds of sensors that can be used in spacecraft ADS. The sensors relevant to this dissertation are magnetometers (MTMs), gyroscopes (gyros), sun sensors, and sun sensor pairs. All sensors are subject to various noise and error possibilities. These can include scaling or misalignment error, nonlinearities, interference or bias. For this work, we will focus on the idea of bias–a slow-varying or constant offset between the "true" and reported value. Specifically, we will model sensors as

$$s = z + \beta + \nu, \tag{3.23}$$

where z is the "true" value, β is the bias, and ν is the noise.

The bias can be set directly, or evolve over time. When evolving over time, it is modeled as a discrete random walk approximating a Wiener process. Each sensor can be different, but the method used for sensors throughout this work is the same and described here. The mathematical background can be found in Section 3.4.3. For the ith sensor (this may include repeats of a sensor, such as each of three orthogonally-aligned gyroscopes), a bias rate represented as $\eta_{b,\text{sens},i}$ is used. Each time the bias evolves, the time it was updated is saved as t_{prev}. When updated again at time t, the new bias is

$$\beta'_{\text{sens},i} = N\left(\beta_{\text{sens},i}, \eta_{b,\text{sens},i}(t - t_{\text{prev}})\right), \tag{3.24}$$

where $N(\mu, \sigma)$ is the normal distribution.

While bias changes with a low frequency, there is also high-frequency noise represented as ν. This noise is treated as a random distribution. Specifically, for the ith sensor

$$\nu_{\text{sens},i} = N(0, \eta_{\text{sens},i}), \tag{3.25}$$

where $\eta_{\text{sens},i}$ is the noise standard deviation.

3.4.4.a Sensors

Sun sensors There are many types of sun sensors, but they all work by indicating a direction towards the sun. In this work, coarse sun sensors are primarily used, that return a value of fractional brightness they detect based on the direction they are pointing. Specifically, they are described with efficiency ζ and axis $\hat{\alpha}$, they read as

$$s = \zeta \max\left(0, \left(\hat{\alpha}^T A(\bar{q}) \frac{\mathbf{S}_{\text{ECI}}}{\|\mathbf{S}_{\text{ECI}}\|}\right)\right) + \beta + \nu, \tag{3.26}$$

where $\mathbf{S}_{\text{ECI}}$ is the vector pointing towards the sun in the ECI frame.

Coarse sun sensors are often used in pairs on opposite sides of a spacecraft. This is described here with efficiency $\zeta_{\pm 1}$ and anti-parallel axes $\pm\hat{\alpha}$ as

$$s = \zeta_m \left(\hat{\alpha}^T A(\bar{q}) \frac{\mathbf{S}_{\text{ECI}}}{\|\mathbf{S}_{\text{ECI}}\|}\right) + \beta + \nu, \tag{3.27}$$

where $m = \text{sgn}\left(\hat{\alpha}^T A(\bar{q}) \frac{\mathbf{S}_{\text{ECI}}}{\|\mathbf{S}_{\text{ECI}}\|}\right)$

Gyroscopes Gyroscopes work by measuring the angular velocity at which they are spinning. This returns the angular velocity of the spacecraft around each gyroscope axis. Gyroscopes are particularly weak to a drifting bias. They read, with axis $\hat{\alpha}$

$$s_i = \left(\hat{\alpha}^T \omega\right) + \beta + \nu. \tag{3.28}$$

Magnetometers Magnetometers (MTMs) work by detecting the magnetic field of the Earth or other celestial body and using that, combined with a model of the magnetic field at the satellite's position, to determine orientation. Specifically, with axis $\hat{\alpha}$, they read

$$s = \left(\hat{\alpha}^T A(\bar{q}) \mathbf{B}_{\text{ECI}}\right) + \beta + \nu, \tag{3.29}$$

where $\mathbf{B}_{\text{ECI}}$ is the local magnetic field vector in the global frame.

This approach could be extended to star trackers or cameras and Earth horizon sensors.

3.4.4.b Actuators

There are many kinds of actuators that can be used in spacecraft ACS. The actuators relevant to this dissertation are magnetorquers (MTQs), reaction wheels (RWs), and thrusters (though not all interactions involving thrusters, such as fuel depletion's effect on mass properties, are modeled). All actuators are subject to various noise and error possibilities. These can include scaling or misalignment error, nonlinearities, interference, or bias. For this work, we will focus on the idea of bias–a slow-varying or constant offset between the "true" and reported value. Specifically, we will model actuators as

$$u_{\text{all}} = u + \beta + \nu, \tag{3.30}$$

where u is the "commanded" value, β is the bias, and ν is the noise.

The bias can be set directly or evolve over time. When evolving over time, it is modeled as a discrete random walk approximating a Wiener process. Specifically, a bias rate is used, represented as η_b. The mathematical background can be found in Section 3.4.3. For each (this may include repeats of an actuator, such as each of three orthogonally-aligned magnetometers), a bias rate represented as η_b is used. Each time the bias evolves, the time it was updated is saved as t_{prev}. When updated again at time t, the new bias is

$$\beta' = N\left(\beta, \eta_b(t - t_{\text{prev}})\right), \tag{3.31}$$

where $N(\mu, \sigma)$ is the normal distribution.

While bias changes with a low frequency, there is also high-frequency noise represented as ν. This noise is treated as a random distribution. Specifically,

$$\nu = N(0, \eta), \tag{3.32}$$

where η is the standard deviation.

Magnetorquers Magnetorquers (MTQs) are essentially electromagnets that work by creating a magnetic moment in the spacecraft. Like a compass, the overall spacecraft magnetic moment then tries to align with the local magnetic field. Specifically, a magnetorquer on axis $\hat{\alpha}$:

$$\boldsymbol{\tau} = \left(A(\bar{q})\mathbf{B}_{\text{ECI}}\right) \times \hat{\alpha}\left(u + \beta + \nu\right) \tag{3.33}$$

where $\mathbf{B}_{\text{ECI}}$ is the local magnetic field vector in the global frame.

Thrusters Thrusters work by propelling mass out of the spacecraft. If thrusters on opposing ends of the spacecraft fire in opposite directions, a torque results. This thesis only considers thrusters like these, used for attitude control, and ignores the translational effects (as well as fuel usage). Specifically, a pair of thrusters about axis $\hat{\alpha}$ create torque as:

$$\boldsymbol{\tau} = \hat{\alpha}\left(u_i + \beta + \nu\right). \tag{3.34}$$

Reaction Wheels Reaction wheels are speed-controlled spinning flywheels with a body fixed axis. They are used to provide torque via conservation of angular momentum. To torque in one direction, the satellite spins the relevant wheel (or a linear combination of wheels of different axes) in the opposite direction, like a person on a skateboard throwing a bowling ball. Specifically, with axis α, the torque is

$$\boldsymbol{\tau} = \hat{\alpha}\left(u + \beta + \nu\right). \tag{3.35}$$

Unlike other actuators described here, the reaction wheels each store angular momentum (represented as h) about their axis α, which is relevant to the system dynamics. Specifically,

$$\tau_{\text{s}} = -\tau = -\left(u + \beta + \nu\right) = \dot{h}_i + j_{\text{RW}}\left(\hat{\alpha}^T\dot{\boldsymbol{\omega}}\right), \tag{3.36}$$

where $\dot{h}$ is defined by equation 3.21, $\dot{\omega}$ is the angular acceleration of the overall satellite (defined in Equation 3.20), and j_{RW} is the reaction wheel's own rotational inertia about its axis $\hat{\alpha}$.

3.4.4.c Disturbances

This work considers four disturbance torques in space, among the most common and significant in low Earth orbit (LEO). The models are based on Fitzgerald [108]. Drag torque is caused by interaction with the extremely thin but nonzero atmosphere. Solar radiation pressure is caused by the momentum of light from the sun being deposited on the spacecraft. The spacecraft can sometimes become magnetized, leaving it with a residual dipole that can create torque in the same way as magnetorquers or a compass. Finally, the gravity gradient disturbance is caused by the mass distribution of the spacecraft and the fall-off in gravity with distance.

Drag torque can be visualized as the satellite acting similar to a weather vane as it moves through the extremely thin atmosphere of low Earth (or other planetary body) orbit. For every surface i (of total number N) on the spacecraft, with drag coefficient $C_{D,i}$, area A_i, angle α_i between the ram vector and the surface normal, and centroid location $\mathbf{c}_i$ (relative to the center of mass), the drag torque is described as

$$\boldsymbol{\tau}_{\text{drag}} = -\frac{1}{2}\rho\,|\mathbf{v}|\sum_{i=1}^{N} A_i C_{D,i}\max\left(0,\cos\alpha_i\right)\mathbf{c}_i \times \mathbf{v}, \tag{3.37}$$

where ρ is the gas density and $\mathbf{v}$ is the velocity of the satellite, expressed in the body frame. (Note that this ignores "shadowing" of surfaces with respect to the "airstream" by other components and is thus only accurate for convex geometries, but this will suffice for many satellites, especially pico- and nanosatellites.)

Solar radiation pressure (SRP), which is caused by the momentum pressure of light and other electromagnetic radiation from the sun, also interacts with the satellite to create torque in a manner similar to a weather vane, is modeled similarly. There are three forces on each surface of the satellite, based on how it interacts with light. There are three coefficients, representing what light fraction undergoes specular reflection ($\eta_{s,i}$), diffuse reflection ($\eta_{d,i}$), or absorption ($\eta_{a,i}$). These sum to unity, $\eta_{s,i} + \eta_{d,i} + \eta_{a,i} = 1$. We also use the variables W for the solar radiance, c for the speed of light, and γ_i as the angle between the surface normal $\hat{\mathbf{n}}_i$ and the vector towards the sun $\hat{\mathbf{s}}$. These create forces on each face as:

$$\mathbf{F}_{s,i} = -2\frac{W}{c}A_i\eta_{s,i}\cos\gamma_i\hat{\mathbf{n}}_i\max\left(\cos\gamma_i,0\right), \tag{3.38}$$

$$\mathbf{F}_{d,i} = \frac{W}{c}A_i\eta_{d,i}\left(-\frac{2}{3}\hat{\mathbf{n}}_i - \hat{\mathbf{s}}\right)\max\left(\cos\gamma_i,0\right), \tag{3.39}$$

and

$$\mathbf{F}_{a,i} = -\frac{W}{c}A_i\eta_{a,i}\hat{\mathbf{s}}\max\left(\cos\gamma_i,0\right). \tag{3.40}$$

The torque on the satellite is therefore

$$\boldsymbol{\tau}_{\text{srp}} = -T_{\text{eclipse}}\sum_{i=1}^{N}\mathbf{c}_i \times \left(\mathbf{F}_{s,i} + \mathbf{F}_{d,i} + \mathbf{F}_{a,i}\right) \tag{3.41}$$

where T_{eclipse} is 1 if not in eclipse and 0 if it is.

There is also the **residual magnetic dipole** caused by the spacecraft becoming magnetized and torquing against an ambient magnetic field, such as the Earth's planetary magnetic field. If the local B-field is expressed as $\mathbf{B}$ in the global or ECI frame and the residual dipole is $\mathbf{m}_{\text{residual}}$, the body frame torque is

$$\boldsymbol{\tau}_{\text{dipole}} = (A(q)\mathbf{B}) \times \mathbf{m}_{\text{residual}}. \tag{3.42}$$

Gravity gradient torque is caused by the small difference in the strength of gravitational force as distance from a central body increases. It can be described as

$$\boldsymbol{\tau}_{\text{gg}} = \frac{3\mu_E}{|\mathbf{r}|^5}\mathbf{r} \times J\mathbf{r} \tag{3.43}$$

where $\mathbf{r}$ is the satellite's position relative to the object it is orbiting (expressed in the spacecraft body frame), J is the satellite inertia tensor relative to the center of mass, and μ_E is the gravitational parameter of Earth (or another central body if applicable).

Arbitrary torques can be specified as a **general torque**, which is simply directly specified in the body frame (in other words, for a given value of the disturbance, $\boldsymbol{\tau}_{\text{gen}} = \boldsymbol{\tau}_{\text{gen}}$). This can be allowed to randomly vary over time as described in [124] and in Section 3.4.3. It can be used to model **propulsion torque** and can also be used in the spacecraft ADCS to track unknown external torques without modeling each of them, as further described in Section 6.3. This can be applied to many control laws, but the cases used in this dissertation are described in Sections 6.2 and 5.4.

3.4.5 Kalman Filter Overview

The Kalman Filter (KF) and its variants are widely used algorithms for estimating the state of a dynamic system in the presence of noise. These filters are based on the recursive prediction-update paradigm, where the system state and its uncertainty are propagated over time using a combination of a dynamics model and noisy measurements. This section provides a brief mathematical overview of the Kalman Filter (KF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Square-Root Unscented Kalman Filter (SRUKF).

3.4.5.a Kalman Filter (KF)

The Kalman Filter (KF) is a recursive algorithm designed to estimate the state of a dynamic system in the presence of noise. It assumes that the system dynamics and measurement models are linear, and that all noise sources are Gaussian and additive. Under these assumptions, the KF provides an optimal estimate of the state by minimizing the mean squared error.

The system is described by the following discrete-time state-space equations:

State Transition:

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \quad w_k \sim \mathcal{N}(0, Q_k), \tag{3.44}$$

where $x_k \in \mathbb{R}^n$ is the state vector at time step k, $A_k \in \mathbb{R}^{n\times n}$ is the state transition matrix that governs how the state evolves over time, $u_k \in \mathbb{R}^m$ is the control input, and $B_k \in \mathbb{R}^{n\times m}$ is the control input matrix. The process noise w_k represents random disturbances in the system, modeled as zero-mean Gaussian noise with covariance matrix Q_k.

Measurement Model:

$$z_k = H_k x_k + v_k, \quad v_k \sim \mathcal{N}(0, R_k), \tag{3.45}$$

where $z_k \in \mathbb{R}^p$ is the measurement vector, $H_k \in \mathbb{R}^{p \times n}$ is the measurement matrix that relates the state x_k to the measurements, and v_k is the measurement noise, modeled as zero-mean Gaussian noise with covariance R_k.

The KF operates in two key steps, Prediction and Update, as detailed below:

Prediction Step: During the prediction step, the state estimate and its associated uncertainty are propagated forward in time using the system dynamics:

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} + B_k u_k, \tag{3.46}$$

$$P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k. \tag{3.47}$$

Here:

- $\hat{x}_{k+1|k}$ is the predicted state estimate at time $k+1$ based on information up to time k and
- $P_{k+1|k}$ is the predicted state covariance matrix, which quantifies the uncertainty in the predicted state.

The state transition matrix A_k and process noise covariance Q_k play critical roles in shaping the predicted state and its uncertainty.

Update Step: In the update step, the Kalman Filter incorporates new measurements to refine the state estimate:

$$K_k = P_{k+1|k} H_k^T \left(H_k P_{k+1|k} H_k^T + R_k \right)^{-1}, \tag{3.48}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k \left(z_k - H_k \hat{x}_{k+1|k} \right), \tag{3.49}$$

$$P_{k+1|k+1} = \left(I - K_k H_k \right) P_{k+1|k}. \tag{3.50}$$

Here:

- K_k is the Kalman gain, which determines the weight given to the new measurement relative to the predicted state,
- $\hat{x}_{k+1|k+1}$ is the updated state estimate after incorporating the measurement z_k (and thus is the estimated state at time $k+1$ using all information up to time $k+1$), and
- $P_{k+1|k+1}$ is the updated covariance matrix, which reflects the reduced uncertainty in the state estimate.

The Kalman gain K_k ensures optimal weighting of the predicted state and the new measurement, balancing the relative uncertainties represented by $P_{k+1|k}$ and R_k.

Summary: The Kalman Filter iteratively alternates between prediction and update steps, enabling real-time state estimation. It is widely used in applications ranging from navigation to control systems due to its computational efficiency and ability to handle noisy environments. However, its performance is contingent on accurate knowledge of the system dynamics (A_k, B_k) and noise covariances (Q_k, R_k).

3.4.5.b Extended Kalman Filter (EKF)

The Extended Kalman Filter extends the KF to systems with nonlinear dynamics and measurements. The system is represented as:

$$x_{k+1} = f(x_k, u_k) + w_k, \quad w_k \sim \mathcal{N}(0, Q_k), \tag{3.51}$$

$$z_k = h(x_k) + v_k, \quad v_k \sim \mathcal{N}(0, R_k), \tag{3.52}$$

where $f(x, u)$ and $h(x)$ are the (potentially) nonlinear state transition and measurement models, respectively. The EKF approximates these nonlinear models by linearizing them using first-order Taylor expansions around the current state estimate, by taking the Jacobians of the f and h.

Prediction Step The state and covariance are propagated forward as:

$$x_{k+1|k} = f(x_{k|k}, u_k), \tag{3.53}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k, \tag{3.54}$$

where F_k is the Jacobian of f with respect to x:

$$F_k = \frac{\partial f}{\partial x}\Big|_{x_{k|k}, u_k}. \tag{3.55}$$

Update Step The Kalman gain is computed using the Jacobian H_k, which linearizes h with respect to x:

$$H_k = \frac{\partial h}{\partial x}\Big|_{x_{k+1|k}}. \tag{3.56}$$

The state and covariance are updated as:

$$K_k = P_{k+1|k} H_k^T \left(H_k P_{k+1|k} H_k^T + R_k\right)^{-1}, \tag{3.57}$$

$$x_{k+1|k+1} = x_{k+1|k} + K_k(z_k - h(x_{k+1|k})), \tag{3.58}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_k H_k P_{k+1|k}. \tag{3.59}$$

The EKF is a straightforward extension of a nonlinear system, but the Jacobian calculation can be expensive. Additionally, it sacrifices accuracy that can be achieved from full calculation of f and h. In doing so, it uses an approximation of the dynamics on top of the estimation, which is already an approximation. Another approach is the Unscented Kalman Filter.

3.4.5.c Unscented Kalman Filter (UKF)

The Unscented Kalman Filter (UKF) addresses the limitations of the Extended Kalman Filter (EKF) by avoiding explicit linearization. Instead of relying on first-order Taylor series approximations, the UKF uses a deterministic sampling method called the *unscented transform* to propagate a set of sigma points through the nonlinear system. This approach ensures better performance in handling highly nonlinear systems by capturing higher-order effects.

Sigma Point Generation The UKF approximates the probability distribution of the state by propagating a set of deterministically chosen *sigma points*, which represent the mean and spread of the state distribution. These sigma points χ are generated around the mean x_k using the square root of the covariance matrix P_k and scaling parameters α, β, and κ:

$$\chi_0 = x_k, \quad \chi_i = x_k \pm \sqrt{(L+\lambda)P_{k_i}}, \quad i = 1, \ldots, L. \tag{3.60}$$

Additionally, we define weights for the mean $W_i^{(m)}$ and covariance $W_i^{(c)}$ as:

$$W_0^{(m)} = \frac{\lambda}{L+\lambda}, \quad W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta), \tag{3.61}$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(L+\lambda)}, \quad i = 1, \ldots, 2L. \tag{3.62}$$

For Equations 3.60 and 3.61:

- $\lambda = \alpha^2(L + \kappa) - L$ is a scaling parameter,
- L is the dimension of the state vector x,
- α controls the spread of the sigma points around the mean,
- κ is a secondary scaling parameter, often set to 0 or $3 - L$, and
- β incorporates prior knowledge of the distribution ($\beta = 2$ is optimal for Gaussian distributions).

Prediction Step Each sigma point is propagated independently through the nonlinear state transition function f:

$$\chi_{i,k+1|k} = f(\chi_{i,k}, u_k). \tag{3.63}$$

The predicted state mean $\hat{x}_{k+1|k}$ and covariance $P_{k+1|k}$ are computed as weighted sums of the propagated sigma points:

$$\hat{x}_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} \chi_{i,k+1|k}, \tag{3.64}$$

$$P_{k+1|k} = \sum_{i=0}^{2L} W_i^{(c)} \left(\chi_{i,k+1|k} - \hat{x}_{k+1|k}\right) \left(\chi_{i,k+1|k} - \hat{x}_{k+1|k}\right)^T + Q_k, \tag{3.65}$$

where:

- $W_i^{(m)}$ and $W_i^{(c)}$ are the weights for the mean and covariance, respectively, and
- Q_k is the process noise covariance matrix.

Update Step The sigma points are propagated through the nonlinear measurement function h:

$$\zeta_{i,k} = h(\chi_{i,k+1|k}), \tag{3.66}$$

to compute the predicted measurement mean $\hat{z}_{k+1|k}$ and covariance $P_{zz,k+1|k}$:

$$\hat{z}_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} \zeta_{i,k}, \tag{3.67}$$

$$P_{zz,k+1|k} = \sum_{i=0}^{2L} W_i^{(c)} \left(\zeta_{i,k} - \hat{z}_{k+1|k}\right) \left(\zeta_{i,k} - \hat{z}_{k+1|k}\right)^T + R_k. \tag{3.68}$$

The cross-covariance between the state and measurements is computed as:

$$P_{xz,k+1|k} = \sum_{i=0}^{2L} W_i^{(c)} \left(\chi_{i,k+1|k} - \hat{x}_{k+1|k}\right) \left(\zeta_{i,k} - \hat{z}_{k+1|k}\right)^T. \tag{3.69}$$

The Kalman gain K_k is calculated as:

$$K_k = P_{xz,k+1|k} P_{zz,k+1|k}^{-1}. \tag{3.70}$$

The state and covariance are updated using the new measurement z_k:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k \left(z_k - \hat{z}_{k+1|k}\right), \tag{3.71}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_k P_{zz,k+1|k} K_k^T. \tag{3.72}$$

Summary By leveraging the unscented transform and deterministic sampling, the UKF achieves greater accuracy than the EKF for nonlinear systems while maintaining a computational complexity that is manageable for many real-time applications.

3.4.5.d Square-Root Unscented Kalman Filter (SRUKF)

The Square-Root Unscented Kalman Filter (SRUKF) is an extension of the UKF that improves numerical stability and computational efficiency by propagating the square root of the covariance matrix S_k, such that $P_k = S_k S_k^T$. By avoiding direct computation of the covariance matrix P_k, the SRUKF ensures positive semi-definiteness and mitigates numerical issues that may arise in scenarios involving small or large state covariances [20].

Prediction Step The prediction step in the SRUKF propagates the square root of the covariance matrix. The sigma points χ_i are generated (Equation 3.60) and propagated (Equation 3.63) as in the UKF, but the covariance update is handled using a numerically stable QR decomposition =:

$$S_{k+1|k} = \text{qr}\left(\left[\sqrt{W_i^{(c)}}\left(\chi_{i,k+1|k} - \hat{x}_{k+1|k}\right), \text{integration noise}\right]\right), \tag{3.73}$$

where:

- $S_{k+1|k}$ is the square root of the predicted covariance matrix,
- $\sqrt{W_i^{(c)}}$ are the square-root weights for the covariance (as in Equation 3.61), and
- The QR decomposition (and taking only the "R" matrix from the decomposition, ignoring the orthonormal "Q", represented here by the function $\text{qr}(M)$) ensures that $S_{k+1|k}S_{k+1|k}^T = P_{k+1|k}$, where $P_{k+1|k}$ is the predicted covariance matrix.

Note that "integration noise" in Equation 3.73 is not the same as R_k as used in the KF or EKF. It is an expected amount of noise and thus corresponds to $\sqrt{R_k}$.

Update Step In the update step, the square root of the measurement covariance $S_{zz,k+1|k}$ and the cross-covariance $P_{xz,k+1|k}$ are computed using the sigma points propagated through the measurement model h. The measurement update proceeds as follows:

1. Compute the predicted measurement mean and deviations:

$$\hat{z}_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} \zeta_{i,k}. \tag{3.74}$$

2. Update the square root of the measurement covariance:

$$S_{zz,k+1|k} = \text{qr}\left(\left[\sqrt{W_i^{(c)}}\left(\zeta_{i,k} - \hat{z}_{k+1|k}\right), \text{measurement noise}\right]\right). \tag{3.75}$$

3. Compute the cross-covariance:

$$P_{xz,k+1|k} = \sum_{i=0}^{2L} W_i^{(c)}\left(\chi_{i,k+1|k} - \hat{x}_{k+1|k}\right)\left(\zeta_{i,k} - \hat{z}_{k+1|k}\right)^T. \tag{3.76}$$

4. Calculate the Kalman gain:

$$K_k = P_{xz,k+1|k} S_{zz,k+1|k}^{-1}. \tag{3.77}$$

5. Update the state and square root of the covariance matrix using repeated rank-one Cholesky updates (or downdates):

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k\left(z_k - \hat{z}_{k+1|k}\right), \tag{3.78}$$

$$S_{k+1|k+1} = \text{CholeskyUpdate}(S_{k+1|k}, K_k S_{zz,k+1|k}). \tag{3.79}$$

Advantages of the SRUKF The SRUKF offers several key advantages:

- **Numerical Stability:** Propagating the square root of the covariance matrix avoids direct computation and factorization of P_k, reducing numerical errors.
- **Positive Semi-Definiteness:** Ensures that the covariance matrix remains valid by construction.
- **Computational Efficiency:** By leveraging the QR decomposition and Cholesky updates, the SRUKF reduces computational overhead compared to recomputing the full covariance matrix.

Summary The SRUKF extends the advantages of the UKF by improving robustness and efficiency, making it particularly suited for systems with high-dimensional states or when numerical stability is a critical concern.

3.5 Conclusion

This chapter outlines the methodological approach and key contributions of this dissertation, building on the technical gaps identified in Section 2.3.2 and defining the areas of work that guide the research. Section 3.2 introduced five interconnected focus areas: dynamics-aware estimation, adaptable laws and goals, disturbance-aware control, trajectory planning, and generalized ADCS. These areas reflect the dissertation's commitment to developing scalable, autonomous ADCS frameworks that can generalize across a wide range of satellite architectures and mission scenarios.

Section 3.3 describes the five major (albeit interconnected) contributions of this dissertation, which resulted from pursuing the aforementioned areas of work and which help to close the identified technical gaps. Three of these are direct contributions laid out in this dissertation. First, the **Generalized Disturbance- and Dynamics-Aware Estimator** (largely discussed in Chapter 4) provides a framework for precise state estimation in systems with significant noise and disturbances, enabling satellites to maintain accuracy in diverse operating conditions. Second, **Disturbance Modeling for Control Inclusion** control law framework (primarily detailed in Chapter 6) enables flexible and efficient control that adapts to environmental challenges. Third, the **Trajectory Planning** framework (Chapter 7) ensures that satellites can autonomously achieve time-varying goals under complex mission requirements. There are also two contributions which are not a specific output, but instead are advancements made possible through a combination of the work in this dissertation. One is **Improved Autonomy for ADCS**, aimed at reducing reliance on ground intervention, enabling satellites to operate independently in dynamic and unpredictable environments, by adjusting to constraints, biases, and disturbances. The strongest results for this use trajectory planning to autonomously account for disturbances and goals, but some of the work on adjustable pointing goals and modified control laws also allow for some real-time adaptation and autonomy. Lastly, there is the **ADCS Architecture for Weak or Under-actuated Satellites**, which leverages the other contributions and areas of work to create ADCS approaches that accommodate systems with limited actuation or sensing capabilities,

such as CubeSats relying on magnetic-only control. Together, these contributions address the limitations of traditional ADCS frameworks by integrating adaptability, scalability, and autonomy into a cohesive system.

The technical background section, Section 3.4, provides the foundational concepts that underpin the dissertation's contributions to ADCS. It begins by reviewing quaternions as a representation for spacecraft orientation, describing their various operations and relationships to rotation. and then discusses the principles of satellite attitude dynamics and control, highlighting the challenges posed by underactuation, dynamic disturbances, and limited sensor capabilities. This is followed by an overview of rotational dynamics, detailing the governing equations of motion and their application to modeling satellite behavior under various conditions. The section also examines the sensors commonly used for attitude determination, such as magnetometers, sun sensors, and gyroscopes, highlighting their roles in providing critical state information and defining mathematically the information they return, including the impact of bias and noise. The discussion extends to actuators, including reaction wheels, magnetorquers, and thrusters, exploring their capabilities and the mathematical relationship between their commands and system dynamics. Lastly, a set of disturbances are discussed (drag, solar radiation pressure, inherent dipole, gravity gradient, propulsion, and a generalized torque), and the method to calculate their torque and effects is described. By synthesizing these elements, this section establishes the technical framework upon which the dissertation's contributions are built, ensuring a solid foundation for the advanced methodologies developed in subsequent chapters.

By connecting the areas of work to specific contributions and detailing their implementation, this chapter provides a roadmap for the ideas of the dissertation. It transitions from identifying challenges and gaps to presenting actionable solutions, beginning with the modeling and estimation framework in Chapter 4. Each subsequent chapter builds upon the foundational contributions laid out here, demonstrating how they collectively advance the state of the art in ADCS. This transition marks a critical shift from defining the problem to demonstrating the potential of the proposed framework, moving closer to achieving the dissertation's vision of a fully autonomous, mission-agnostic ADCS.

Chapter 4

Modeling and Estimation of Spacecraft Attitude

4.1 Introduction

Attitude estimation (also called attitude determination) has been a cornerstone challenge in spacecraft development since the dawn of space exploration. Attitude estimation is used to provide information about the orientation of the spacecraft. This information can be used, for example, to understand precisely which direction a satellite is pointing when it captures an image and thus exactly what the image is. The outputs of attitude estimation are also used by its sibling, attitude control, where the information about the spacecraft's rotation and orientation is used to feed control algorithms that generate actuator commands in order to achieve the desired spacecraft state. To serve these duties, attitude estimation relies on information from sensors. The sensing hardware can include gyroscopes, magnetometers, sun and Earth sensors, star trackers and cameras, and radio beacons [5]. The information from these sensors can be turned into an estimated attitude through a number of algorithms, including the TRIAD vector estimation algorithm, the extended Kalman filter (EKF), and the unscented Kalman filter (UKF) [6]. Many more complex approaches to estimation also exist, including robust filtering for detecting measurement faults [29]–[31], lower-computational-cost approaches [17], active control to improve parameter estimation [32], full GPS inclusion [33], and sparse-matrix approaches [34]. Various existing approaches to estimation are discussed in Section 2.2. These methods, while effective in specific scenarios, often fail to meet the needs of small satellites, which are constrained by low-cost sensors, underactuated control systems, and high-disturbance environments. These limitations are particularly relevant to the **General ADCS Approach** technical gap (Section 2.3.2.a) and to the original mission underpinning this work (Section 2.4).

ADCS (attitude determination and control systems) use different levels of information about the system dynamics and its environment. It is entirely possible, for example, to estimate attitude only with a three-axis magnetometer (MTM) and high-precision gyroscopes, while completely ignoring the system dynamics [18]. In that approach, it is possible not to track angular velocity as part of the estimated state, instead using the readings from the gyroscopes as raw input (while compensating for gyroscope bias). This works fine when noise

is small but can cause problems when higher-noise gyroscopes are used, such as in CubeSats [35]. One option is to ignore high-noise gyroscopes and focus on direct attitude sensing using star cameras, tracking angular velocity as an additional state and incorporating system dynamics [35]. Both approaches leave information on the table. Even if the gyroscopes are noisy, they still provide some level of information that can be (carefully) incorporated. The control commands (and expected motion) also provide additional information when incorporated. For example, if a torque is commanded to actuators, the expected change in angular velocity is known from the commands and the mass properties of the satellite. When the expected change in angular velocity is compared to the measured change from the gyroscopes, more information about the gyroscope biases is understood. Similarly, the angular velocity change that results from magnetic torque provides information about the satellite's orientation relative to the magnetic field. Using the noise and bias of the control torques allows for more specific modeling of the covariance and noise in the system, in addition to accounting for it.

A different approach would be adaptive control or system identification (SysID). Overall, the goal of understanding parameters and better estimating attitude is to improve attitude control. The methods commonly called adaptive control directly adjust control errors from varying or uncertain parameters [36], [37]. SysID typically seeks to determine or improve a system's dynamic model by processing extensive datasets, often in batch or recursive formats that prioritize model accuracy over immediacy. Adaptive control, conversely, dynamically adjusts control laws to maintain system performance, integrating parameter updates into the control logic. In contrast, the approach presented here assumes a known, albeit imperfect, dynamic model and emphasizes leveraging real-time data–including noisy sensor outputs–for state and parameter estimation. This approach, coming from the **Improved Attitude State Estimation** area of work (Section 3.2.1) and serving as the centerpiece of the **Generalized Disturbance- and Dynamics-Aware Estimator** contribution (Section 3.3.1), focuses on real-time estimation under uncertainty rather than on refining system models or adapting control strategies.

This is similar to the Attitude Parameter-Calibration Kalman Filter (APKF) [35], though a notable distinction is the incorporation of noisy gyroscope data. While the APKF demonstrates strong performance with high-accuracy sensors, its reliance on star trackers limits its applicability for lower-cost sensor configurations. While the APKF and other approaches completely exclude high-noise gyroscope data, the method described here demonstrates that even low-quality measurements can contribute valuable information when their uncertainties are modeled. This inclusion enhances the overall robustness of the estimation process.

One set of parameters to include in the dynamics relates to disturbances. Modeling known disturbances, analytically or by estimation, allows for more precise estimation than a general "process error" term. We have more information about the system than simple filters include; for example, the torque from drag and residual dipoles can be modeled nearly exactly, up to the accuracy of their parameters [85]. This is especially important when a disturbance heavily impacts the dynamics, such as a constant body-frame disturbance like propulsion on an underactuated satellite (see Section 2.4), and is thus a key part of the **ADCS Architecture for Weak or Underactuated Satellites** contribution (Section 3.3.5). When disturbances cannot be fully modeled, or when the parameters are uncertain, an "all-in-one" disturbance torque can be estimated. The all-in-one disturbance torque effectively serves as the residual

on state estimation, accounting for uncaptured disturbances/dynamics errors. This can then be used in controllers – for example, as a feedforward adjustment in PID instead of the integral term. This ties to an autonomous system (see the **Improved Autonomy for ADCS** contribution in Section 3.3.4) and is similar to the torque inclusion in [35].

In addition, the use of UKF (unscented Kalman Filter) and SRUKF (square-root unscented Kalman Filter) formulations in this work provides a critical advantage in handling nonlinearities. By employing sigma-point transformations, the UKF ensures more accurate state propagation in the presence of nonlinearity, while the SRUKF's square-root formulation addresses numerical stability issues common in systems with widely varying state magnitudes. These features make this framework particularly well-suited for missions with complex dynamics and high uncertainty.

Finally, the software implementation reflects the framework's modular philosophy, tied directly to the **Generalized ADCS** area of work (Section 3.2.5). Designed to be nearly plug-and-play, the code supports a wide range of satellite configurations with minimal modification. This adaptability positions the framework as a versatile tool for nanosatellite missions, complementing existing methods that prioritize specialized sensor setups.

To summarize, in order to better estimate the system state and understand the disturbances so that they can be more directly countered, this work details the creation and testing of an estimator framework for ADCS that fully models system dynamics (see Section 4.2), control effort and bias, disturbances, and sensor bias. The estimation framework is created in an object-oriented manner, described in Section 4.3, to allow for easy modifications and to simplify applying tests to different satellites, noise environments, and more. Each estimator, which can include an Extended Kalman Filter or more advanced approaches, processes the sensor and control information, relying on an internal "digital twin" of the satellite to model the dynamics, sensors, actuators, and disturbances. Results for various satellite test cases are shown in Section 4.4. A summary of those test cases can be found in Table 4.1.

Table 4.1: Summary of Test Cases in Chapter 4

Test Case	**Setup Details**[a]	**Section**	**Purpose**	**Focus**	**Results & Notes**
Case A	TRMM: Magnetometers, Sun Sensors	4.4.2	Validate dynamics-aware filter under high initial errors	Compare to USQUE	Faster convergence (sub-degree accuracy in 1 hour). Superior steady-state accuracy (0.01°).

Continued on next page

Table 4.1 Continued

Test Case	Setup Details[a]	Section	Purpose	Focus	Results & Notes
Case B	TRMM: Magnetometers, Sun Sensors	4.4.3	Test performance under dynamics-aware filter with smaller initial errors	Compare to USQUE	Rapid convergence to 0.01°. Improved robustness over baseline methods.
Case C	3U CubeSat: CubeSat-class sensors, Poor-quality Gyroscopes, Magnetometers, Sun Sensors	4.4.4	Test performance under noisy sensors and large initial errors	Sensor performance under noise; filter convergence rate	Achieved 0.1° angular error and 10^{-3}°/s velocity error. Rapid sub-degree convergence in 10 minutes.
Case D	3U CubeSat: CubeSat-class sensors, Poor-quality Gyroscopes, Magnetometers, Sun Sensors	4.4.5	Demonstrate performance with favorable starting conditions	Filter stability and accuracy	Achieved 0.05° - 0.1° angular error. Stability under disturbances.
Case E	3U CubeSat: CubeSat-class sensors, Poor-quality Gyroscopes, Magnetometers, Sun Sensors	4.4.6	Evaluate importance of modeling actuator bias and disturbances	Effect of including disturbances and biases	Ignoring actuator bias caused divergence to 180°. Ignoring disturbances caused divergence to 1°-5°.
Case F	3U CubeSat: CubeSat-class sensors, Poor-quality Gyroscopes, Magnetometers, Sun Sensors, Propulsion Disturbance	4.4.7	Assess disturbance estimation performance	Propulsion disturbance tracking	Pointing accuracy of 0.1°. Estimated torque to within 5×10^{-7} Nm. Ties directly to original mission case. (Section 2.4)

Continued on next page

Table 4.1 Continued

Test Case	Setup Details[a]	Section	Purpose	Focus	Results & Notes
Case G	6U CubeSat: Mid-quality Gyroscopes, Magnetometers, Sun Sensors; 3MTQ + 3RW; Propulsion Disturbance, Residual Dipole	4.4.8	Test ability to track many disparate variables simultaneously	Bias, momentum, and dipole tracking	Maintained 0.1° angular error. Accurate bias estimation, propulsion tracking. Good dipole tracking (10^{-2} Am^2).

[a] 3U CubeSat: Three-unit CubeSat; MTQ: Magnetorquer; RW: Reaction Wheel.

4.2 System Definition

The core of any estimation method is the state–the parameters or values that the system is trying to estimate. For this framework, the state has several components: the base state (basic rotational dynamics values), actuator biases, sensor biases, and disturbance parameters. The exact parts of each of these vary by application instance, but the form is:

$$\mathbf{x}_{\text{full}} = \begin{bmatrix} \mathbf{x}_{\text{base}}^T & \beta_{\text{actuator}}^T & \beta_{\text{sensor}}^T & \mathbf{p}_{\text{dist}}^T \end{bmatrix}^T \tag{4.1}$$

where the presence of biases (β) and disturbance parameters ($\mathbf{p}_{\text{dist}}$) depends on how the satellite is set up and whether or not the components are active (e.g., in the case of disturbances).

The base state is:

$$\mathbf{x}_{\text{base}} = \begin{bmatrix} \bar{q}^T & \boldsymbol{\omega}^T & \mathbf{h}^T \end{bmatrix}^T \tag{4.2}$$

where $\bar{q}$ is the quaternion defining orientation (converted into a 4-vector), $\boldsymbol{\omega}$ is the body-frame angular velocity, and $\mathbf{h}$ is the vector of momenta stored in each actuator (which is typically zero and thus absent, except for reaction wheels).

The system is trying to estimate the state given in Equations 4.1-4.2 using data from various inputs. This includes noisy and potentially biased (or even wholly incorrect) readings from different sensors (described further in Section 4.3.2). The system can also use the dynamics of the system, as these are the rules that govern state evolution, providing information on the state based on how it changes over time. Specifically, the dynamics of these components are governed by the Satellite class (see Section 4.3.1) as follows (allowing quaternions to operate as 4-vectors in matrix multiplication and ω to operate as a pure quaternion for multiplication in $\bar{q}\boldsymbol{w}$), where J is the inertia tensor of the satellite about the center of mass, including all reaction wheels, $j_{\text{act},i}$ is the moment of inertia of the ith reaction wheel about its axis, and $\mathbf{u}$ represents the control commands.

$$\dot{\bar{q}} = \frac{1}{2}\bar{q}\boldsymbol{\omega} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -[\boldsymbol{\omega}]^\times \end{bmatrix}\bar{q} \tag{4.3}$$

$$\dot{\boldsymbol{\omega}} = J_{\text{noRW}}^{-1}\left(-\boldsymbol{\omega}\times\left(J_{\text{noRW}}\omega + \sum_{i=1}^{N_{\text{act}}}\hat{\alpha}_{\text{act},i}h_i\right) + \tau_{\text{all}}\right) \tag{4.4}$$

$$\dot{h}_i = -\tau_{s,i} - j_{\text{act,i}}\left(\hat{\alpha}_{\text{act,i}}^T\dot{\boldsymbol{\omega}}\right) \tag{4.5}$$

where $J_{\text{noRW}} = J - \sum_{i=1}^{N_{\text{act}}} j_{\text{act},i}\hat{\alpha}_{\text{act},i}\hat{\alpha}_{\text{act},i}^T$ represents the spacecraft inertia tensor ignoring all momentum storage, and $\tau_{\text{all}} = \sum_{i=1}^{N_{\text{act}}}\boldsymbol{\tau}_{\text{act},i} + \sum_{i=1}^{N_{\text{dist}}}\boldsymbol{\tau}_{\text{dist},i}$ is the sum of all actuator and disturbance torques.

These equations define the system's "real-world" dynamics and can thus be used to complete the filter's predictive step or used in a simulation. The dynamics of other components (biases, noise, and disturbances) are contained in classes relevant to each and are described in the rest of this section.

4.2.1 Timing Questions

Timing is a crucial element in satellite attitude estimation, influencing the accuracy, stability, and overall performance of Attitude Determination and Control Systems (ADCS). Various timing-related factors, such as time-varying orbital parameters, time step size, update rates, and synchronization between sensors and estimators, must be carefully managed to ensure effective operation.

Satellite estimation must account for dynamic, time-dependent parameters that shift continuously as the satellite orbits Earth. These include orbital position and velocity, which are essential for accurately predicting disturbance torques such as gravity gradient and aerodynamic drag. The Earth's magnetic field also varies with time and location, making it necessary to update magnetic disturbance models in real time for effective control using magnetorquers or for interpreting magnetometer data. Similarly, solar radiation pressure changes based on the satellite's orientation and position relative to the Sun, impacting torque estimation and the usage of sun sensors.

Additionally, the choice of time step and update rate in the estimation framework directly affects the estimator's ability to track these time-dependent parameters. The time step, which defines how frequently the dynamic model is propagated, plays a key role in capturing high-frequency dynamics and reducing numerical integration errors. Smaller time steps generally improve accuracy but increase computational demands. Conversely, the update rate determines how often new sensor measurements are incorporated into the estimator. High update rates are beneficial for rapidly changing dynamics or when using high-noise sensors, whereas lower update rates may suffice for slower systems. Balancing these parameters is critical to achieving accurate state estimates while respecting the onboard processor's computational limitations.

For instance, a CubeSat equipped with magnetometers and sun sensors might achieve a good balance with a conveniently similar time step of 1 second and an update rate of 1 Hz. This combination allows the system to effectively track dynamic changes without overburdening onboard computational resources. However, the selection of time step and update rate might be tailored to each mission's specific requirements, factoring in the satellite's dynamics, sensor characteristics, and processing power. Different time steps are experimented with in Cases A and B, in Sections 4.4.2 and 4.4.3, respectively.

Time-varying disturbances add yet another complicating factor for ADCS estimation. Disturbances such as gravity gradient torque, aerodynamic drag, and solar radiation pressure must be accurately modeled and tracked within the estimation framework. For example, gravity gradient torque depends on the satellite's orientation and orbital position, while aerodynamic drag varies with atmospheric density, which itself fluctuates based on solar activity and altitude. Solar radiation pressure is similarly dependent on the satellite's position relative to the Sun. Estimation frameworks like the Unscented Kalman Filter (UKF) or Square-Root Unscented Kalman Filter (SRUKF) are particularly well-suited to track these time-varying parameters, improving robustness and accuracy in dynamic environments.

Tuning timing parameters, including time step size, update rate, and synchronization strategies, can affect and enhance the performance of the estimation framework. By considering the inherent timing challenges in estimation, satellites can better adapt to dynamic environments and maintain high-performance attitude control throughout their mission lifecycles.

4.3 Software Approach

This section describes the development of a modular, object-oriented software framework to enable accurate spacecraft attitude estimation in dynamic, noisy, and disturbance-prone environments. The framework integrates realistic spacecraft dynamics, sensor and actuator models, and disturbance effects with advanced estimation techniques. It is designed with three primary goals:

- Generalizability: The framework can be applied to a wide range of satellites, from large spacecraft to small, underactuated CubeSats, with varying sensor and actuator configurations.
- Dynamics-Awareness: By incorporating system dynamics, actuator commands, disturbances, and sensor biases into the estimation process, the framework improves estimation accuracy, even under low-quality or noisy sensor conditions. The disturbances and other effects (such as biases) should be estimatable when necessary, allowing their inclusion in dynamics calculations and controls.
- Flexibility: Each component–sensors, actuators, disturbances, and dynamics–can be easily customized or replaced, enabling testing and analysis of different spacecraft architectures and control methods.

The software framework has the following advantages:

1. Dynamics-Aware Estimation: A state estimator that incorporates full system dynamics, actuator noise, and disturbance models, improving accuracy over traditional methods that ignore these effects.
2. Real-Time Disturbance Tracking: Implementation of estimators capable of tracking time-varying disturbances, such as residual magnetic dipole and propulsion torques, enabling better control and planning performance.

3. Seamless Integration of Sensors and Actuators: Modular classes for sensors and actuators that account for noise, bias, and real-world constraints, ensuring realistic simulations and estimation. This object-oriented system allows for an arbitrary set of actuators and sensors for any satellite.

4. Flexible and Modular Design: An object-oriented architecture that allows easy addition or modification of components, including sensors, actuators, disturbances, and estimation methods. The disturbances and biases can be easily turned on or off for various tests, and their relevant parameters can be changed and/or allowed to evolve over time according to specific random-walk models.

5. Separation of Control Noise: A novel implementation that treats control noise separately from process noise, improving covariance estimation in Kalman-based filters.

The framework is structured around a Satellite class that serves as a digital twin of the spacecraft. This is not only used in the simulation but is also used onboard the spacecraft as a "virtual satellite" within the ADS. It captures the spacecraft dynamics, including system estimates of biases and disturbances, to allow for accurate dynamics, random-walk models, and other functions to be used for attitude determination and control.

The Satellite class, as shown in Figure 4.1, is held in a modular framework:

- Sensors: Magnetometers, gyroscopes, sun sensors, and others, modeled with bias and noise.

- Actuators: Magnetorquers, reaction wheels, and thrusters, with torque calculations and momentum tracking.

- Disturbances: Environmental torques such as gravity gradient, drag, solar radiation pressure, and residual dipole.

By holding all of this information in one class-based place, the Satellite class becomes invaluable for estimation and control, as dynamics, Jacobians, sensor covariance, and other vital functions are simple to call, even if the properties, actuators, and sensors differ between satellites.

The Satellite class is therefore a key part of estimation (and other dynamics used throughout this work). An Estimator class (with subclasses for Extended Kalman Filters (EKFs), Unscented Kalman Filters (UKFs), Square Root UKFs, etc.) manages the estimation and updates, calling the Satellite model for dynamics, sensor models, noise, and more. The Satellite class itself can also be used outside of the estimator. For example, when testing the estimator, a "real" instantiation can be created and modeled–including with properties that differ from the Estimator's understanding to test robustness. In simulation, the simulated ADS internal satellite can differ from the "real" satellite in the outer level of the simulation (though the same class can be reused for that to simplify code structure). This allows for testing in simulation the effect of mis-estimated biases, rates of random change, differences in mass parameters, etc.

The following subsections describe each component of the framework, including its mathematical formulation, implementation details, and methods for integrating it into the estimation process.

4.3.1 Satellite Class

The Satellite class serves as the core representation of the spacecraft, encapsulating its physical properties, dynamics, and components. This class acts as a digital twin, enabling realistic simulation of spacecraft behavior and state evolution. It is central to both the estimation process and the broader attitude control framework. Due to its nature as a flexible, detailed, and accurate model of spacecraft dynamics, noise, and more, it also finds ready usage in simulation and testing.

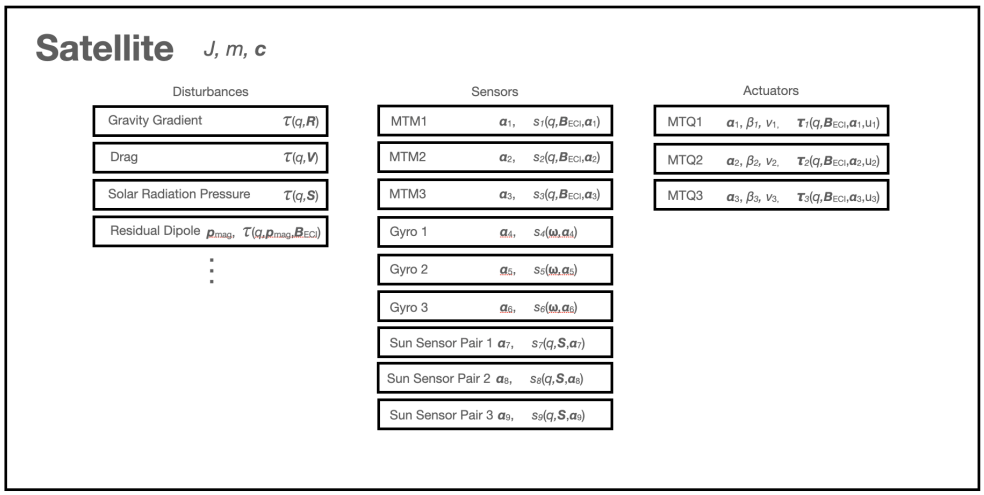


Figure 4.1: Diagram of the Satellite class containing classes for sensors, disturbances, and actuators, with their own listed parameters, as well as satellite parameters: mass m, center of mass $\mathbf{c}$, and inertia matrix J.

The Satellite class holds physical and mass parameters (inertia matrix J, mass m, center of mass position $\mathbf{c}$), as well as lists of sensors, actuators, and disturbances. Combining all of this information, the Satellite class has methods to represent important aspects of and simulate a spacecraft. These methods include:

1. Turning disturbances on or off.
2. Momentum management: generating a momentum list from actuator properties, or updating the actuator properties from given values.
3. Matching itself to a provided state (such as from the previous estimate): setting the momentum to match what is provided, setting sensor and actuator biases and disturbance parameters to match those that have been estimated, and copying their expected growth/variances to match those provided. When other commands, such as dynamics, are then run, the unit reflects the provided values. This is useful when the Satellite class is being used as a digital model of the system for an estimator or other ADCS methods, as this internal model satellite needs to be made to match the previous estimate. Additionally, this method turns off internal noise so the estimator can prescribe the noise components.

4. Update randomness: there are commands that update all of the random values inside the sensors, actuators, and disturbances, according to their internal update functions. Specifically, actuator noise and bias, sensor bias, and disturbance parameters can be updated. This is useful for simulating a satellite and can be used in particle filters or Monte Carlo-based approaches.

5. Dynamics: for a given state, orbital information,[1] and control command, find the dynamics of the satellite, including disturbances and actuator biases. The Jacobian of the state change $\dot{\mathbf{x}}$ with respect to state, control command, sensor biases (0), actuator biases, and disturbance main parameters can also be returned, as well as the Hessians of any pair of those. The Jacobians and Hessians are necessary for some estimation and control optimization techniques, such as the Extended Kalman Filter and some methods of Model Predictive Control.

6. RK4: for a given state, time step, orbital information, and control command, perform Runge-Kutta 4 integration across the time step using the dynamics function to estimate the next state of the satellite. This includes changing orbital parameters across the time step, as well as disturbances and actuator biases. The values of disturbance parameters and actuator biases are treated as constant across the time step. The Jacobian of the new state with respect to state, control command, sensor biases (though this is 0 as the sensor biases have no direct impact on dynamics), actuator biases, and disturbance main parameters can also be returned, as well as the Hessians of any of those with the control command.

   As part of the state is quaternions (see Eq. 4.2), there is the question of how to integrate them. The approach taken in this work simply numerically integrates them as vectors and normalizes at each step of RK4 (the Jacobians/Hessians of the normalization are included in the Jacobians and Hessians of this as well). It has been shown that, for well-behaved systems, this performs similarly to higher-computation "proper" geometric integration [23]. (The system developed does allow for proper geometric integration (CG5), but the Jacobians and Hessians are not yet implemented.)

7. Control bounds: bound a given control command to the limits of the actuators.

8. Control covariance: generate the covariance in the control values by taking the control noise variance from each actuator and creating a block diagonal matrix. This represents the covariance in executed control.

9. Sensor covariance: generate the covariance in the sensor readings for a given list of sampled sensors by taking the noise variance from each sensor and creating a block diagonal matrix. This represents the covariance in sensed readings.

10. Sensor information: for a given list of sampled sensors along with the base state (angular velocity and orientation, Eq. 4.2) and local vectors (sun position, Earth's magnetic field,

[1] Orbital information is passed as a separate object into dynamics and related methods. It includes values representing (in the inertial frame) the satellite's orbital position, orbital velocity, local magnetic field vector, direction towards the sun, and local atmospheric density. Depending on the disturbances in use, not all of this information will be used.

etc.), the satellite can return the list of readings from those sensors. This can be used to create simulated sensor readings (including noise) or to find the expected readings in an estimator (without noise). The Jacobians of this list with respect to the base state and the sensor biases can also be found by stacking the Jacobians of each sensor. The Jacobians are vital to an EKF.

4.3.2 Sensors

The sensor class represents all types of sensors, with specific types being represented as subclasses. Currently, in the tool as developed, there are magnetometers (MTMs), gyroscopes (gyros), GPS (not used in this discussion, which focuses on attitude), sun sensors, and sun sensor pairs.

Each sensor has properties and methods to represent important aspects:

1. Reading: Each sensor takes a reading of some property in the body frame, s_i. The reading call includes bias and noise, as described in this list, but separate methods can find the value without noise or without either bias or noise. It can also return the Jacobians of the reading generated relative to the "base state" (angular velocity and orientation, plus any stored angular momentum) and the sensor's bias, which are necessary for an EKF.

2. Bias: Each sensor can be set to have a bias or not. The bias, $\beta_{\text{sens},i}$, is added to the basic reading value as $s_i = z_i + \beta_{\text{sens},i}$. The bias can be set directly or evolve over time. When evolving over time, it is modeled as a discrete random walk approximating a Wiener process. A more detailed mathematical background can be found in Section 3.4.3. Generally, a bias rate is used, represented as $\eta_{b,\text{sens},i}$. Each time the bias evolves, the time it was updated is saved as t_{prev}. When updated again at time t, the new bias is

$$\beta'_{\text{sens},i} = N\left(\beta_{\text{sens},i}, \eta_{b,\text{sens},i}(t - t_{\text{prev}})\right), \tag{4.6}$$

   where $N(\mu, \sigma)$ is the normal distribution. The bias is updated on reading.

   Additionally, sensors have a Boolean property of "estimate_bias" to describe whether this bias is being estimated or is simply set and applied to the reading.

3. Noise: While bias changes with a low frequency, there is also high-frequency noise represented. Each sensor can be set to have noise or not. The noise, $\nu_{\text{sens},i}$, varies over time according to a customizable noise-evolution model with various parameters. The default is simply

$$\nu'_{\text{sens},i} = N(0, \eta_{\text{sens},i}), \tag{4.7}$$

   where $\eta_{\text{sens},i}$ is a settable parameter. The way that the noise function updates itself is customizable to allow for slowly changing parameters. Noise is generated and added on every call of a noisy reading (not on clean readings or readings with only bias, and this is important for the UKF). By default, noise is added to the reading and bias as $s_i = z_i + \beta_{\text{sens},i} + \nu_{\text{sens},i}$, but this is customizable.

Readings are calculated for each sensor, designated with index i, as follows:

- MTMs with axis $\hat{\alpha}_{\text{sens},i}$

$$s_i = \left(\hat{\alpha}_{\text{sens},i}^T A(\bar{q}) \mathbf{B}_{\text{ECI}}\right) + \beta_{\text{sens},i} + \nu_{\text{sens},i} \tag{4.8}$$

- Gyros with axis $\hat{\alpha}_{\text{sens},i}$

$$s_i = \left(\hat{\alpha}_{\text{sens},i}^T \omega\right) + \beta_{\text{sens},i} + \nu_{\text{sens},i} \tag{4.9}$$

- Sun sensor with efficiency ζ_i and axis $\hat{\alpha}_{\text{sens},i}$:

$$s_i = \zeta_i \max\left(0, \left(\hat{\alpha}_{\text{sens},i}^T A(\bar{q}) \frac{\mathbf{S}_{\text{ECI}}}{\|\mathbf{S}_{\text{ECI}}\|}\right)\right) + \beta_{\text{sens},i} + \nu_{\text{sens},i} \tag{4.10}$$

- Sun sensor pair with efficiency $\zeta_{i,\pm 1}$ and anti-parallel axes $\pm\hat{\alpha}_{\text{sens},i}$:

$$s_i = \zeta_{i,m}\left(\hat{\alpha}_{\text{sens},i}^T A(\bar{q}) \frac{\mathbf{S}_{\text{ECI}}}{\|\mathbf{S}_{\text{ECI}}\|}\right) + \beta_{\text{sens},i} + \nu_{\text{sens},i}, \tag{4.11}$$

  where $m = \text{sgn}\left(\hat{\alpha}_{\text{sens},i}^T A(\bar{q}) \hat{s}\right)$

This approach can also be extended to include star trackers or cameras and Earth horizon sensors.

4.3.3 Actuators

The actuator class represents all types of actuators, with specific types of actuators being represented as subclasses. Currently, there are magnetorquers (MTQs), reaction wheels (RWs), and a type of "magic" actuator that simply provides torque (this could be said to represent ideal thruster pairs, with exact torque and limitless fuel).

Each actuator has properties and methods to represent important aspects:

1. Torque generation: each actuator has an axis in the body frame that defines its axis of action, α_i. When given a command, u_i (and orbital information where relevant, such as the magnetic field for MTQs), the instance returns the torque, $\boldsymbol{\tau}_i$, created in the body frame by its actions along this axis. The default torque call includes bias and noise, as described in this list, but separate methods can find the value without noise or without either bias or noise. This is valuable for estimation. It can also return the Jacobians of torque generated relative to the "base state" (angular velocity and orientation), any stored momentum in that actuator, the command supplied, and its bias, as well as the Hessian of torque generated to any pair of those variables. There is also a maximum value of actuation that the satellite and algorithms can read to limit their commands. Generated commands above this value produce a warning.

2. Bias: each actuator can be set to have a bias or not. The bias, $\beta_{\text{act},i}$, is added to the commanded value as $u_i + \beta_{\text{act},i}$. The bias can be set directly or evolve over time. When evolving over time, it is modeled as a discrete random walk approximating a Wiener process. A more detailed mathematical background can be found in Section 3.4.3.

Generally, a bias rate is used, represented as $\eta_{b,\text{act},i}$. Each time the bias evolves, the time it is updated is saved as t_{prev}. When updated again at time t, the new bias is

$$\beta'_{\text{act},i} = N(\beta_{\text{act},i}, \eta_{b,\text{act},i}(t - t_{\text{prev}})), \tag{4.12}$$

where $N(\mu, \sigma)$ is the normal distribution and $\eta_{b,\text{act},i}$ is a settable parameter governing how quickly the bias changes. The bias is updated on command, though an additional setting in the torque function allows for it to be done whenever that function is called as well.

Additionally, actuators have a Boolean property of "estimate_bias" to describe whether this bias is being estimated or is simply set and applied to the torque.

3. Noise: while bias changes with a low frequency, high-frequency noise is also represented. Each actuator can be set to have actuation noise or not. The noise, $\nu_{\text{act},i}$, varies over time according to a customizable noise-evolution model with various parameters. The default is simply

$$\nu'_{\text{act},i} = N(0, \eta_{\text{act},i}), \tag{4.13}$$

where $\eta_{\text{act},i}$ is a settable parameter that controls the variance of the noise and $N(\mu, \sigma)$ is again the Normal distribution with mean μ and variance σ^2. Because the functions underlying torque might be called repeatedly for one time step (such as in numerical integration) or might be called twice in a way that needs to be consistent (such as torque on body and stored momentum), noise is only updated on command, not on every call (though it can be done as part of a torque call via optional argument). Noise is added to the command and bias as $u_i + \beta_{\text{act},i} + \nu_{\text{act},i}$.

4. Momentum storage: Each actuator is assigned to have momentum stored or not. It then has a moment of inertia around its axis, $j_{\text{act},i}$, and a maximum momentum that can be stored. There is also a set of functions similar to the torque generation in this list, but instead of corresponding to the torque, they are applied to the momentum storage, $\tau_{s,i}$, with the accompanying Jacobians and Hessians. Finally, it has a noise term that works like the sensor noise terms when it comes to sensing the current momentum stored (note that sensing bias is not implemented for this, but torque bias and noise are applied to this as well as the body torque).

Specifically, body torque and torque on momentum storage are calculated for each actuator, with the ith actuator assumed to be the correct class, as follows:

- Magnetorquer on axis $\hat{\alpha}_{\text{act},i}$:

$$\boldsymbol{\tau}_i = (A(\bar{q})\mathbf{B}_{\text{ECI}}) \times \hat{\alpha}_{\text{act},i}\,(u_i + \beta_{\text{act},i} + \nu_{\text{act},i}) \tag{4.14}$$

$$\tau_{\text{s,i}} = 0 \tag{4.15}$$

- Thrusters on axis $\hat{\alpha}_{\text{act},i}$:

$$\boldsymbol{\tau}_i = \hat{\alpha}_{\text{act},i}\,(u_i + \beta_{\text{act},i} + \nu_{\text{act},i}) \tag{4.16}$$

$$\tau_{\text{s,i}} = 0 \tag{4.17}$$

- RWs on axis $\hat{\alpha}_{\text{act},i}$:

$$\boldsymbol{\tau}_i = \hat{\alpha}_{\text{act},i} \left(u_i + \beta_{\text{act},i} + \nu_{\text{act},i} \right) \tag{4.18}$$

$$\tau_{\text{s,i}} = - \left(u_i + \beta_{\text{act},i} + \nu_{\text{act},i} \right) \tag{4.19}$$

This approach can also be extended to include other actuators, such as control moment gyroscopes.

4.3.4 Disturbances

The disturbance class represents all types of disturbances, with specific types being represented as subclasses. Currently, there are gravity gradient, drag, solar radiation pressure, residual dipole, propulsion torque, and an all-in-one disturbance torque. Gravity gradient, drag, and SRP are always modeled analytically. The other three have a specific parameter (moment for dipole and torque for propulsion or all-in-one disturbance) that can be set to a constant or estimated (it can also be allowed to vary over time in simulation).

Each disturbance has properties and methods to represent important aspects:

1. Torque generation: Each disturbance has a formula, based on those in Fitzgerald [108], that returns the torque in the body frame. It can also return the Jacobians of torque generated relative to the orientation. Torques that have a varying parameter, $p_{\text{dist},i}$, also have a way to calculate the Jacobian relative to that parameter, which is necessary for the EKF. It can also find the Hessian with respect to orientation and the parameter.

2. Time evolution: For the disturbances that vary over time (propulsion, all-in-one, and inherent dipole), if they are set to be time-varying, they can evolve when an update command is called. When evolving over time, they are modeled as a discrete random walk approximating a Wiener process. A more detailed mathematical background can be found in Section 3.4.3. Generally, a rate is used, represented as $\eta_{\text{dist},i}$. Each time the parameter evolves, the time is updated and saved as t_{prev}. When updated again at time t, the new parameter value is

$$\mathbf{p}'_{\text{dist},i} = N_3 \left(\mathbf{p}_{\text{dist},i}, I_{3\times 3} \left(\eta_{\text{dist},i} (t - t_{\text{prev}}) \right)^2 \right), \tag{4.20}$$

   where $N_n(\mu, \sigma^2)$ is the n-variate normal distribution. Some disturbances also have the magnitude of their main parameter capped at a maximum value, as a random walk can veer off towards infinity, but there is a limit to the torque a propulsion system can create or the strength of a satellite's inherent dipole. If so, the new value is simply divided by the norm, then multiplied by this maximum value.

3. Active/not: Disturbances have an "active" parameter, which allows them to be turned on and off throughout a run or mission. When off (active=False), the torque and relevant Jacobians/Hessians return 0.

The disturbances can also be used in control algorithms by calculating the disturbance torque and accounting for it in the control law. This is especially helpful for something like the all-in-one disturbance torque, which can eliminate the need for an integral term in some control algorithms.

4.3.5 Estimation

The **Estimator class** serves as the central framework for spacecraft state estimation by integrating dynamics, biases, disturbances, and sensor measurements into a unified representation. Acting as the "brain" of the attitude determination process, it updates the satellite state, biases, and disturbances at each time step while ensuring modularity, adaptability, and computational efficiency.

A key detail of the Estimator class is that it does not implement the estimation update method itself. Instead, it acts as a generalized interface that manages data inputs and outputs, simplifying the workload for specific estimation algorithms. The actual update logic (e.g., Extended Kalman Filter or Unscented Kalman Filter) is implemented in classes that inherit from the Estimator class. This design allows flexibility in using different estimation techniques while minimizing redundancy.

State Representation

The Estimator maintains an augmented state vector x, which includes not only the spacecraft's physical state but also biases and disturbances. It also maintains a covariance estimate of these values and information on expected integration noise. This design ensures that all sources of uncertainty are seamlessly integrated into the estimation process.

The state vector x, as described in Section 4.2, is structured as:

$$x = \begin{bmatrix} \bar{q}^T & \omega^T & h^T & \beta_{\text{act}}^T & \beta_{\text{sens}}^T & p_{\text{dist}}^T \end{bmatrix}^T, \tag{4.21}$$

where:

- $\bar{q}$: Quaternion orientation, constantly re-normalized to maintain unit length.
- ω: Angular velocity in the body frame.
- h: Stored angular momentum (e.g., reaction wheels).
- β_{act}: Actuator biases, updated as described in Equation 4.12.
- β_{sens}: Sensor biases, updated as described in Equation 4.6.
- p_{dist}: Disturbance parameters such as residual dipole torques or solar pressure forces, updated as described in Equation 4.20.

The modular design allows biases and disturbances to be included, fixed, or excluded based on the estimation scenario. If biases and disturbances are excluded, they are assumed to be zero. If they are included, the Estimator propagates their estimates and covariances over time.

Core Responsibilities

The Estimator class simplifies the process for specific filters by managing the following tasks:

State Management The Estimator holds a virtual satellite (an object of the `Satellite` class) that represents the spacecraft's current state, dynamics, sensors, actuators, and disturbances as perceived by the estimator. It maintains the full state vector, including associated covariance and process noise.

Data Management for Filters The Estimator streamlines the operation of specific estimation algorithms by:

- Saving and loading state estimates (including covariances) between iterations.
- Handling cross terms in the covariance matrix. This may include zeroing them out when certain values are turned off or setting them to zero for nonsensical terms. This behavior is adjustable.
- Dynamically including or excluding sensors based on availability while ensuring proper uncertainty propagation. For example, automatically discounting sun sensor readings during eclipse or when readings fall below their noise threshold, as when they are pointing away from the sun, there is no information to be gained from them. (This is not true, however, for sun sensor pairs. Unless in eclipse, one side of a pair should be pointing within a 180° β angle of the sun.)
- Ensuring covariance propagation when sensors are unpolled without issuing corrections, to avoid divergence but understand how far things may change over time.

Bias and Disturbance Integration The Estimator handles actuator and sensor biases, as well as time-varying disturbances:

- Actuator biases are always active to account for drift, even with zero commands.
- Sensor biases evolve at every time step, maintaining consistency with the covariance matrix. However, they are not updated if a sensor is not polled, as no new information is available.
- Disturbance parameters can be dynamically toggled on or off, with flexible reset options:
  - Reinitialize to the original state.
  - Restore the saved state prior to deactivation.[2]
  - Set to zero while retaining the saved covariance.

[2]For propulsion disturbances, which the authors expect to be the most-used case for a disturbance that is turned off and on, the saved value is used, as we expect the largest problems from propulsion to come from alignment between the satellite's center of mass and the thrust vector, which should be roughly constant. The estimator class has shortcuts for disabling intermittent aspects–such as propulsion disturbance when propulsion is turned on or off and sun-related sensors and disturbances during eclipse.

Quaternion Handling The Estimator supports both:

- Vector-based normalization: Quaternions are treated as four-element vectors and normalized after updates.
- Geometric integration: A three-element exponential representation captures quaternion evolution more explicitly.

Helper methods are provided for quaternion "addition" (multiplication), "subtraction" (multiplication by an inverse), and normalization, as well as methods to add, subtract, and compare system states with each state component added or subtracted correctly.

Inheritance and Modularity

The Estimator class provides a shared interface for specific estimation algorithms. Inherited classes implement the estimation update method. There are several options for this, but this work addresses three of them:

- Extended Kalman Filter (EKF): Linearizes dynamics using first-order approximations.
- Unscented Kalman Filter (UKF): Uses sigma points for better handling of nonlinearities.
- Square-Root UKF (SRUKF): Ensures numerical robustness by propagating the square-root covariance.

Other sensor fusion techniques can also be implemented within this modular framework.

4.3.5.a Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) is implemented as a subclass of the Estimator class. It defines the specific method for the measurement update step, leveraging the general prediction architecture of the base Estimator class. The EKF estimates the spacecraft state by linearizing the system dynamics and measurement models using first-order approximations.

Prediction Step During the prediction step, the spacecraft state x and covariance P are propagated forward in time using the nonlinear dynamics model inside the virtual satellite (see Section 4.3.1), combined with environmental information and control inputs. Runge-Kutta 4 (RK4) numerical integration ensures accurate state propagation, while the covariance propagation accounts for process noise and control uncertainties.

The state is first propagated forward as:

$$x_{k+1|k} = f(x_k, u_k), \tag{4.22}$$

where f is the nonlinear dynamics model provided by the Satellite class. The covariance is propagated as:

$$P_{k+1|k} = G_k P_{k|k} G_k^T + B_k U_k B_k^T + Q_k \tag{4.23}$$

where:

- G_k is the state transition Jacobian, linearized numerically using finite differences during RK4 propagation.
- $B_k U_k B_k^T$ is the control noise contribution. See Section 4.3.5.b.
- Q_k is the process or integration noise.

Computation of G_k and B_k Matrices The Jacobians G_k (state transition matrix) and B_k (control input matrix) are constructed numerically based on Jacobians from the Satellite class. Specifically:

- The Jacobians are first computed for the dynamics state (orientation, angular velocity, and momentum) based on the Jacobians built into the Satellite class.
- The matrices are then expanded to include the full state (see the form in Equation 4.1), which accounts for sensor and actuator biases as well as disturbances:

$$G_k = \begin{bmatrix} A_{\text{dyn}} & 0 & & 0 & 0 \\ 0 & I_{\text{bias,act}} & 0 & 0 & \\ 0 & 0 & I_{\text{bias,sens}} & 0 & \\ 0 & 0 & 0 & I_{\text{dist}} & \end{bmatrix}, \quad B_k = \begin{bmatrix} B_{\text{dyn}} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.24}$$

  Here:

  - A_{dyn} and B_{dyn} are the Jacobians of the dynamics state, computed during RK4 in the Satellite class (and including all of the steps of RK4 in their calculation).
  - $I_{\text{bias,act}}$, $I_{\text{bias,sens}}$, and I_{dist} are identity blocks corresponding to the actuator and sensor biases, ensuring these parameters persist during propagation.

This structure ensures that the full state, including biases and disturbances, is appropriately propagated while maintaining the sparsity of the Jacobian matrices.

Quaternion Integration and Jacobians As described in Section 4.3.1, quaternions are integrated as vectors during the RK4 propagation step (unless the integration method is changed). After each step, quaternions are normalized to enforce unit length. The Jacobians used for covariance propagation account for the effects of quaternion normalization, ensuring consistency between the state and covariance evolution.

Measurement Update Step The EKF corrects the predicted state and covariance using sensor measurements. The measurement Jacobian H_k is constructed by stacking blocks for the state and bias Jacobians. Specifically:

$$H_k = \begin{bmatrix} H_{\text{state}} \\ 0 \\ H_{\text{bias,sens}} \\ 0 \end{bmatrix}. \tag{4.25}$$

Here:

- H_{state} is the partial derivative of the sensor models $h(x)$ with respect to the state, pulled from the Satellite object, based on which sensors are in use.
- $H_{\text{bias,sens}}$ accounts for the sensor biases (largely an identity matrix sandwiched by zeros, but pulled from the Satellite model and could theoretically be something else), ensuring consistency when estimating bias parameters alongside the state.

Inactive sensors (e.g., sun sensors in eclipse or with near-zero predicted readings) are excluded by setting their corresponding rows in H_k and predicted measurements to zero.

The Kalman gain K_k is computed using the measurement covariance R_k and the predicted covariance $P_{k+1|k}$:

$$K_k = P_{k+1|k} H_k^T \left(H_k P_{k+1|k} H_k^T + R_k \right)^{-1}. \tag{4.26}$$

Here, R_k is the measurement noise covariance. Specifically, the gain computation is implemented using the `scipy.linalg.solve` function for numerical robustness.

The state is corrected as:

$$x_{k+1|k+1} = x_{k+1|k} + K_k(z_k - h(x_{k+1|k})), \tag{4.27}$$

where z_k is the actual sensor measurement and $h(x_{k+1|k})$ is the predicted measurement.

The covariance is updated as:

$$P_{k+1|k+1} = P_{k+1|k} \left(I - H_k^T K_k^T \right). \tag{4.28}$$

For numerical stability, the covariance matrix is symmetrized after the update: $P_{k+1|k+1} = \frac{1}{2} \left(P_{k+1|k+1} + P_{k+1|k+1}^T \right)$.

EKF Summary The EKF provides an efficient and tractable method for nonlinear state estimation by leveraging first-order linearization. Quaternions are integrated and normalized at each step, while the covariance propagation accounts for normalization effects. The EKF integrates seamlessly with the Estimator class, benefiting from its state management, bias handling, and sensor availability features.

4.3.5.b Kalman Control Noise

In this work, control noise is separated from process noise in the Extended and Unscented Kalman Filters. The implementation of control noise in the UKF simply adds sigma points corresponding to the variance in control. The EKF, however, requires the control noise to be added with the state dynamics Jacobians. Consider:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{u} + \nu_{\text{control}}, t\right). \tag{4.29}$$

where $\mathbf{x}$ is the state, $\mathbf{u}$ is the control, ν_{control} is the control noise, and $\mathbf{f}$ represents the time evolution of the state. The t component captures things that vary with orbit, like atmospheric density and the magnetic field, as discussed in Section 4.2.1.

Discretizing Equation 4.29 between time steps:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t) + \nu_{\text{control}}(t), t\right) dt \\ &= g\left(\mathbf{x}_k;\ t_k, t_{k+1};\ \mathbf{u}_k + \tilde{\nu}_{\text{k}}\right) + \nu_{\text{int,k}}\end{aligned} \tag{4.30}$$

where $\mathbf{u}$ is treated as constant throughout the integration step, $\tilde{\nu}$ is a noise value that corresponds to the time-integration of the random process ν_{control} over the given time interval, g is the selected method of numerical integration, and $\nu_{\text{int,k}}$ is the error from numerical integration. For clarity, we write $g\left(\mathbf{x}_k;\ t_k, t_{k+1};\ \mathbf{u}_k + \tilde{\nu}_{\text{k}},\right) = g_k\left(\mathbf{x}_k;\ \mathbf{u}_k + \tilde{\nu}_{\text{k}},\right)$.

We thus model the predictive step in an EKF as:

$$\hat{\mathbf{x}}_{k+1|k} = E\left[g_k\left(\mathbf{x}_k;\ \mathbf{u}_k + \tilde{\nu}_{\text{k}}\right) + \nu_{\text{int,k}}\right] = g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right), \tag{4.31}$$

assuming the control noise and integration noise are zero-mean. The covariance of this is

$$\begin{aligned}P_{k+1|k} &= E\left[\mathbf{x}_{k+1}\mathbf{x}_{k+1}^T\right] - E\left[\mathbf{x}_{k+1}\right] E\left[\mathbf{x}_{k+1}\right]^T \\ &= E\left[\mathbf{x}_{k+1}\mathbf{x}_{k+1}^T\right] - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T.\end{aligned} \tag{4.32}$$

Applying Equation 4.30 gives

$$\begin{aligned}P_{k+1|k} = E\left[\left(g_k\left(\mathbf{x}_k;\ \mathbf{u}_k + \tilde{\nu}_{\text{k}}\right)\right)\left(g_k\left(\mathbf{x}_k;\ \mathbf{u}_k + \tilde{\nu}_{\text{k}}\right)\right)^T\right] \\ + \operatorname{cov}\left(\nu_{\text{int,k}}\right) - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T,\end{aligned} \tag{4.33}$$

assuming the integration error is independent of the state and control error. We approximate the system as

$$\begin{aligned}P_{k+1|k} = E\left[\left(g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right) + B_k\tilde{\nu}_{\text{k}}\right)\left(g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right) + B_k\tilde{\nu}_{\text{k}}\right)^T\right] \\ + \operatorname{cov}\left(\nu_{\text{int,k}}\right) - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T,\end{aligned} \tag{4.34}$$

where B_k is the Jacobian of $g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right)$ with respect to $\mathbf{u}_k$. Our system can then be rewritten, treating the control error as zero-mean:

$$\begin{aligned}P_{k+1|k} = E\left[g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right) g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right)^T\right] \\ + B_k\operatorname{cov}\left(\tilde{\nu}_k\right) B_k^T \\ + \operatorname{cov}\left(\nu_{\text{int,k}}\right) - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T.\end{aligned} \tag{4.35}$$

We use $U_k = \operatorname{cov}\left(\tilde{\nu}_k\right)$ and $Q_k = \operatorname{cov}\left(\nu_{\text{int,k}}\right)$. Linearizing around $\hat{\mathbf{x}}_{k|k}$, we have

$$\begin{aligned}P_{k+1|k} = E\left[g_k\left(\hat{\mathbf{x}}_{k|k} + \delta\mathbf{x}_k;\ \mathbf{u}_k\right) g_k\left(\hat{\mathbf{x}}_{k|k} + \delta\mathbf{x}_k;\ \mathbf{u}_k\right)^T\right] \\ + B_kU_kB_k^T + Q_k - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T.\end{aligned} \tag{4.36}$$

Approximating with G_k as the derivative of $g_k\left(\mathbf{x}_k;\ \mathbf{u}_k\right)$ with respect to $\mathbf{x}_k$:

$$\begin{aligned}P_{k+1|k} = E\left[g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right) g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right)^T\right] \\ + G_k\operatorname{cov}\left(\delta\mathbf{x}_k\right) G_k^T \\ + B_kU_kB_k^T + Q_k - \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T.\end{aligned} \tag{4.37}$$

Because $\delta\mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$, we have $\text{cov}(\delta\mathbf{x}_k) = P_{k|k}$. As we have accounted (with varying assumptions) for all sources of error, we have

$$E\left[g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right) g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right)^T\right] = g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right) g_k\left(\hat{\mathbf{x}}_{k|k};\ \mathbf{u}_k\right)^T = \hat{\mathbf{x}}_{k+1|k}\hat{\mathbf{x}}_{k+1|k}^T \tag{4.38}$$

exactly, so we find:

$$P_{k+1|k} = G_k P_{k|k} G_k^T + B_k U_k B_k^T + Q_k, \tag{4.39}$$

which is similar to the conventional EKF expression, with only an added control noise term.

It is worth noting that Equations 4.31-4.39 include control error, error in state estimation, integration error, and disturbing torques. If the model is used and the disturbing torques are erroneous (or some disturbance is not included in the state but is present in reality), then Equation 4.30 needs to include a noise torque term. Assuming this is randomly distributed between times, independent of other errors, and zero-mean, that torque noise can be treated similarly to the $\nu_{int,k}$ term. (If these assumptions are incorrect, then it should be approximated or represented as an all-in-one torque, as described in Section 3.4.4.c.) The equations are the same, except with $Q_k = \text{cov}(\nu_{\text{int,k}}) + H J_{x,\omega} \text{cov}\left(\int_{t_k}^{t_{k+1}} \boldsymbol{\tau}_{\text{dist}}(t)dt\right) J_{x,\omega}^T H^T$ or $Q_k = \text{cov}(\nu_{\text{int,k}}) + C_k T_k C_k^T$, where T_k is the covariance integral of the unestimated torque and C_k is the derivative of g_k with respect to a torque term; it is equal to the columns of G_k corresponding to an all-in-one disturbance torque.

4.3.5.c Unscented Kalman Filter (UKF)

The Unscented Kalman Filter (UKF) is implemented as a subclass of the Estimator class. Unlike the EKF, which relies on linearized Jacobians, the UKF propagates a set of sigma points through the nonlinear dynamics and measurement models. These sigma points provide a more accurate representation of the state distribution, making the UKF better suited for highly nonlinear systems.

Sigma Point Generation and Weights The UKF generates a set of sigma points χ around the current state x_k using the Cholesky square root of the state covariance P_k, ensuring that the mean and covariance are preserved. The spread and weighting of the sigma points are determined by three parameters:

- α: Determines the spread of the sigma points around the mean. A typical value is $\alpha = 10^{-3}$.
- β: Incorporates prior knowledge of the distribution; for Gaussian distributions, $\beta = 2$ is optimal.
- κ: Controls the scaling of sigma points. A common choice is $\kappa = 3 - L$, where L is the state dimension.

The parameter λ is defined as:

$$\lambda = \alpha^2(L + \kappa) - L, \tag{4.40}$$

where L is the dimension of the state.

The sigma points are generated as:

$$\chi_0 = x_k, \quad \chi_i = x_k + \sqrt{(L+\lambda)P_k}_i, \quad \chi_{i+L} = x_k - \sqrt{(L+\lambda)P_k}_i, \quad i = 1, \ldots, L, \tag{4.41}$$

where $\sqrt{(L+\lambda)P_k}$ is the scaled square root of the covariance matrix, computed via Cholesky decomposition. Quaternions are dealt with appropriately (as if they are dimension 3), as described in [18].

The weights for the mean ($W^{(m)}$) and covariance ($W^{(c)}$) are:

$$W_0^{(m)} = \frac{\lambda}{L+\lambda}, \quad W_i^{(m)} = \frac{1}{2(L+\lambda)}, \quad i = 1, \ldots, 2L, \tag{4.42}$$

$$W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta), \quad W_i^{(c)} = \frac{1}{2(L+\lambda)}, \quad i = 1, \ldots, 2L. \tag{4.43}$$

Noise Handling in Sigma Point Generation The Cholesky decomposition is performed not just for the state covariance, but also for the other sources of noise, creating sets pf sigma points that correspond to each source of uncertainty.

- **Integration Noise**: Integration noise can be included or excluded based on the flag settings. If separate, it is treated as additional sigma points.
- **Sensor Noise**: Sensor noise contributions can be explicitly separated into sigma points, enabling fine control over noise propagation.
- **Control Noise**: Control noise is incorporated directly into the dynamics propagation step by modifying the sigma points before integration.

This noise separation ensures that the effects of different uncertainties are handled cleanly without being combined into a single process noise term.

Prediction Step The state sigma points are propagated through the nonlinear dynamics $f(x, u)$ using RK4 integration:

$$\chi_{k+1|k} = f(\chi_k, u_k), \tag{4.44}$$

where f is the dynamics model provided by the Satellite class. The control noise sigma points (if included separately) are created by propagating the system mean with the control noise:

$$\chi_{\gamma,k+1|k} = f(x_k, u_k + \gamma), \tag{4.45}$$

where γ are the sigma points from control noise. This allows for nonlinear control effects to be better modeled by the dynamics than adding control error as part of process noise. Similarly, the integration noise sigma points (if included separately) are created by propagating the system mean, then adding integration noise:

$$\chi_{\iota,k+1|k} = f(x_k, u_k) + \iota, \tag{4.46}$$

where ι are the sigma points from process or integration noise. (The addition here may not be normal addition but is instead "state vector" addition as controlled by the Estimator class.

For most vectors, addition will proceed normally, but for things like quaternions, it will be the appropriate method.)

The predicted state $x_{k+1|k}$ and covariance $P_{k+1|k}$ are reconstructed from the propagated sigma points:

$$x_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} \chi_{i,k+1|k}, \quad P_{k+1|k} = \sum_{i=0}^{2L} W_i^{(c)} (\chi_{i,k+1|k} - x_{k+1|k})(\chi_{i,k+1|k} - x_{k+1|k})^T + Q_k, \tag{4.47}$$

where Q_k is the integration or process noise covariance (when used instead of sigma points).

Measurement Update Step The sigma points are propagated through the measurement model $h(x)$ to predict sensor outputs:

$$Z_i = h(\chi_{i,k+1|k}). \tag{4.48}$$

The sensor noise sigma points (if included separately) are propagated similarly:

$$Z_i = h(\chi_{0,k+1|k}) + \mu, \tag{4.49}$$

where μ are the sigma points from sensor noise.

The predicted measurement $z_{k+1|k}$ and its covariance S_k are computed as:

$$z_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} Z_i, \quad S_k = \sum_{i=0}^{2L} W_i^{(c)} (Z_i - z_{k+1|k})(Z_i - z_{k+1|k})^T + R_k, \tag{4.50}$$

where R_k is the measurement noise covariance (when used instead of sigma points).

The cross-covariance between the state and measurements is:

$$P_{xz} = \sum_{i=0}^{2L} W_i^{(c)} (\chi_{i,k+1|k} - x_{k+1|k})(Z_i - z_{k+1|k})^T. \tag{4.51}$$

The Kalman gain K_k is computed as:

$$K_k = P_{xz} S_k^{-1}. \tag{4.52}$$

The state and covariance are updated as:

$$x_{k+1|k+1} = x_{k+1|k} + K_k(z_k - z_{k+1|k}), \quad P_{k+1|k+1} = P_{k+1|k} - K_k S_k K_k^T. \tag{4.53}$$

For numerical stability, the covariance matrix is symmetrized after the update:

$$P_{k+1|k+1} = \frac{1}{2}\left(P_{k+1|k+1} + P_{k+1|k+1}^T\right). \tag{4.54}$$

Bias and Disturbance Handling The UKF handles sensor biases, actuator biases, and disturbances as part of the state vector. These parameters are propagated through the sigma points and updated alongside the dynamics-based system state.

UKF Summary The UKF provides an improved method for nonlinear state estimation by propagating sigma points through the dynamics and measurement models. It avoids the need for explicit Jacobians, making it well-suited for highly nonlinear systems. The UKF seamlessly integrates with the Estimator class and handles biases and disturbances robustly.

4.3.5.d Square Root Unscented Kalman Filter (SRUKF)

The Square-Root Unscented Kalman Filter (SRUKF) is implemented as a subclass of the Estimator class. It builds upon the UKF (Section 4.3.5.c) but improves numerical stability by propagating the square root of the covariance matrix instead of the full covariance. This ensures the positive semi-definiteness of the covariance matrix and mitigates numerical conditioning issues during long-term estimation.

Sigma Point Generation and Weights The SRUKF follows the same sigma point and weight generation scheme as the UKF, as in Section 4.3.5.c. A set of sigma points χ is generated around the current state x_k, using the square root of the covariance matrix S_k, where S_k satisfies $P_k = S_k S_k^T$.

Prediction Step The state sigma points are propagated through the nonlinear dynamics model $f(x, u)$ using RK4 integration, identically to the UKF:

$$\chi_{i,k+1|k} = f(\chi_{i,k}, u_k), \tag{4.55}$$

where f is the dynamics model provided by the Satellite class. When sigma points are used for integration and control noise, they are propagated

$$\chi_{\gamma,k+1|k} = f(x_k, u_k + \gamma), \tag{4.56}$$

where γ are the sigma points from control noise, and

$$\chi_{\iota,k+1|k} = f(x_k, u_k) + \iota, \tag{4.57}$$

where ι are the sigma points from process or integration noise.

The square root of the predicted covariance $S_{k+1|k}$ is updated using a numerically stable QR decomposition. First, the deviations of the propagated sigma points from the predicted mean $x_{k+1|k}$ are stacked as columns:

$$\Delta\chi_i = \chi_{i,k+1|k} - x_{k+1|k}. \tag{4.58}$$

The updated square root covariance $S_{k+1|k}$ is obtained as:

$$S_{k+1|k} = \text{qr}\left(\left[\sqrt{W_i^{(c)}}\Delta\chi_i, \text{ integration noise}\right]\right), \tag{4.59}$$

where the QR decomposition ensures numerical robustness. When process or integration noise sigma points are used, there is no need to add any process integration noise matrix here, as they are accounted for separately as described in Equation 4.57. In typical UKF approaches, control noise is treated as process noise, but it can be added instead as in Equation 4.56, reducing integration and process noise (even if not using integration sigma points).

Measurement Update Step The sigma points are propagated through the measurement model $h(x)$ to predict the sensor outputs:

$$Z_i = h(\chi_{i,k+1|k}). \tag{4.60}$$

The sensor noise sigma points (if included separately) are propagated similarly:

$$Z_i = h(\chi_{0,k+1|k}) + \mu, \tag{4.61}$$

where μ are the sigma points from sensor noise.

The predicted measurement $z_{k+1|k}$ is computed as:

$$z_{k+1|k} = \sum_{i=0}^{2L} W_i^{(m)} Z_i. \tag{4.62}$$

The deviations of the predicted measurements Z_i from the mean measurement $z_{k+1|k}$ are stacked, and the square root of the measurement covariance S_z is updated using QR decomposition:

$$S_z = \text{qr}\left(\left[\sqrt{W_i^{(c)}}(Z_i - z_{k+1|k}), \text{ sensor noise}\right]\right). \tag{4.63}$$

When sensor noise sigma points are used, there is no need to add any sensor noise matrix here, as it is taken care of in Equation 4.61.

The cross-covariance between the state and the measurements is computed as:

$$S_{xz} = \sum_{i=0}^{2L} W_i^{(c)}(\chi_{i,k+1|k} - x_{k+1|k})(Z_i - z_{k+1|k})^T. \tag{4.64}$$

The Kalman gain K_k is obtained as:

$$K_k = S_{xz} S_z^{-T}. \tag{4.65}$$

In practice, this is performed with two operations of `np.linalg.lstsq`.

The state is updated as:

$$x_{k+1|k+1} = x_{k+1|k} + K_k(z_k - z_{k+1|k}). \tag{4.66}$$

Once again, state "addition" may be different for mathematical objects like quaternions.

The square root covariance is updated using a Cholesky rank-one update (or downdate):

$$S_{k+1|k+1} = \text{cholupdate}(S_{k+1|k}, K_k S_z, -1). \tag{4.67}$$

Bias and Disturbance Handling The SRUKF handles sensor biases, actuator biases, and disturbances as part of the state vector. These parameters are included in the sigma points, propagated, and updated consistently with the dynamic state.

SRUKF Summary The SRUKF improves numerical stability and computational efficiency compared to the standard UKF by propagating the square root of the covariance matrix. Using QR decomposition and Cholesky updates ensures positive semi-definiteness of the covariance throughout the estimation process. The SRUKF handles integration, control, and sensor noise explicitly and integrates seamlessly with the Estimator class for flexible state and bias tracking.

4.3.5.e Estimation of Covariance Terms

A key challenge in many Kalman filters–of all kinds–and related approaches is the estimation of various noise sources. A traditional Kalman Filter, as described in Section 3.4.5, typically has two matrices for this: a process noise term Q_k, which describes how much uncertainty is added by the forward propagation; and a measurement noise term R_k, which describes how much noise there is on the measurements. These are often described as covariances, as they are supposed to approximate the covariance of the relevant parameter or process.

Different estimation approaches may use additional or different terms, but this captures the idea. Choosing these uncertainty values for a given system can be a difficult step in setting up an estimator. There are many different suggested approaches, but trial-and-error and tuning either by hand or with a simulation loop is common. However, some of these values can be found more directly. The filter in this work splits the process noise between integration error, control noise, controller biases, and disturbances, and the measurement error into sensor bias and sensor uncertainty. The aim of this is to simplify choosing the error parameters for a future user by instead relying on values that can be more directly estimated from data sheets (like sensor noise and drift rate) or mathematical analysis (like integration error[3]). Further refinement of this approach may yield additional benefits, but it already creates a more straightforward setup for estimators.

4.3.5.f Summary

The Estimator, and its children class, provides a versatile and robust framework for spacecraft state estimation. By centralizing state management, propagating dynamics, handling biases and disturbances, and ensuring data consistency, it simplifies the implementation and operation of advanced estimation algorithms. Its modularity and flexibility make it adaptable to a wide range of mission scenarios.

4.4 Tests and Results

This framework and the estimator from this chapter are tested in several ways. First, in Cases A and B (Sections 4.4.2 and 4.4.3), the approach from this work is compared against a well-regarded traditional estimator. It is tested for a (simulated) large satellite on which the traditional estimator is demonstrated. The difference between Cases A and B is that

[3]Perhaps the integration error for each timestep to a formula based on integration error bound/estimation formulas for RK4. This could further simplify estimator initialization. A method to do so was partially explored (and is still in the estimation code at https://github.com/patrickmckeen/PhD_Dissertation_Code).

Case A starts with larger errors and thus shows convergence, while Case B has smaller initial errors and is more representative of behavior once the estimators have converged. Cases C and D (Sections 4.4.4 and 4.4.5) then show the behavior of the dynamics-aware filter on a nanosatellite, with worse sensors and less inertia. Case C represents the system behavior when starting with large error and converging. Case D shows the behavior when starting with small error and continuing. Cases E and F are used to explore the importance of certain aspects of the dynamics-aware filter. Specifically, Case E shows the effects of ignoring the actuator bias or the disturbances, while Case F demonstrates the importance of considering a propulsion torque, when present, and shows how well the estimator can converge on the torque value.

4.4.1 Simulation Information

There are many choices and parameters that go into a satellite attitude estimation system. There are even more choices to be made, including details and information relevant to replicating the results or implementing the approach, when there is a simulation included. Information is necessary on the spacecraft (both simulated and "internal" to the estimator–what the ADS knows about itself), the sensors and actuators, the orbit, and the simulation itself. In each case, further details are provided in relevant tables. However, some information is included here to clarify details of the approach.

The "real" dynamics are integrated using an RK45 variable time-step solver between time steps. Sensor and actuator biases, as well as disturbances, are updated as described in Sections 3.4.4.a, 3.4.4.b, and 3.4.4.c, respectively, with random distributions. For comparison with expected values on the magnetometer and sun sensor, the position of the satellite is assumed to be known exactly. The magnetic field is calculated with the 13th generation IGRF (up to degree $N = 13$), and the sun position information is obtained with the Python package Skyfield. Disturbances are modeled with exact information. However, there may be discrepancies between the disturbance experienced by the "real" satellite and the virtual satellite based on errors in the orientation estimation. For example, if the estimated orientation is such that the major principal axis of the satellite is pointing nadir, it will estimate zero gravity gradient torque, while the "real" satellite may experience a torque corresponding to its actual orientation.

In many cases, the initial attitude can be approximated by combining the measurements from the sun sensors and magnetometers, resulting in rapid convergence. This is certainly possible (and largely implemented). However, such initialization is not used in these cases or this work, primarly to allow a more direct comparison to USQUE in Cases A and B. It also highlights the ability of the filter to learn attitude (comparatively) slowly through multiple measurements, when information is incomplete or when initialized without a good starting guess.

4.4.2 Case A: TRMM Satellite with Initial Attitude and Bias Errors

This system is first tested against a simple baseline attitude filter under a variety of conditions. The baseline is the simple UnScented QUaternion Estimator (USQUE) by Crassidis and Markley [18]. A shorter time span is used for the 1s update period cases (12 hours instead of

24 hours) due to memory limitations. There are more involved and mission-specific filters, including versions of the unscented attitude filter that account for sensor bias and other factors, but this is a well-behaved and generalizable filter that could easily be the basis of an approach for a nanosatellite project. The test case in the USQUE paper [18] (specifically, the Tropical Rainfall Measuring Mission (TRMM) with initial attitude and bias errors) is recreated, with the addition of all disturbances (except propulsion) described in [108], as well as active magnetic control with MTQs, and biases on the sensors and actuators. Detailed estimation information can be found in Table 4.2.

Table 4.2: Satellite Properties and Test Parameters for Case A

Property	**Value/Description**
Mass	$m = 3000$ kg
Inertia Matrix	$J = 500 \cdot \text{diag}([1, 3, 3])$ kg·m^2
MTQ	Maximum Magnetic Moment: 100 Am^2 Initial Bias: $5 * \begin{bmatrix} \frac{1}{3\sqrt{2}} & \frac{1}{3\sqrt{2}} & \frac{4}{3\sqrt{2}} \end{bmatrix}$ Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ Bias Drift Rate: 0.000001 Am^2/s$^{0.5}$
Gyroscope[a]	Initial Bias: $[0.1, 0.1, 0.1]$ deg/hr Noise Std Dev: 0.31623 μrad/s$^{0.5}$ Bias Drift Rate: 3.1623×10^{-4} μrad/s$^{1.5}$
Magnetometer	Initial Bias: $[-0.9948, -0.0199, -0.0995]$ nT Noise Std Dev[b] 50 nT·s$^{0.5}$ Bias Drift Rate: 1 nT/s$^{0.5}$
Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0.015, 0.027, -0.009]$ (unitless) Noise Std Dev: 0.0003 s$^{0.5}$ Bias Drift Rate: 0.000003 s$^{-0.5}$
Residual Dipole	Initial Value: $[0.5, 0.001, 2]$ Am^2 Capped Max: 2 Am^2 Drift Rate: 0.0001 Am^2/s$^{0.5}$
Estimated MTQ	Maximum Magnetic Moment: 100 Am^2 (not used in USQUE by definition) Initial Bias: $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ (not used in USQUE by definition) Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ (not used in USQUE by definition) Bias Drift Rate: 0.000001 Am^2/s$^{0.5}$ (not used in USQUE by definition)
Estimated Gyroscope	Initial Bias: $[0, 20.1, 0]$ deg/hr (based on [18]) Noise Std Dev: 0.31623 μrad/s$^{0.5}$ Bias Drift Rate: 3.1623×10^{-4} μrad/s$^{1.5}$
Estimated Magnetometer	Initial Bias: $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ nT

Continued on next page

Table 4.2 – *Continued from previous page*

Property	**Value/Description**
	Noise Std Dev: 50 nT·$s^{0.5}$ Bias Drift Rate: 1 nT/$s^{0.5}$ (MTM bias ignored in USQUE)
Estimated Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0, 0, 0]$ (unitless) Noise Std Dev: 0.0003 $s^{0.5}$ Bias Drift Rate: 0.000003 $s^{-0.5}$ (Sun sensor bias ignored in USQUE)
Estimated Residual Dipole	Initial Value: $[0, 0, 0]$ Am^2 (not used in USQUE by definition) Capped Max: 2 Am^2 (not used in USQUE by definition) Drift Rate: 0.0001 $Am^2/s^{0.5}$ (not used in USQUE by definition)
Orbit Parameters	Circular orbit Initial Position: 7200 km along ECI x-axis Inclination: 35° Initial Velocity: 7.4 km/s, perpendicular to orbital position J2 effects included
Update Period	1 second or 10 seconds
Simulation Duration	12 hours (1s timestep, shorter due to memory limitations) or 24 hours (10s timestep)
Control Algorithms	No Control for first 30 min B-dot (with numeric differentiation) applied from 30 min until 2 hours; Gain: 10^{10} Lovera Magnetic PD Algorithm applied after 2-hour mark: Gain (Proportional: 10, Velocity: 1000)
Quaternion Pointing Goals	0h to 2h: None (no control or B-dot) 2h to 6h (3h in 12h case): [1,0,0,0] 6h to 12h (3h to 6h in 12h case): [0,0,1,0] 12h to 18h (6h to 9h in 12h case): [0,1,0,0] 18h to 24h (9h to 12h in 12h case): [1,0,0,0]
Initial State	Angular Velocity: $[0, \frac{1}{15}, 0]$ deg/s Quaternion: $[0.63, -0.33, -0.63, 0.33]$ (body z-axis is pointing nadir and body x-axis is pointing ram)
Estimated Initial State	Angular Velocity: $[0, 0, 0]$ deg/s (not used by definition in USQUE)

Continued on next page

Table 4.2 – *Continued from previous page*

Property	**Value/Description**
	Quaternion: $[-0.16, -0.85, 0.31, 0.39]$ (from third case in [18], rotations of -50, 50, and 160 degrees around x, y, and z axes, calculated as an MRP from the real value)
USQUE Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(50°)^2$ Initial $P_{\text{gyro bias}}$: $(20°/\text{hr})^2$ Integration covariances calculated as in [18]
Dynamics-Aware Filter Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(50°)^2$ Initial $P_{\text{gyro bias}}$: $(20°/\text{hr})^2$ Initial P_{ω}: $(20°/\text{hr})^2$ Initial $P_{\text{mtq bias}}$: $(5\ \text{Am}^2)^2$ Initial $P_{\text{mtm bias}}$: $(5\ \text{nT})^2$ Initial $P_{\text{sun bias}}$: $(0.03)^2$ Initial P_{dipole}: $(1\ \text{Am}^2)^2$ Integration covariances on biases match bias drift rates times Δt Attitude integration covariance: $10^{-12}(\Delta t)^2\ \text{rad}^2$ (based on experimentation) Angular velocity integration covariance: $10^{-17}(\Delta t)\ \text{rad}^2/\text{s}^2$ (based on experimentation)
SRUKF Parameters	$\alpha = 1$ $\beta = 2$ $\kappa = 3 - (L + 3)$ (with the additional 3 added for control noise and L is the state length)

[a] Based on [18].

[b] This is 40% of the value in [27], to account for this being a higher-quality mission with better sensors.

The results for Case A demonstrate the clear advantages of the dynamics-aware filter compared to the baseline Unscented Quaternion Estimator (USQUE) in terms of accuracy, convergence speed, and robustness to system dynamics. Figure 4.2 illustrates the overall angular error over time, where the dynamics-aware filter achieves rapid convergence to an error of approximately 0.01° for both the 1 s and 10 s time steps. In contrast, USQUE stabilizes at a significantly higher error, between 1° and 10°, with slower convergence. This performance difference highlights the importance of accounting for system dynamics, control efforts, and disturbances, which are incorporated in the dynamics-aware filter. While the USQUE is a brilliant filter, it does not deal well with disturbances or biased attitude sensors, even if they are otherwise high-quality.

There is a small but interesting detail in Figure 4.2, where there is a difference in the initial convergence behavior of USQUE compared to the dynamics-aware filter. USQUE appears to converge more quickly within the first 30 minutes, stabilizing at an angular error of

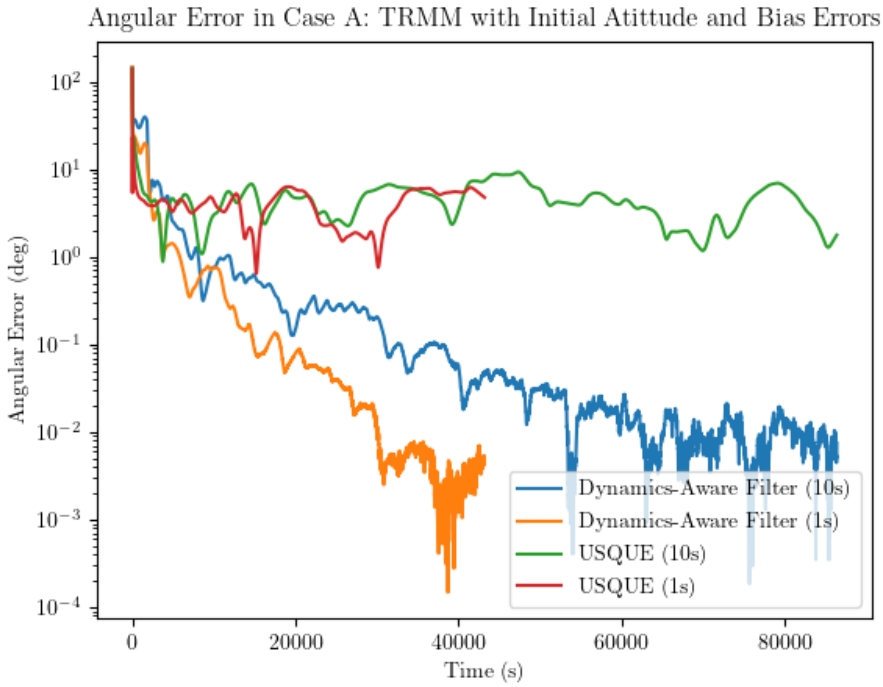


Figure 4.2: Angular estimation error in Case A (Section 4.4.2, TRMM with Initial Attitude and Bias Errors, case details in Table 4.2), showing quicker convergence and higher accuracy with the Dynamics-Aware Filter. The 1s time step simulations run for 12 hours instead of 24 (unlike the 10s time step simulations) due to memory constraints.

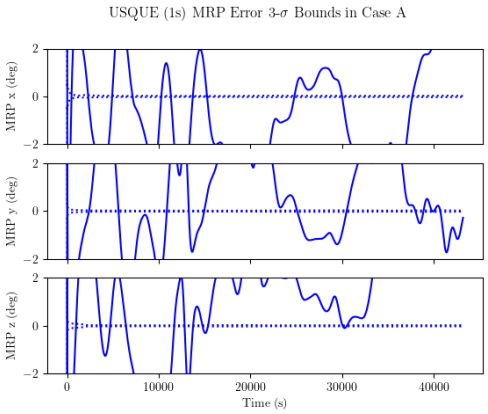
approximately 4°, around its steady-state error. The source is unclear, but its behavior could potentially be explained by the simplicity of USQUE's state representation, which focuses solely on attitude without accounting for additional factors such as biases or disturbances. The lower dimensionality of its state space allows USQUE to converge more quickly to its final accuracy level.

In contrast, the dynamics-aware filter must estimate a more complex state that includes biases and disturbances, enabling it to achieve a much higher level of final accuracy, approximately 0.01°. However, this increased complexity results in a longer initial convergence period as the filter resolves the additional degrees of freedom in its state estimation. This suggests that in some circumstances there may be a trade-off in the dynamics-aware filter: increased convergence time for smaller steady-state error. This could potentially be addressed by using the initial measurements (such as sun direction and magnetic field) to estimate an initial orientation.

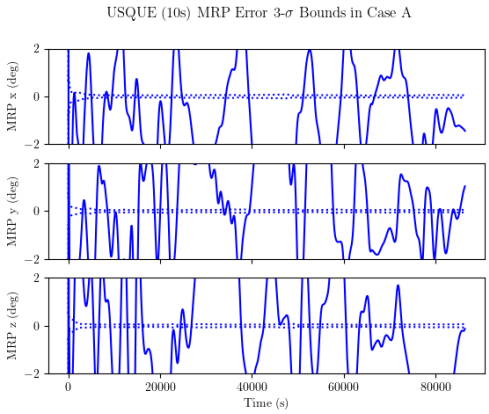
Further insight into convergence can be found in Figure 4.3, where the angular estimation errors (as Modified Rodrigues Parameters, or MRPs [18], [125]) are shown for each test with the 3σ error bounds on the estimation. Figures 4.3a and 4.3b, displaying the 1 and 10 second USQUE results, respectively, show that the attitude variance converges very quickly to a low number because the uncertainties from disturbances and actuator bias and noise are not accounted for. This over-certainty in angular estimation is likely a key component in the USQUE's lesser accuracy in this case, and we can see the MRP estimates struggling to converge. Comparing these to Figures 4.3c and 4.3d, which show the results for the dynamics-aware filter with 1 second and 10 second timesteps, we can see that the dynamics-aware filter does reach its 3σ bounds, taking roughly 5 hours. The dynamics-aware filter is also overconfident in its angular estimation initially but converges over time. Further tuning of uncertainty sources could result in slower-converging error bounds on angle, which could improve performance.

Figure 4.4 emphasizes the advantages of the dynamics-aware filter in estimating angular velocity. The filter achieves a steady-state angular velocity error of approximately $10^{-5\circ}$/s, compared to USQUE's $10^{-3\circ}$/s. This reduction in error is particularly important for trajectory planning and control, as precise angular velocity estimates lead to smoother maneuvers and reduced actuation effort. Additionally, the consistent performance of the dynamics-aware filter across both time steps (1 s and 10 s) demonstrates its robustness to different control and sensing update rates.

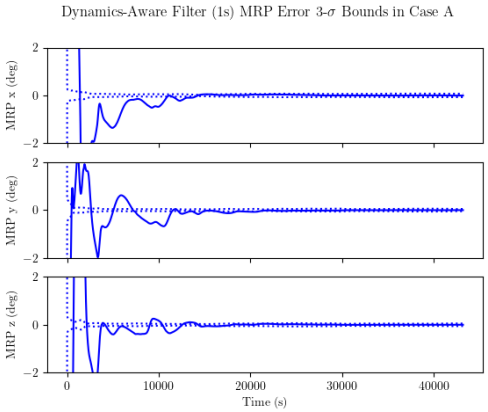
Overall, these results showcase the effectiveness of the dynamics-aware filter in significantly improving attitude estimation accuracy and convergence speed, even under challenging initial conditions with high attitude and bias errors. The ability to leverage control dynamics and disturbances enables the filter to outperform traditional methods like USQUE, making it particularly suitable for underactuated or resource-constrained systems such as small satellites. These findings underline the importance of incorporating detailed system modeling into attitude estimation frameworks to achieve enhanced performance and autonomy in satellite ADCS.



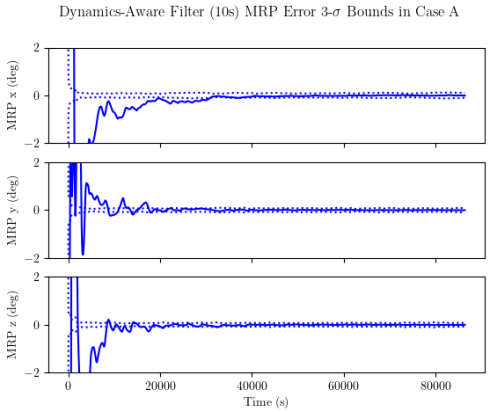
(a) USQUE (1s)



(b) USQUE (10s)



(c) Dynamics-Aware Filter (1s)



(d) Dynamics-Aware Filter (10s)

Figure 4.3: Case A (case details in Table 4.2) angular estimation error (as Modified Rodrigues Parameters (MRPs) [18], [125]) with 3σ error bounds for each test case used: 1 and 10 second time steps for the USQUE and this work's dynamics-aware filter. Both USQUE filters struggle to converge, while the Dynamics-Aware Filter converges to within their 3σ bounds in roughly 5 hours.

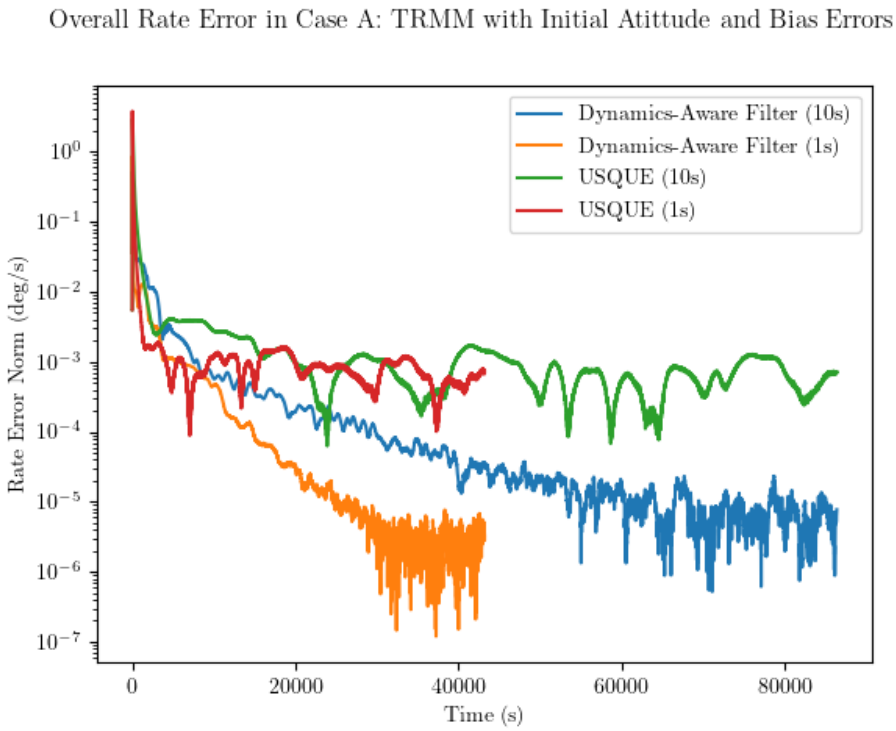


Figure 4.4: Case A (Section 4.4.2, TRMM with Initial Attitude and Bias Errors, case details in Table 4.2) angular velocity error, showing faster convergence and higher accuracy for the Dynamics-Aware Filter. The 1s time step simulations run for 12 hours instead of 24 (unlike the 10s time step simulations) due to memory constraints.

4.4.3 Case B: TRMM Satellite with Only Small Initial Bias Errors

The tests from Case A, in Section 4.4.2, are also repeated with a "close" start. This is to match the first example in [18] (TRMM with only small initial bias errors) and to show what happens over time, after the filters have converged. The satellite, orbit, and most parameters are the same, as described in Table 4.2. The difference is how accurate the estimator's initial estimates are. The changes are described in Table 4.3. In this case, the initial estimated angular velocity and attitude are exactly correct, the initial estimate of gyro biases is all zeros, and $P_{\text{att}} = (0.5°)$. For the dynamics-aware filter from this work, $P_\omega = (0.5°/\text{min})^2$.

Table 4.3: Satellite Properties and Test Parameters for Case B (only those that differ from Case A, Table 4.2)

Property	**Value/Description**
Estimated Gyroscope	Initial Bias: $[0, 0, 0]$ deg/hr Noise Std Dev: 0.31623 μrad/s$^{0.5}$ Bias Drift Rate: 3.1623×10^{-4} μrad/s$^{1.5}$
Estimated Initial State	Angular Velocity: $[0, \frac{1}{15}, 0]$ deg/s (not used by definition in USQUE, matches "real" value) Quaternion: $[0.63, -0.33, -0.63, 0.33]$ (from first case in [18], matches "real" value)
USQUE Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(0.5°)^2$ Initial $P_{\text{gyro bias}}$: $(0.2°/\text{hr})^2$ Integration covariances calculated as in [18]
Dynamics-Aware Filter Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(0.5°)^2$ Initial $P_{\text{gyro bias}}$: $(0.2°/\text{hr})^2$ Initial P_ω: $(0.5°/\text{min})^2$

The results for Case B demonstrate the stability and performance advantages of the dynamics-aware filter when initialized with small errors, compared to the baseline Unscented Quaternion Estimator (USQUE). Figure 4.5 illustrates the overall angular error over time, where both filters converge quickly due to the close initial conditions. However, the dynamics-aware filter achieves a significantly lower steady-state error of approximately 0.01°, while USQUE stabilizes at an error of 1° or greater. This reduction in error underscores the robustness of the dynamics-aware filter, particularly in maintaining higher accuracy over extended periods. This discrepancy stems from USQUE's limitations in handling sensor bias and disturbances even under favorable initial conditions.

Further insight into convergence can be found in Figure 4.6, where the angular estimation errors (as MRPs) are shown for each test with the 3σ error bounds on the estimation. Figures 4.6a and 4.6b, displaying the 1 and 10-second USQUE results respectively, show that the attitude variance converges very quickly to a low number because the uncertainties from disturbances and actuator bias and noise are not accounted for. This over-certainty

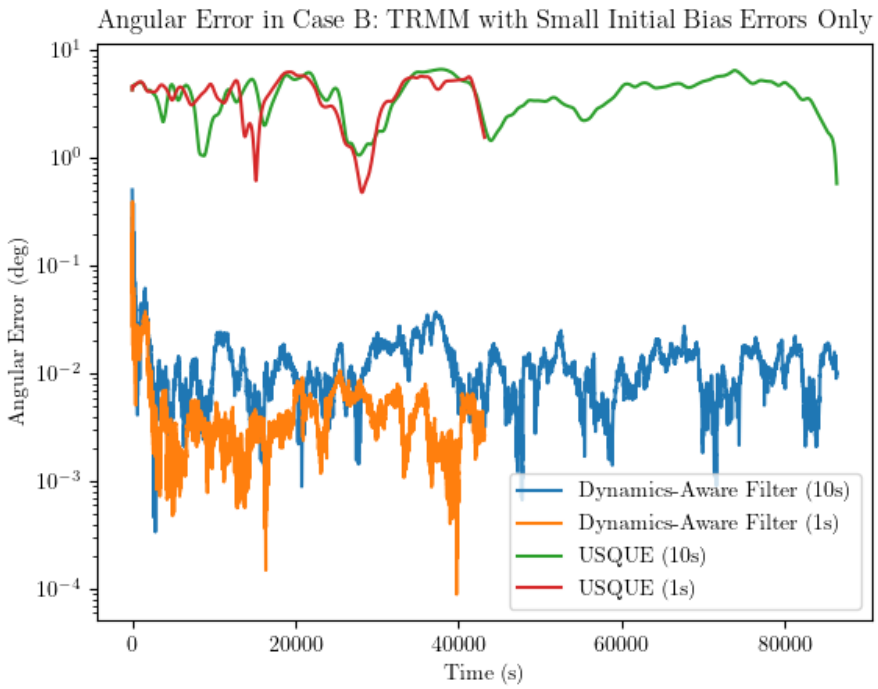
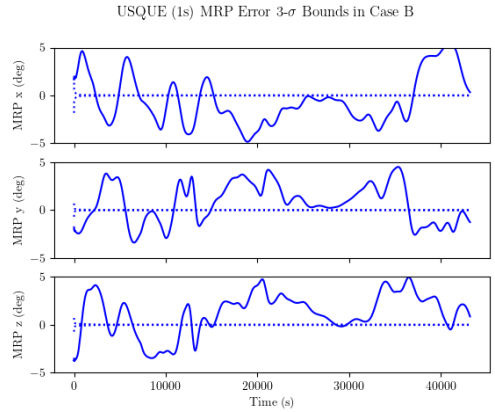
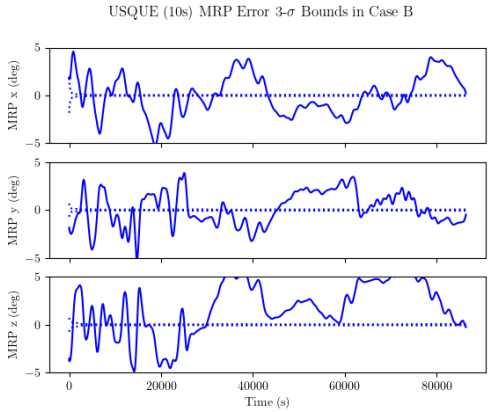


Figure 4.5: Angular estimation error in Case B (Section 4.4.3, TRMM with Only Initial Bias Errors, case details in Table 4.3), showing higher accuracy with the Dynamics-Aware Filter. With the "close" initial estimate, both filters converge very quickly. The 1s time step simulations run for 12 hours instead of 24 (unlike the 10s time step simulations) due to memory constraints.

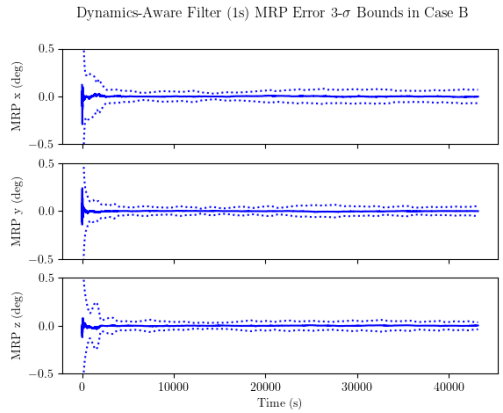
in angular estimation is likely a key component in USQUE's lesser accuracy in this case. Comparing these to Figures 4.6c and 4.6d, which show the results for the dynamics-aware filter with 1-second and 10-second timesteps, we can see that the dynamics-aware filter starts within its 3σ bounds and stays inside them over time. The dynamics-aware filter is also overconfident in its angular estimation initially but converges over time. Further tuning of uncertainty sources could result in slower-converging error bounds on angle, which could improve performance.



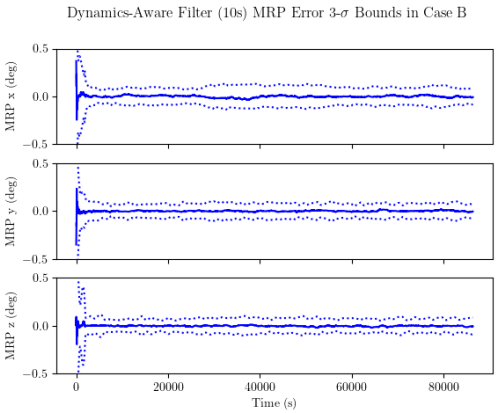
(a) USQUE (1s)



(b) USQUE (10s)



(c) Dynamics-Aware Filter (1s)



(d) Dynamics-Aware Filter (10s)

Figure 4.6: Case B (case details in Table 4.3) angular estimation error (as Modified Rodrigues Parameters (MRPs) [18], [125]) with 3σ error bounds for each test case used: 1 and 10-second time steps for the USQUE and this work's dynamics-aware filter. Both USQUE filters struggle to converge, while the Dynamics-Aware Filter converges to within their 3σ bounds in roughly 5 hours.

Figure 4.7 emphasizes the angular velocity estimation accuracy of the dynamics-aware filter. The filter achieves a steady-state angular velocity error of approximately 10^{-5}°/s, compared to USQUE's 10^{-3}°/s. This lower error ensures smoother tracking, better dynamics estimation, and reduced actuation effort, which is crucial for long-duration missions. Additionally, the dynamics-aware filter's consistent performance across both time steps (1 s and 10 s) demonstrates its reliability for various operational settings.

These results confirm that the dynamics-aware filter not only excels under challenging initial conditions, as shown in Case A, but also maintains its superiority with smaller initial errors. By leveraging system dynamics and disturbances, the filter achieves greater accuracy and stability, making it an ideal choice for high-performance attitude determination and control. Its ability to handle varying conditions and provide precise state estimates underscores its value for autonomous and resource-constrained satellite missions.

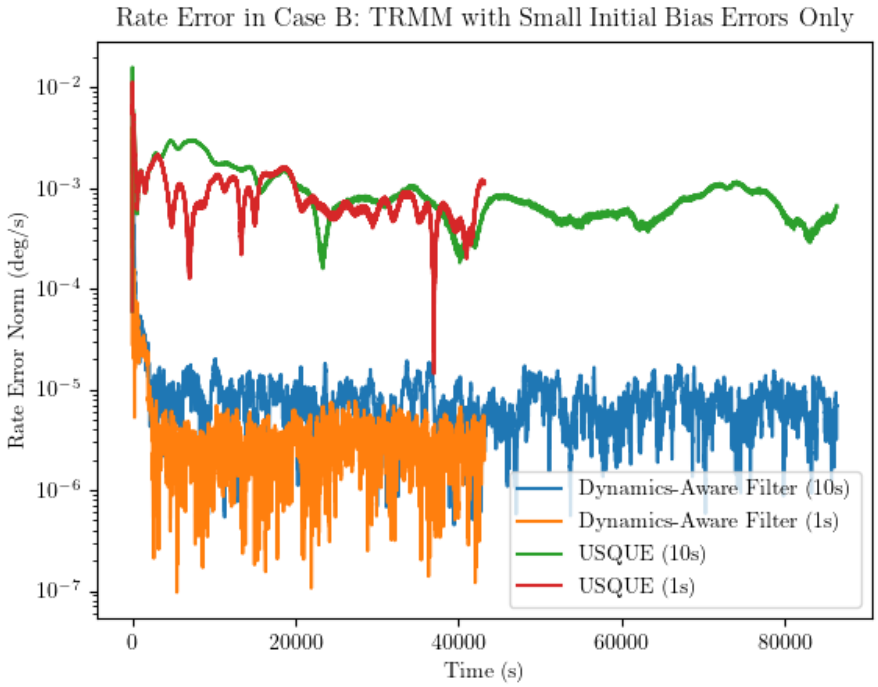


Figure 4.7: Case B (Section 4.4.3, TRMM with Only Initial Bias Errors, case details in Table 4.3) angular velocity error, showing higher accuracy for the Dynamics-Aware Filter. The 1s time step simulations run for 12 hours instead of 24 (unlike the 10s time step simulations) due to memory constraints.

4.4.4 Case C: CubeSat with Initial Bias and Attitude Errors

The dynamics-aware filter is perhaps most useful for nanosatellites where the sensors may be significantly worse than those in Sections 4.4.2 and 4.4.3. With such sensors, tracking the angular velocity and disturbances alongside the gyroscope bias increases the information available to the filter by incorporating knowledge of the dynamics and disturbances. Thus, other tests are run benchmarking this filter in a CubeSat test case. This satellite has significantly worse sensors, particularly MEMS gyros. It is also smaller and less stable inertially. Details can be found in Table 4.4.

Table 4.4: Satellite Properties and Test Parameters for Case C

Property	**Value/Description**
Mass	$m = 4$ kg
Inertia Matrix	$J = \begin{bmatrix} 0.0314 & 5.9 \times 10^{-5} & -0.0067 \\ 5.9 \times 10^{-5} & 0.0341 & -0.0001 \\ -0.0067 & -0.0001 & 0.01005 \end{bmatrix}$ kgm^2;
MTQ	Maximum Magnetic Moment: 1 Am^2 Initial Bias: $0.05 * \begin{bmatrix} \frac{1}{3\sqrt{2}} & \frac{1}{3\sqrt{2}} & \frac{4}{3\sqrt{2}} \end{bmatrix}$ Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ Bias Drift Rate: 0.000001 Am^2/s$^{0.5}$
Gyroscope[a]	Initial Bias: $\frac{0.1}{\sqrt{11}} * \begin{bmatrix} 1 & -1 & 3 \end{bmatrix}$ deg/s Noise Std Dev: 0.0004 deg/s$^{0.5}$ Bias Drift Rate: 0.03 deg/s$^{1.5}$
Magnetometer	Initial Bias: $[-9.948, -0.199, -0.995]$ nT Noise Std Dev[b] 300 nT·s$^{0.5}$ Bias Drift Rate: 1 nT/s$^{0.5}$
Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0.015, 0.027, -0.009]$ (unitless) Noise Std Dev: 0.0003 s$^{0.5}$ Bias Drift Rate: 0.000003 s$^{-0.5}$
Estimated MTQ	Maximum Magnetic Moment: 1 Am^2 (not used in USQUE by definition) Initial Bias: $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ (not used in USQUE by definition) Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ (not used in USQUE by definition) Bias Drift Rate: 0.000001 Am^2/s$^{0.5}$ (not used in USQUE by definition)
Estimated Gyroscope	Initial Bias: $[0, 0, 0]$ deg/hr Noise Std Dev: 0.0004 deg/s$^{0.5}$ Bias Drift Rate: 0.03 deg/s$^{1.5}$
Estimated Magnetometer	Initial Bias: $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ nT

Continued on next page

Table 4.4 – *Continued from previous page*

Property	Value/Description
	Noise Std Dev: 300 nT·$s^{0.5}$ Bias Drift Rate: 1 nT/$s^{0.5}$ (MTM bias ignored in USQUE)
Estimated Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0, 0, 0]$ (unitless) Noise Std Dev: 0.0003 $s^{0.5}$ Bias Drift Rate: 0.000003 $s^{-0.5}$ (Sun sensor bias ignored in USQUE)
Orbit Parameters	Approximately circular orbit Initial Position: 7000 km along ECI x-axis Inclination: 45° Initial Velocity: 8 km/s, perpendicular to orbital position J2 effects included
Update Period	1 second
Simulation Duration	3 hours
Control Algorithms	No Control for first 5 min B-dot (with numeric differentiation) applied from 5 min until 15 min; Gain: 10^6 Lovera Magnetic PD Algorithm applied after 2-hour mark: Gain (Proportional: 10, Velocity: 1000)
Quaternion Pointing Goals	0 min to 15 min: None (no control or B-dot) 15 min to 45 min: [1,0,0,0] 45 min to 90 min: [0,0,1,0] 90 min to 135 min: [0,1,0,0] 135 min to 180 min: [1,0,0,0]
Initial State	Angular Velocity: $[-0.00698, 0.00563, -0.01497]$ deg/s Quaternion: $[0.897, -0.391, 0.164, -0.128]$
Estimated Initial State	Angular Velocity: $[0, 0, 0]$ deg/s (not used by definition in USQUE) Quaternion: $[1, 0, 0, 0]$
USQUE Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(10 \text{ rad})^2$ Initial $P_{\text{gyro bias}}$: $(0.2\ ^\circ/\text{s})^2$ Integration covariances calculated as in [18]
Dynamics-Aware Filter Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(10 \text{ rad})^2$ Initial $P_{\text{gyro bias}}$: $(0.2\ ^\circ/\text{s})^2$ Initial P_{ω}: $(1^\circ/\text{s})^2$ Initial $P_{\text{mtq bias}}$: $(0.1 \text{ Am}^2)^2$ Initial $P_{\text{mtm bias}}$: $(100 \text{ nT})^2$

Continued on next page

Table 4.4 – *Continued from previous page*

Property	**Value/Description**
	Initial $P_{\text{sun bias}}$: $(0.03)^2$ Integration covariances on biases match bias drift rates times Δt Attitude integration covariance: $10^{-12}(\Delta t)^2$ rad^2 (based on experimentation) Angular velocity integration covariance: $10^{-17}(\Delta t)$ rad^2/s^2 (based on experimentation)
SRUKF Parameters	$\alpha = 1$ $\beta = 2$ $\kappa = 3 - (L + 3)$ (with the additional 3 added for control noise and L is the state length)

[a] Similar to MEMS gyros [27][25].
[b] Based on [27].

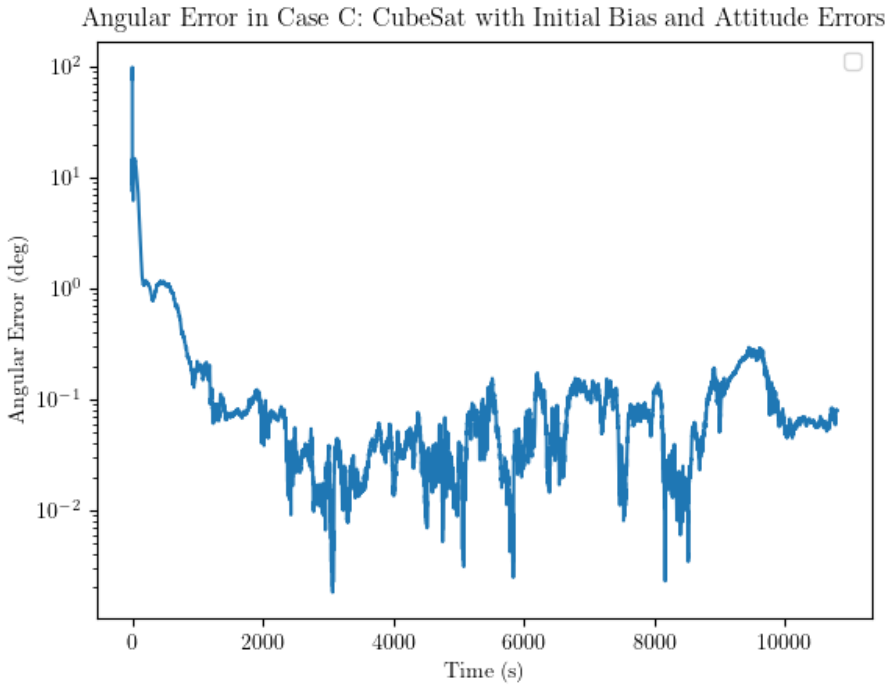


Figure 4.8: Case C (Section 4.4.4, CubeSat with Initial Bias and Attitude Errors, case details in Table 4.4) showing the estimator orientation estimate converging to within 1 degree (usually close to 0.1 degree) on a CubeSat with simple sensors.

The results for Case C demonstrate the dynamics-aware filter's superior performance in handling noisy, low-quality sensors and larger initial errors, conditions typical for small satellites such as CubeSats. Figure 4.8 shows the overall angular error over time, where the dynamics-aware filter converges to an error of approximately 0.1°, despite significant sensor

noise and actuator bias. Standard methods typically achieve a performance of roughly 1° (or greater) accuracy under similar conditions (still quite good for a CubeSat), an order of magnitude higher than the results shown here [27], [34], [120], [126]. Figure 4.8 also shows that the angular error goes within 1° almost immediately–in less than 10 minutes. In contrast, traditional methods often require longer time scales of at least 20 minutes to reach such accuracy [126]. This rapid convergence and high accuracy are noteworthy for CubeSat-class systems, where computational resources are limited, and sensors and actuators may be quite poor.

Figure 4.9 highlights the angular velocity estimation performance. The dynamics-aware filter achieves steady-state errors as low as 10^{-3} °/s, even with poor-quality gyroscope readings. Accurate angular velocity estimation ensures smoother control maneuvers and minimizes actuator usage, which is critical for energy-constrained CubeSats.

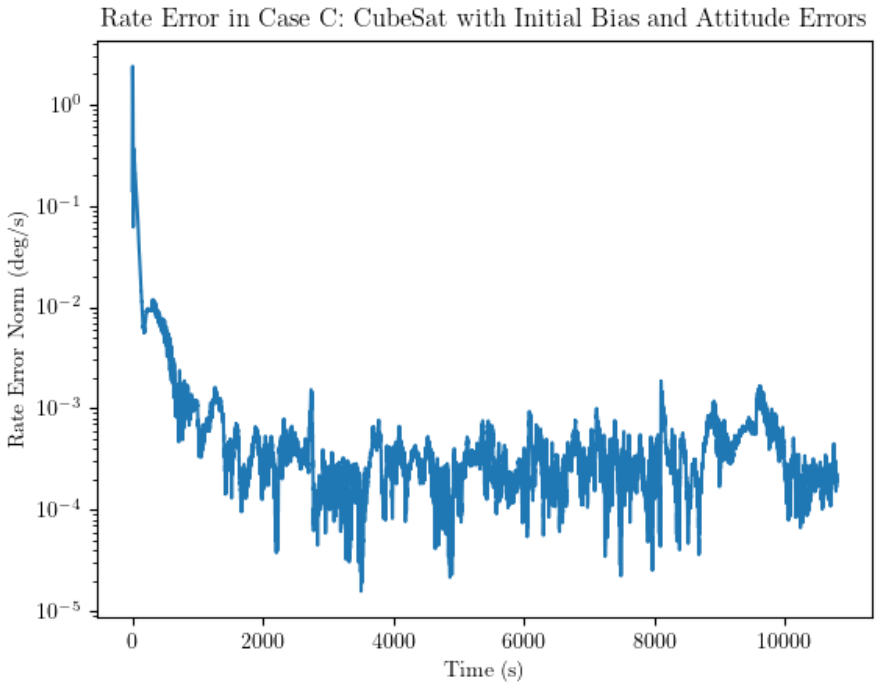
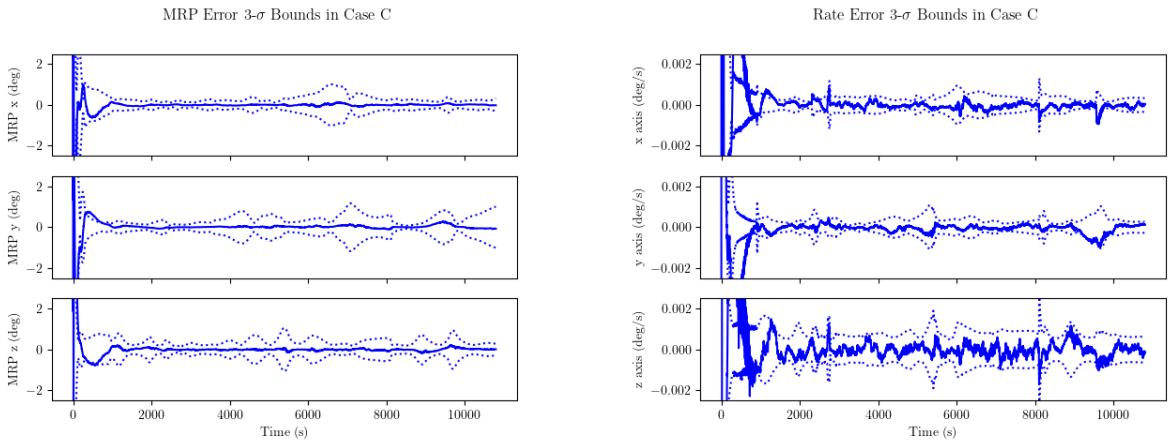


Figure 4.9: Case C (Section 4.4.4, CubeSat with Initial Bias and Attitude Errors, case details in Table 4.4) shows the norm of the estimator's angular velocity error, running on a CubeSat with simple sensors. It remains within 0.001 degrees/sec.

The convergence rate of the dynamics-aware filter further reinforces its utility. Traditional filters applied to CubeSats, such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF), often struggle to achieve sub-degree accuracy in the presence of sensor noise, with reported steady-state errors of 0.5° to 2.5° [27], [34], [120], [126]. By comparison, the dynamics-aware filter achieves 0.1° accuracy in a fraction of the time, while remaining robust to disturbances and sensor biases. Figure 4.10 shows the estimation error for both MRPs (Figure 4.10a) and angular velocity (Figure 4.10b) staying within their respective 3-sigma values.



(a) Angular Error (as MRP)

(b) Angular Velocity Error

Figure 4.10: Angular and angular velocity estimation error from Case C (see details in Table 4.4). The left plot shows angular error (as Modified Rodrigues Parameters (MRP)), and the right plot shows angular velocity error.

Overall, these results demonstrate that the dynamics-aware filter enables small satellites to achieve high-precision attitude determination with low-cost sensors and underactuated actuators. This capability significantly expands the operational potential of CubeSats, enabling precise pointing goals without requiring high-end star trackers or gyroscopes. The rapid convergence rate and low steady-state errors achieved by the filter make it a promising solution for autonomous and resource-constrained systems, further emphasizing its applicability for future small satellite missions.

4.4.5 Case D: CubeSat with Only Small Initial Bias Errors

The test from Case C in Section 4.4.4 (a 3U CubeSat with lower-quality sensors) is repeated with a "close" start to demonstrate performance over time after the filter has already converged. In this simulation, the initial estimated angular velocity and attitude are exactly correct, $P_{\text{att}} = (0.5°)$, and $P_{\omega} = (0.5°/\text{min})^2$. This is summarized in Table 4.5.

Table 4.5: Satellite Properties and Test Parameters for Case D (only those that differ from Case C, Table 4.4)

Property	**Value/Description**
Estimated Initial State	Angular Velocity: $[-0.00698, 0.00563, -0.01497]$ deg/s (matching "real" value, not used in USQUE) Quaternion: $[0.897, -0.391, 0.164, -0.128]$ (matching "real" value)
USQUE Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(1 \text{ rad})^2$

Continued on next page

Table 4.5 – *Continued from previous page*

Property	Value/Description
Dynamics-Aware Filter Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(1 \text{ rad})^2$ Initial P_{ω}: $(0.5°/\text{min})^2$

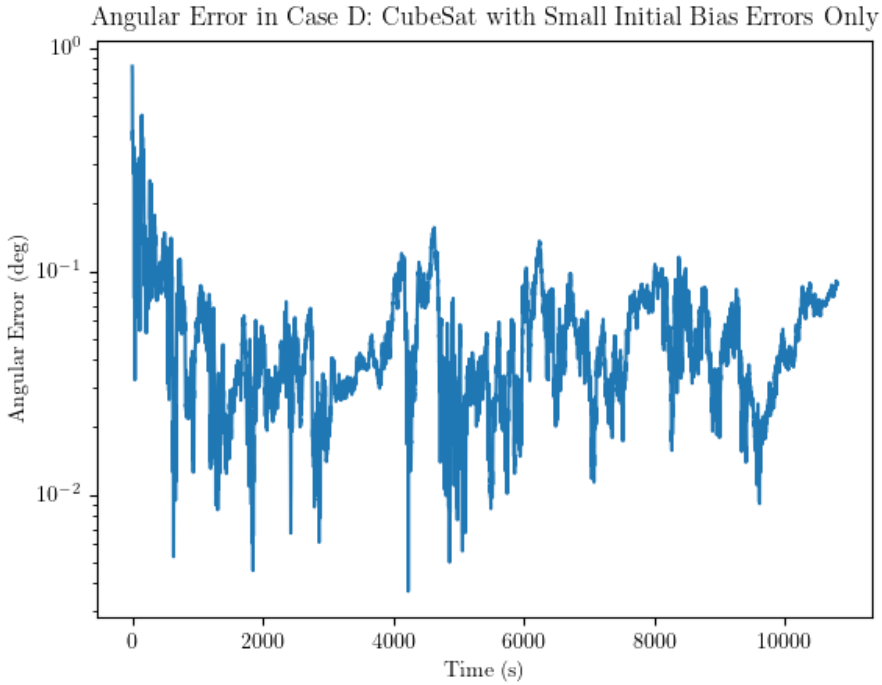
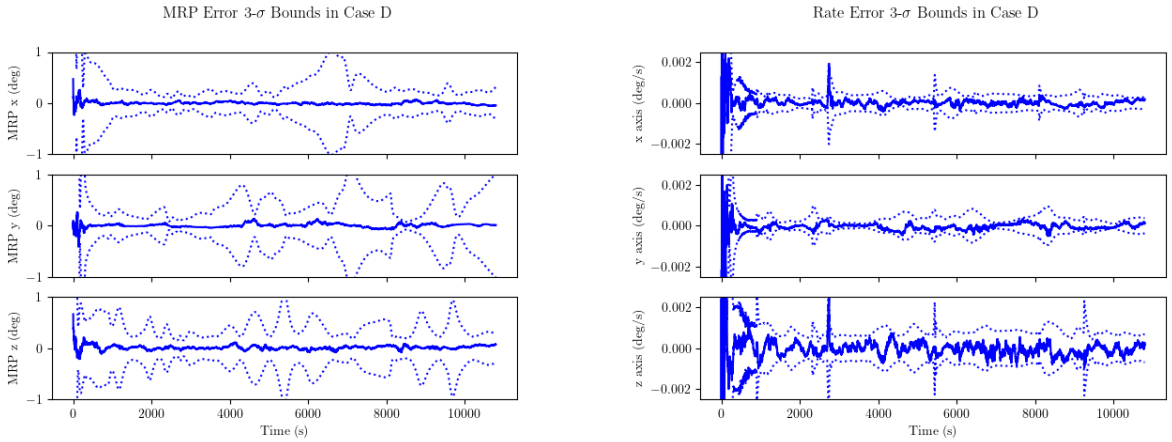


Figure 4.11: Case D (Section 4.4.5, CubeSat with Only Initial Bias Errors, case details in Table 4.5) shows the estimator orientation estimate staying at roughly 0.1 degrees on a CubeSat with simple sensors.

The results for Case D demonstrate the robustness of the dynamics-aware filter when initialized with smaller errors, typical of scenarios where an attitude determination system has already converged or the spacecraft is subject to only minor initial disturbances. Figure 4.11 shows the overall angular error over time, where the dynamics-aware filter quickly stabilizes at an error of approximately 0.05°-0.1°. This steady-state accuracy is remarkable for CubeSat-class missions, where standard filters often stabilize at errors between 0.5° and 2.5° [27], [34], [120], [126].

Figure 4.13 highlights the angular velocity estimation performance. The dynamics-aware filter achieves a steady-state error of approximately 5×10^{-4} °/s, a high level of accuracy for a CubeSat. This level of precision in angular velocity estimation ensures smoother control maneuvers, reduces the need for frequent corrections, and minimizes actuator usage, which is particularly important for energy-constrained small satellites.

The superior performance of the dynamics-aware filter in Case D highlights its ability to maintain high accuracy and rapid convergence even under favorable initial conditions. This is critical for CubeSat-class missions, where disturbances such as drag and magnetic torques can quickly degrade performance without robust estimation. The filter's ability to stabilize at such low steady-state errors provides significant operational advantages, enabling precise pointing goals and autonomous operation with minimal ground intervention. Figure 4.12 shows the estimation error for both MRPs (Figure 4.12a) and angular velocity (Figure 4.12b) staying well within their respective 3-sigma values.



(a) Angular Error (as MRP)

(b) Angular Velocity Error

Figure 4.12: Angular and angular velocity estimation error from Case D (See details in Table 4.5). The left plot shows angular error (as Modified Rodrigues Parameters (MRP)), and the right plot shows angular velocity error.

Overall, these results reinforce the utility of the dynamics-aware filter for resource-constrained systems like CubeSats. By leveraging system dynamics, accounting for disturbances, and effectively managing sensor biases, the filter outperforms traditional approaches in both convergence speed and steady-state accuracy. This makes it an ideal choice for high-precision, autonomous attitude control in next-generation small satellite missions.

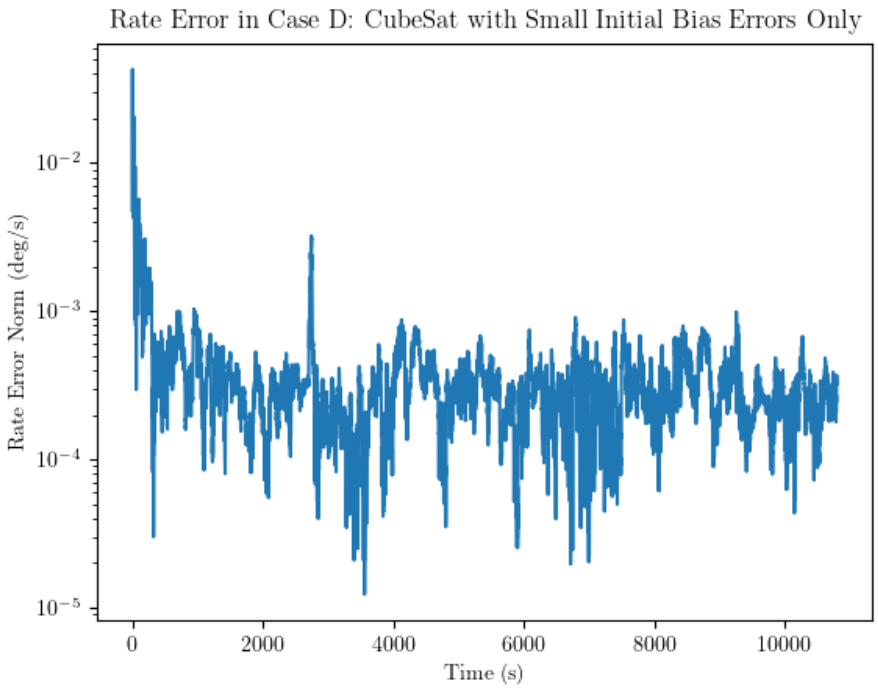


Figure 4.13: Case D (Section 4.4.5, CubeSat with Only Initial Bias Errors, case details in Table 4.5) showing the norm of the estimator's angular velocity error, running on a CubeSat with simple sensors. It remains within 0.001 degrees/sec.

4.4.6 Case E: Effect of Ignoring Actuator Bias or Disturbances

The dynamics-aware estimator in Case C incorporates system dynamics to track the angular velocity and orientation (as a quaternion) but also models actuator bias and disturbances. To test whether these are important, the same tests as in Case C are run (with the same setup, see Table 4.4), but excluding actuator bias or disturbances from the estimator (but not the "real" satellite).

The results for this test demonstrate the importance of accounting for actuator bias and disturbances in the dynamics-aware filter. Figure 4.14 compares the angular estimation error for two scenarios: one where actuator bias is ignored and one where it is modeled within the filter. When actuator bias is included in the estimator, the error remains at approximately 0.1° throughout the simulation. However, when actuator bias is ignored, the estimation error diverges rapidly, staying almost 180° off. This catastrophic result highlights the critical need to account for actuator bias in systems where even small control errors can accumulate over time.

Similarly, Figure 4.15 illustrates the effect of ignoring disturbances within the estimator. In the scenario where disturbances are accounted for, the angular error remains around 0.1°, consistent with prior results from Case C. In contrast, ignoring disturbances leads to a significant degradation in performance, with errors stabilizing around 1°-5°. While not as extreme as the actuator bias case, this result emphasizes the importance of modeling disturbances, especially for small satellites where external torques like aerodynamic drag and solar radiation pressure can introduce significant perturbations.

Together, these results clearly demonstrate that both actuator bias and disturbances must be incorporated into the estimation process to maintain high accuracy. The dynamics-aware filter achieves superior performance precisely because it models these effects, allowing it to maintain low steady-state errors and avoid divergence. Neglecting these factors, as shown in Case E, leads to substantial inaccuracies, undermining the filter's ability to provide reliable state estimation.

Overall, Case E validates the value of the dynamics-aware filter and highlights its sensitivity to model fidelity. By accounting for actuator bias and disturbances, the filter ensures accurate and stable attitude estimation, making it well-suited for resource-constrained small satellites operating in dynamic environments.

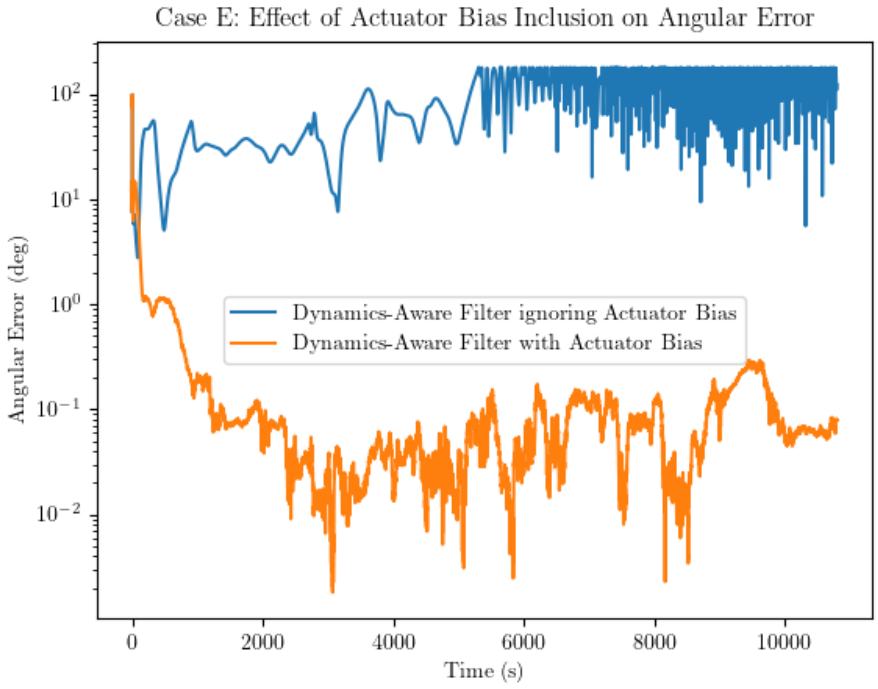


Figure 4.14: Case E (Section 4.4.6, case details identical to those in Table 4.4 without actuator bias included in estimator) compares the dynamics-aware filter's performance when actuator bias (starting value of $\beta_{\text{MTQ}} = 0.05 * \left[\frac{1}{3\sqrt{2}} \quad \frac{1}{3\sqrt{2}} \quad \frac{4}{3\sqrt{2}}\right]$, drift rate of $\eta_{\text{b,mtq}} = 0.000001$ $\text{Am}^2/\text{s}^{0.5}$) is ignored by the estimator or considered. Including the actuator bias results in an accuracy of around 0.1 degrees, while excluding it results in being 180 degrees off.

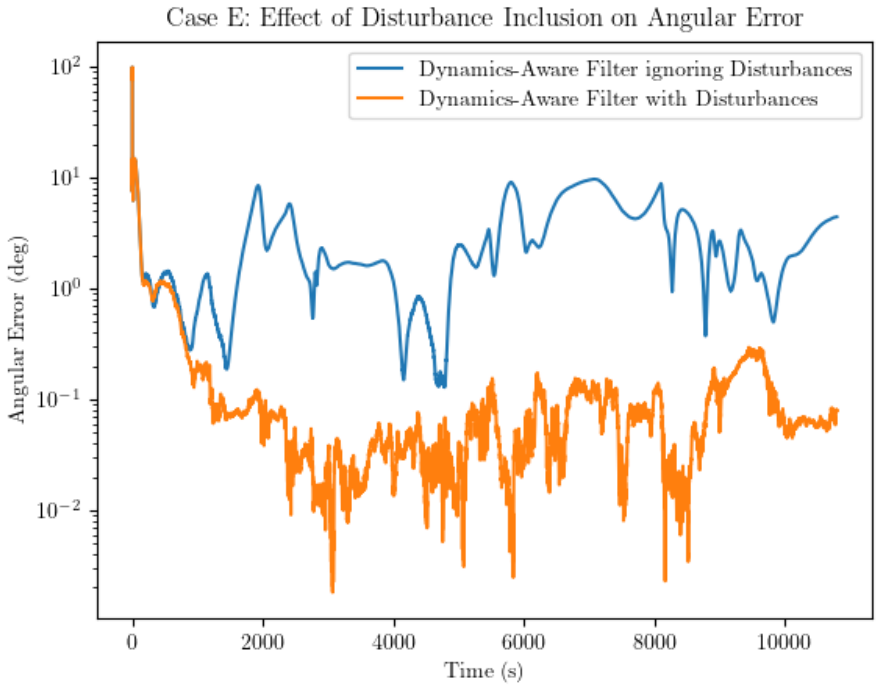


Figure 4.15: Case E (Section 4.4.6, case details identical to those in Table 4.4 without disturbances included in estimator) compares the dynamics-aware filter's performance when disturbances are ignored by the estimator or considered. Including the disturbances results in an accuracy of around 0.1 degrees, while excluding them results in an estimation error of around 1 degree.

4.4.7 Case F: CubeSat with Propulsion

To understand how the system deals with disturbances and to test the initial use case (see Section 2.4), a repeat of Case C (Section 4.4.4) is run, with the addition of a disturbing propulsion torque with an initial value of $\left[\frac{-3}{\sqrt{74}} \quad \frac{-8}{\sqrt{74}} \quad \frac{1}{\sqrt{74}}\right]$ μNm, a capped maximum value of 10 μNm, and a drift rate of $\eta_{\text{dipole}} = 0.01$ $\mu\text{Nm/s}^{0.5}$.

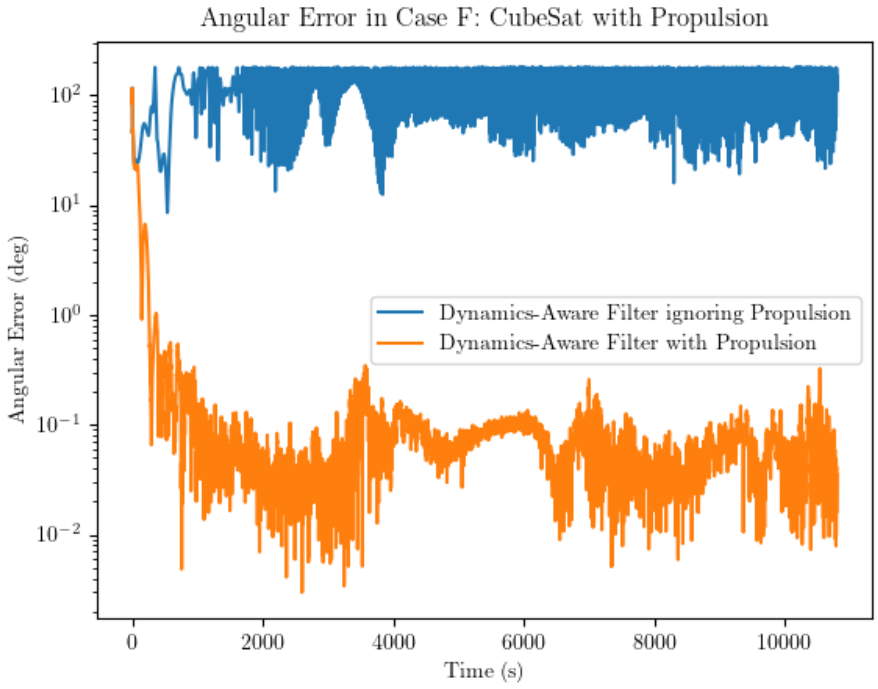


Figure 4.16: Case F (Section 4.4.7, case details identical to those in Table 4.4 with disturbing propulsion torque added throughout) compares the dynamics-aware filter's performance when a propulsion disturbance is ignored by the estimator or considered. Including the propulsion results in an accuracy of around 0.1 degrees, while excluding it results in an estimation error of around 180 degrees.

The results for this test highlight the dynamics-aware filter's capability to estimate and track propulsion disturbances, even with noisy and low-quality CubeSat sensors. Figure 4.16 illustrates the angular error over time when the propulsion disturbance is either ignored or included in the estimation process. Similar to removing bias in Case E (Section 4.4.6), when the disturbance is ignored, the error diverges to 180°, demonstrating a complete failure of the estimator to converge. In contrast, when the propulsion torque is modeled and included in the estimation, the filter maintains an angular error around 0.1°, consistent with prior results. This underscores the critical importance of including propulsion effects within the estimator to maintain accurate attitude determination.

Figure 4.17 further demonstrates the system's ability to estimate the propulsion disturbance. The estimated torque (dotted lines) converges rapidly and closely tracks the true disturbance (solid lines) over time, despite the inherent noise in the CubeSat sensors. While

the estimate is not perfect, it achieves a tracking accuracy on the order of 5×10^{-7} Nm, which is an order of magnitude smaller than the disturbance magnitude itself. This result is particularly noteworthy given the low fidelity of the sensors used in the simulation.

The ability to accurately estimate and track propulsion disturbances has significant implications for CubeSat attitude control. By incorporating the disturbance estimates into the dynamics-aware filter, the system can provide valuable insight into the propulsion effects acting on the spacecraft. This information can be leveraged to counteract the disturbances directly or include them in the control design, as demonstrated later in Chapter 6, and this applies directly to the motivational case in Section 2.4. The inclusion of such disturbances enables improved robustness and accuracy for small satellites operating in dynamic environments, where propulsion effects can otherwise cause significant pointing errors.

Overall, Case F demonstrates the effectiveness of the dynamics-aware filter in managing propulsion disturbances using simple CubeSat sensors. The filter's ability to accurately estimate and track these disturbances enables precise state estimation and lays the groundwork for more advanced disturbance-aware control strategies. This capability is essential for enhancing the performance and autonomy of resource-constrained small satellites.

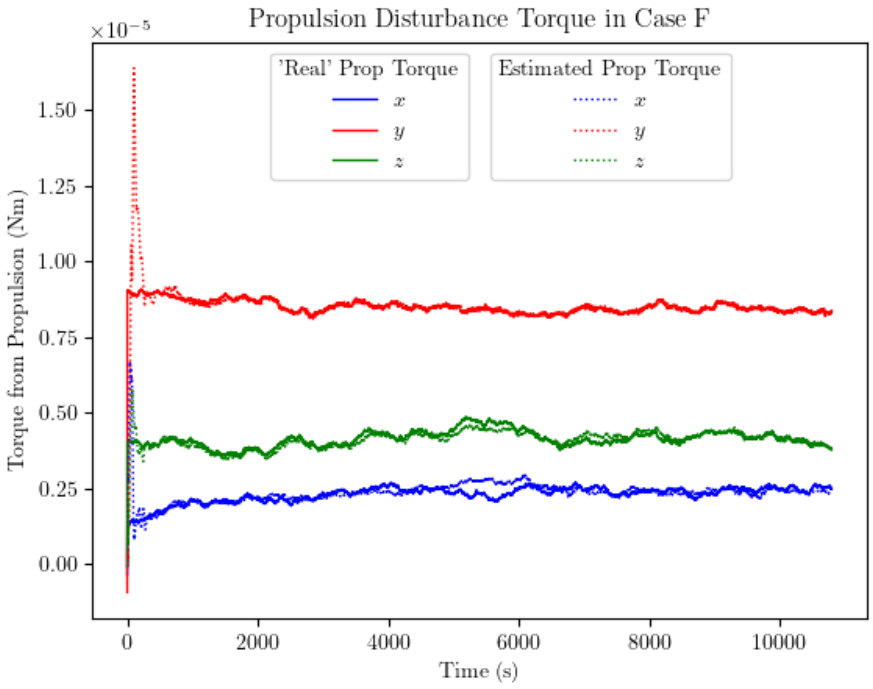


Figure 4.17: Case F (Section 4.4.7, case details identical to those in Table 4.4 with disturbing propulsion torque added throughout) compares the dynamics-aware filter's estimate of a propulsion disturbance against the "real" value. The system estimate quickly converges and generally tracks the torque to roughly $5{\times}10^{-7}$ Nm, an order of magnitude less than the disturbance magnitude.

4.4.8 Case G: Many Variables

Lastly, to demonstrate the ability of the estimator to follow a variety of important variables–biases and disturbances among them–a new test case is run. The test involves a 6U CubeSat sensing with mid-level gyroscopes, magnetometers, and sun sensors; actuating with both reaction wheels and magnetorquers; and experiencing disturbances including both a propulsion torque and residual dipole. Many values must be tracked–the dipole, the propulsion, sensor biases, and reaction wheel momenta, in addition to the orientation and angular velocity. Details of the test parameters can be found in Table 4.6.

Table 4.6: Satellite Properties and Test Parameters for Case G

Property	**Value/Description**
Mass[a]	$m = 10.165$ kg
Inertia Matrix[a]	$J = \text{diag}\left(\begin{bmatrix}0.0969 & 0.1235 & 0.1918\end{bmatrix}\right)$ kg·m^2
MTQ	Three-axis, identical on each Maximum Magnetic Moment: 5 Am^2 Initial Bias:[b] $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ Bias Drift Rate:[b] 0 Am^2/s$^{0.5}$
RW	Three-axis, identical on each Maximum Torque: 0.005 Nm Control Noise Std Dev: 0.00001 Nm s$^{0.5}$ Wheel Inertia (about spin axis): 0.0014 kg m^2 Maximum Momentum: 0.015 Nms Momentum Self-Measurement Noise: 10^{-12} Nms
Gyroscope	Initial Bias:[c] $\frac{0.1}{\sqrt{11}} * \begin{bmatrix}1 & -1 & 3\end{bmatrix}$ deg/s Noise Std Dev:[c] 0.0004 deg/s$^{0.5}$ Bias Drift Rate:[c] 0.03 deg/s$^{1.5}$
Magnetometer	Initial Bias: $[-9.948, -0.199, -0.995]$ nT Noise Std Dev:[d] 300 nT·s$^{0.5}$ Bias Drift Rate: 1 nT/s$^{0.5}$
Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0.0015, 0.0027, -0.009]$ (unitless) Noise Std Dev: 0.0003 s$^{0.5}$ Bias Drift Rate: 0.000003 s$^{-0.5}$
Estimated MTQ	Three-axis, identical on each Maximum Magnetic Moment: 5 Am^2 Initial Bias:[b] $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ Bias Drift Rate:[b] 0 Am^2/s$^{0.5}$
Estimated RW	Three-axis, identical on each Maximum Torque: 0.005 Nm Control Noise Std Dev: 0.00001 Nm s$^{0.5}$

Continued on next page

Table 4.6 – *Continued from previous page*

Property	**Value/Description**
	Wheel Inertia (about spin axis): 0.0014 kg m^2 Maximum Momentum: 0.015 Nms Momentum Self-Measurement Noise: 10^{-12} Nms
Estimated Gyroscope	Initial Bias: $[0, 0, 0]$ deg/hr Noise Std Dev: 0.0004 deg/$s^{0.5}$ Bias Drift Rate[f]: 0.045 deg/$s^{1.5}$
Estimated Magnetometer	Initial Bias: $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ nT Noise Std Dev: 300 nT·$s^{0.5}$ Bias Drift Rate[f]: 1.5 nT/$s^{0.5}$ (MTM bias ignored in USQUE)
Estimated Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0, 0, 0]$ (unitless) Noise Std Dev: 0.0003 $s^{0.5}$ Bias Drift Rate[f]: 0.0000045 $s^{-0.5}$ (Sun sensor bias ignored in USQUE)
Propulsion Disturbance	Initial Value: $\frac{50}{\sqrt{74}}[-3, -80, 10]$ μNm Maximum Magnitude: 100 μNm Drift Rate: 0.1 μNm/$s^{-0.5}$ Applied during 400s-1500s and 2800s-4500s.
Residual Dipole	Initial Value: $[0.05, 0.0001, 0.2]$ Am^2 Maximum Magnitude: 0.5 Am^2 Drift Rate: 0.0001 $Am^2/s^{-0.5}$
Estimated Propulsion Disturbance	Initial Value: $[0, 0, 0]$ μNm Maximum Magnitude: 100 μNm Drift Rate[f]: 0.2 μNm/$s^{-0.5}$ Applied during 400s-1500s and 2800s-4500s.
Estimated Residual Dipole	Initial Value: $[0, 0, 0]$ Am^2 Maximum Magnitude: 0.5 Am^2 Drift Rate[f]: 0.0002 $Am^2/s^{-0.5}$
Other Disturbances	Gravity gradient, drag, and solar radiation pressure are also included, in both the estimator and the "real" satellite.[h]
Orbit Parameters	Approximately circular orbit Initial Position: 7000 km along ECI x-axis Inclination: 45° Initial Velocity: 8 km/s, perpendicular to orbital position J2 effects included
Update Period	1 second
Simulation Duration	4500 seconds

Continued on next page

Table 4.6 – *Continued from previous page*

Property	Value/Description
Control Algorithms	No Control for first 5 seconds. RW/MTQ-compatible B-dot variant[e]applied from 5s until 20s MTQ/RW-balanced PD Algorithm[g]applied after 20s mark
Attitude Goals[i]	0s to 20s: None (no control or B-dot) 20s to 100s: $\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0\end{bmatrix}$ 100s to 200s: $\begin{bmatrix}0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}$ 200s to 1500s: $\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0\end{bmatrix}$ 1500s to 2500s: $\frac{2}{3}\begin{bmatrix}-1 & 1 & -1\end{bmatrix}$ 2500s on: $\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0\end{bmatrix}$
Initial State	Angular Velocity: $[0.02, 0.02, 0.002]$ deg/s Quaternion: $[0.153, 0.685, 0.695, 0.153]$ RW-stored Angular Momentum: $-5 * 10^{-5} * [1, 1, 1]$ Nms
Estimated Initial State	Angular Velocity: $[0.02, 0.02, 0.002]$ deg/s Quaternion: $[0.153, 0.685, 0.695, 0.153]$ RW-stored Angular Momentum: $-5 * 10^{-5} * [1, 1, 1]$ Nms
Dynamics-Aware Filter Covariances	No cross terms on initial state estimate covariance Initial P_{att}: $(0.1 \text{ deg})^2$ Initial $P_{\text{gyro bias}}$: $(0.2\ ^\circ/\text{s})^2$ Initial P_ω: $(0.1^\circ/\text{s})^2$ Initial P_h: $(10^{-6} \text{ Nms})^2$ Initial $P_{\text{mtm bias}}$: $(3000 \text{ nT})^2$ Initial $P_{\text{sun bias}}$: $(0.06)^2$ Initial P_{dipole}: $(0.15 \text{ Am}^2)^2$ Initial $P_{\text{prop dist}}$: $(10^{-4} \text{ Nm})^2$ Integration covariances on biases match bias drift rates times Δt Attitude integration covariance: $10^{-12}(\Delta t)^2$ rad^2 (based on experimentation) Angular velocity integration covariance: $10^{-17}(\Delta t)$ rad^2/s^2 (based on experimentation)
SRUKF Parameters	$\alpha = 1$ $\beta = 2$ $\kappa = 3 - (L + 3)$ (with the additional 3 added for control noise and L is the state length)

[a] Based on the physical properties of ASTERIA [1].
[b] No bias used for MTQs because MTQ bias (when modeled linearly) is dynamically

identical to dipole, which is already being tracked and estimated. One can think of this as estimating the sum of those two variables.

[c] Similar to MEMS gyros [27][25].

[d] Based on [27].

[e] Basically velocity-countering using the estimated angular velocity, balanced between MTQ and RW, similar to [127].

[f] Differs from real value, simulating imperfect knowledge.

[g] Based on [127].

[h] These disturbances are modeled with exact information as described in Section 4.3.4. However, there may be discrepancies between the disturbances on the "real" and virtual satellites from orientation estimation error. For example, if the estimated orientation is such that the major principal axis of the satellite is pointing nadir, it will estimate zero gravity gradient torque, while the "real" satellite may experience a torque corresponding to its actual orientation.

[i] Represented here as an MRP relative to an orbital frame (from [128]) with the z-axis pointing zenith and the x-axis pointing toward the positive orbit normal.

The results for this case illustrate the effectiveness of the dynamics-aware filter in tracking a variety of disparate variables simultaneously, even under conditions of significant noise and disturbances. Figure 4.18 shows the angular estimation accuracy, where the filter stays mostly beneath an error of approximately 0.1° despite the added complexity of tracking many parameters. This level of accuracy is consistent across the simulation, showcasing the filter's robustness in managing a diverse set of variables.

Figure 4.20 demonstrates the filter's ability to estimate stored angular momentum accurately, with errors consistently below 10^{-4} Nms. Similarly, the propulsion torque is tracked effectively, as shown in Figure 4.21, with the estimated values closely following the true disturbances despite sensor noise and drift. Even when the propulsion is turned off (but the value is allowed to evolve, despite not affecting the dynamics[4]) and back on again, the disturbance quickly re-converges and tracks the new value. The residual dipole tracking, illustrated in Figure 4.23, highlights the filter's capacity to isolate and estimate magnetic effects accurately, achieving an estimation precision of approximately 10^{-2} Am^2.

The bias plots shown in Figure 4.24 highlight the filter's ability to estimate and track various sensor biases effectively. Across all bias variables, including gyroscope bias, magnetometer bias, and sun sensor bias, the dynamics-aware filter quickly converges to accurate estimates, with the steady-state errors that remain mostly orders of magnitude smaller than the signals. This capability is critical for mitigating the long-term effects of biased measurements and ensuring consistent accuracy in state estimation.

The 3σ error bound plots shown in Figure 4.25 compare the filter's estimation error to its variance across several different state variables, including attitude (represented as MRP in Figure 4.25a), angular velocity (Figure 4.25b), stored momentum (Figure 4.25f), and various sensor biases (Figures 4.25c-4.25e). In all cases, the variables generally respect the error bounds, showing tight convergence and accurate understanding of the uncertainty environment. These results validate the filter's ability to handle complex systems with

[4]This may not be physically sensible, but it is implemented as the sort of worst-case situation for how an unknown variable might behave during a time period for which it cannot be sensed.

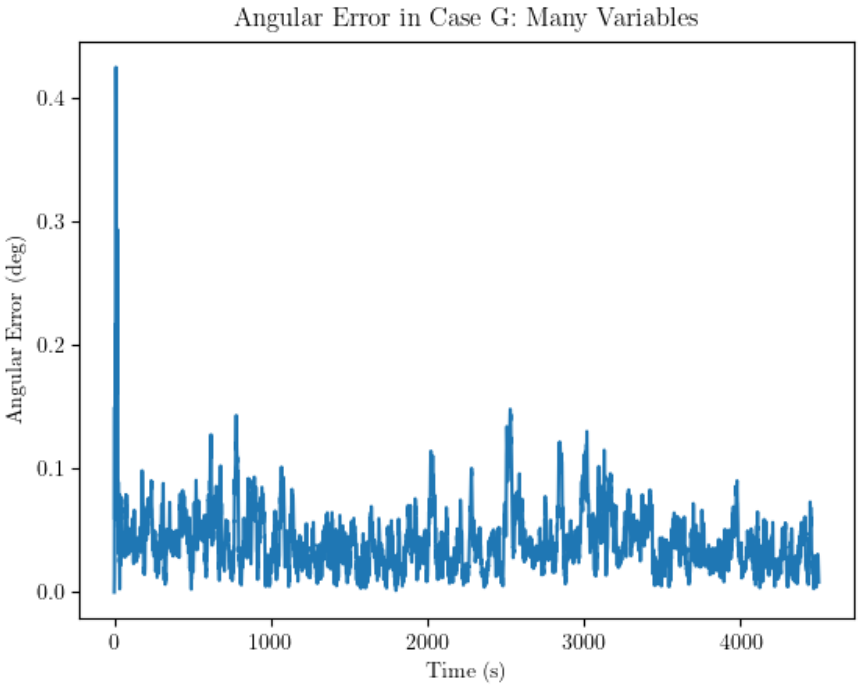


Figure 4.18: Case G (Section 4.4.8, case details in Table 4.6) demonstrates angular estimation accuracy under disturbances and many variables needing to be tracked.

multiple time-varying parameters, such as biases, disturbances, and actuation effects. By accurately estimating these variables, the dynamics-aware filter not only provides reliable state estimates but also enables further optimization in system design and control. The precise tracking of propulsion disturbances, angular momentum, and magnetic dipoles has direct implications for CubeSat mission planning and operational autonomy. For instance, accurate disturbance estimation can be used to improve control strategies or refine actuator usage, enhancing overall system efficiency.

In summary, Case G demonstrates that the dynamics-aware filter can successfully track a wide range of disparate variables simultaneously while maintaining high estimation accuracy. This capability is particularly critical for small satellite systems, where multiple disturbances and operational constraints must be managed effectively to achieve mission goals.

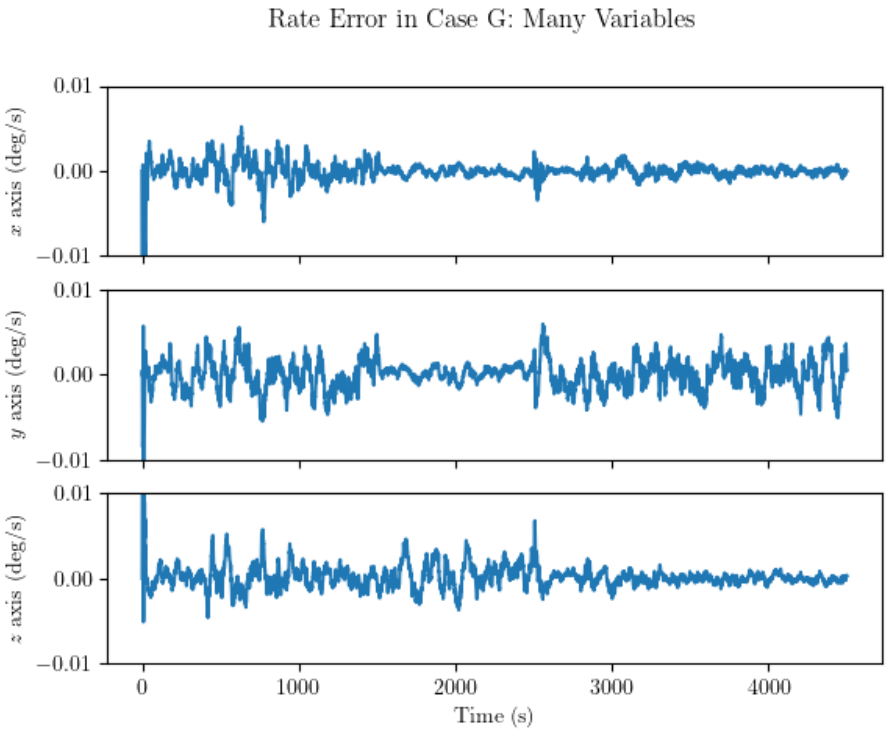


Figure 4.19: Case G (Section 4.4.8, case details in Table 4.6) demonstrates angular velocity estimation accuracy under disturbances and many variables needing to be tracked.

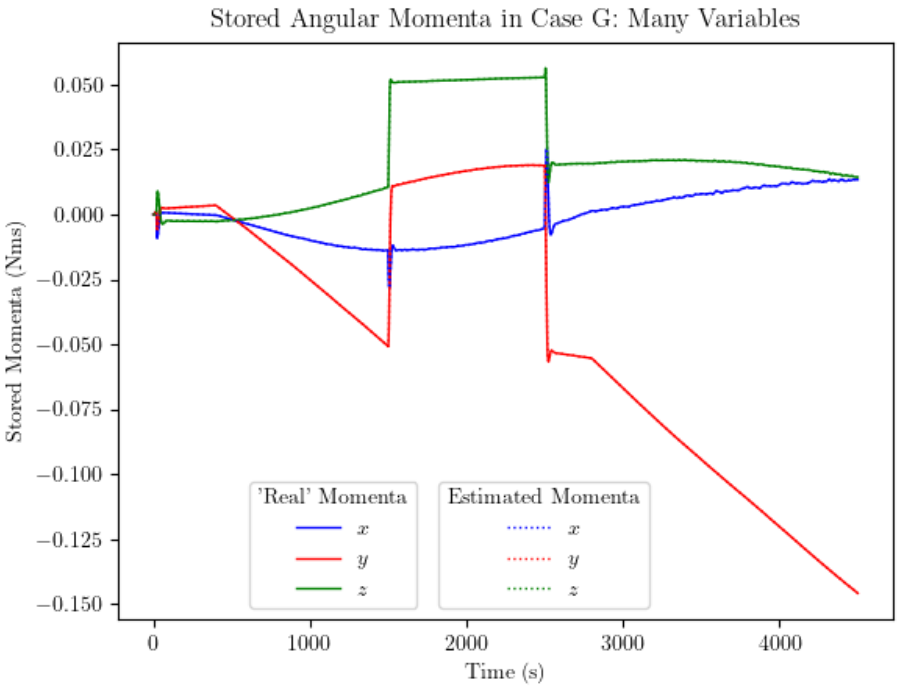


Figure 4.20: Case G (Section 4.4.8, case details in Table 4.6) demonstrates angular momentum estimation accuracy under disturbances and many variables needing to be tracked.

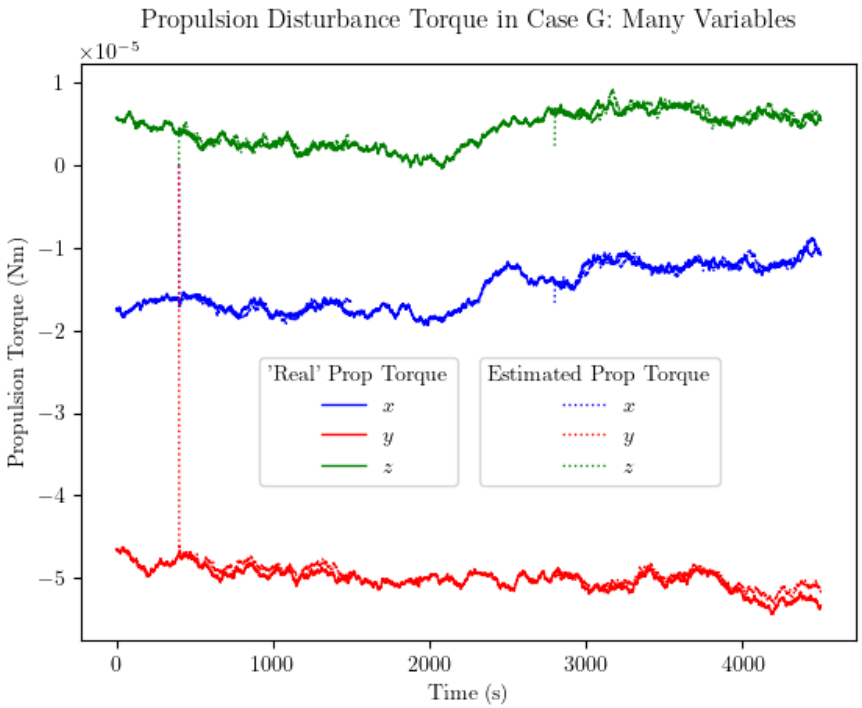


Figure 4.21: Case G (Section 4.4.8, case details in Table 4.6) demonstrates propulsion torque estimation accuracy under disturbances and many variables needing to be tracked.

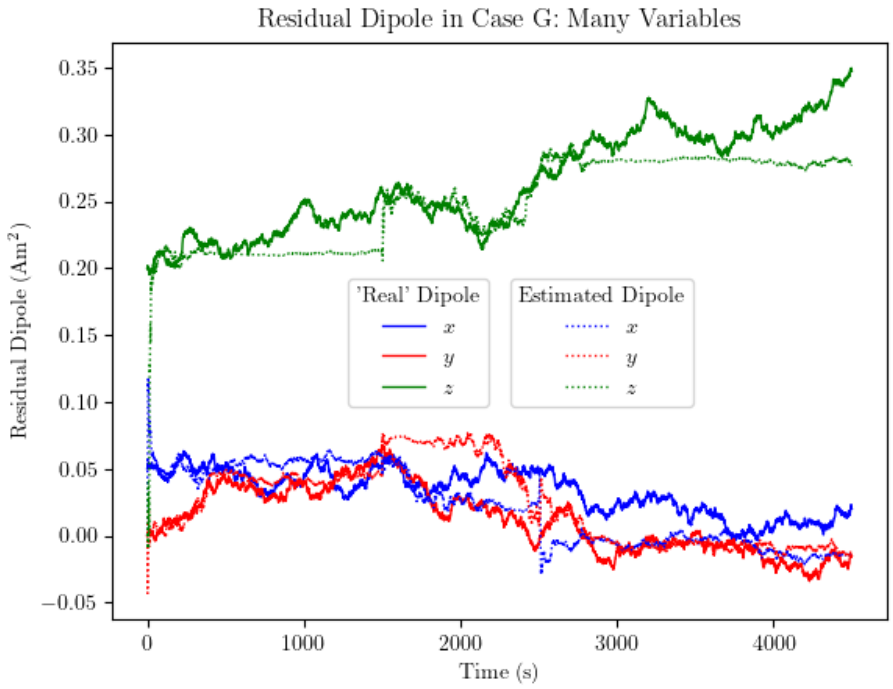


Figure 4.22: Case G (Section 4.4.8, case details in Table 4.6) demonstrates residual dipole estimation accuracy under disturbances and many variables needing to be tracked.

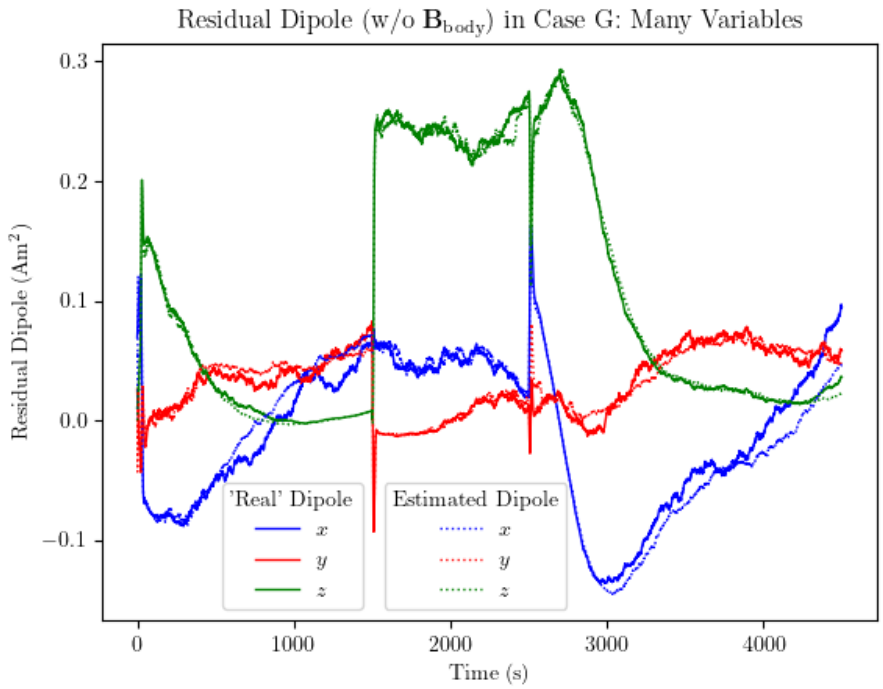
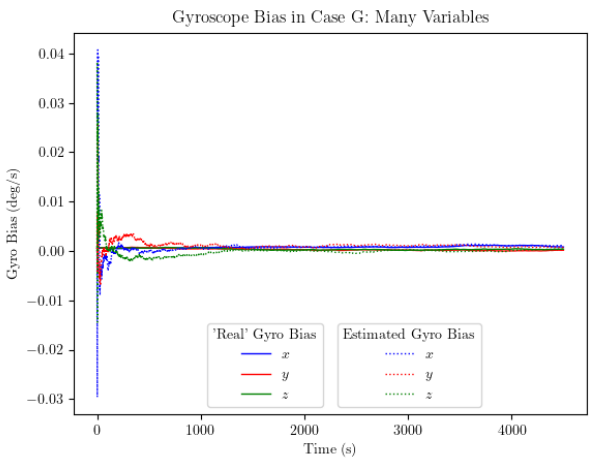
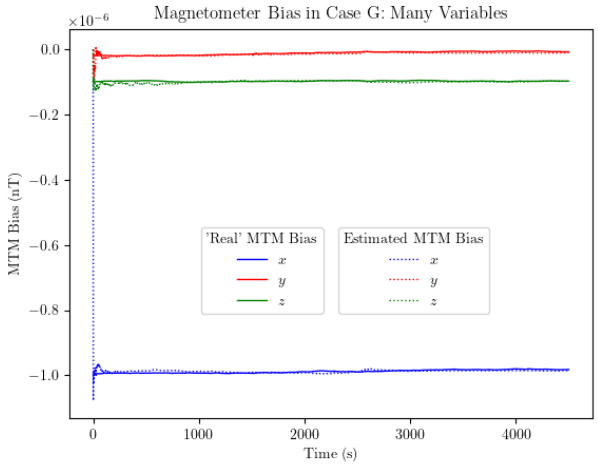


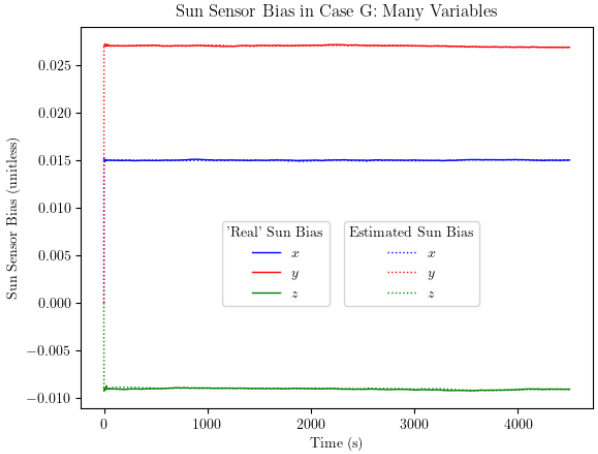
Figure 4.23: Case G (Section 4.4.8, case details in Table 4.6) demonstration of residual dipole (perpendicular to **B** field) estimation accuracy under disturbances and many variables needing to be tracked.



(a) Gyroscope Bias

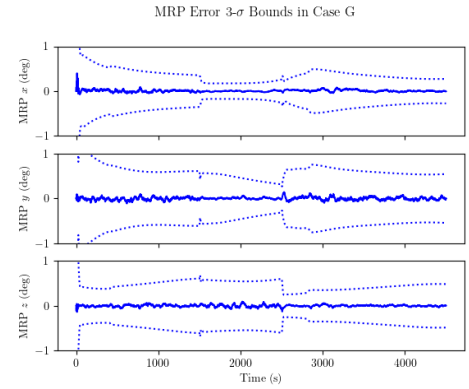


(b) Magnetometer Bias

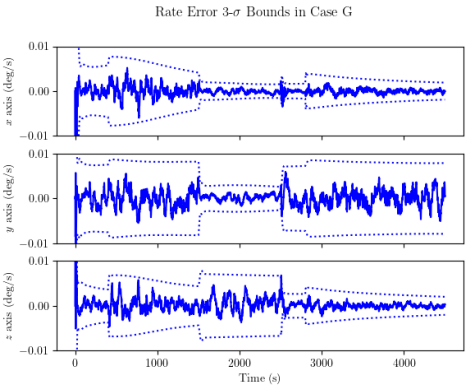


(c) Sun Sensor Bias

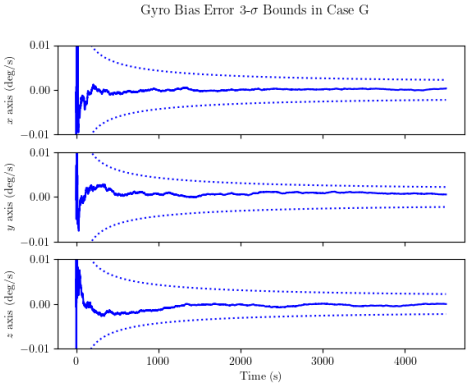
Figure 4.24: Case G (Section 4.4.8, case details in Table 4.6) demonstration of bias estimation accuracy for various sensors under disturbances and many variables needing to be tracked.



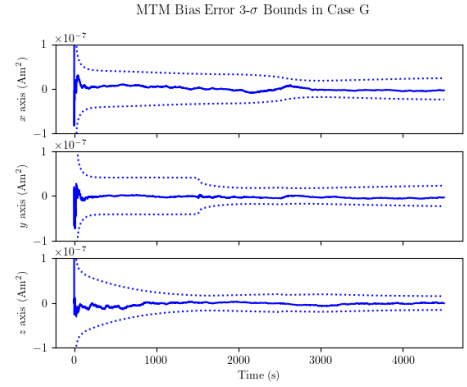
(a) Attitude (MRP)



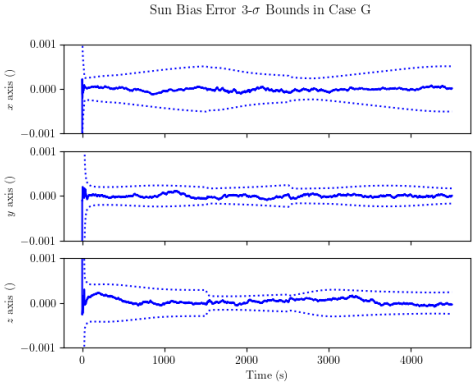
(b) Angular Velocity



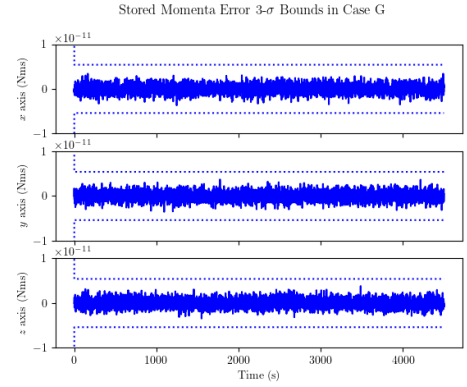
(c) Gyro Bias



(d) MTM Bias



(e) Sun Bias



(f) Stored Momenta

Figure 4.25: Case G (details in Table 4.6) estimation errors with 3σ error bounds. The estimation errors stay within the 3σ bounds and, in most cases, converge faster than the bounds.

4.5 Extensions

Given the modular and far-reaching nature of the dynamics, disturbance, and bias-aware approach, there are many ways it can be extended. This can be used to cover more and different systems, to improve speed and accuracy, or to include a wider variety of variables. A few ideas are listed in this section.

Further stability analysis on this estimator is needed. Monte Carlo tests are a good start, performed on a variety of spacecraft across a variety of orbits, with different sensor suites, and for varying dynamic parameters such as scale and inertia distribution. The impact of mis-modeling inertia parameters, torque values, or the local magnetic field should also be studied, and perhaps these could be included in future versions of the estimator [35]. There are potential questions of observability due to the configurable nature of this framework. For example, if a user adds two different all-in-one disturbance torques, these have a high risk of being impossible to separate (unless there is a good initial guess and they are very slow-varying). An investigation (analytical or experimental) to describe what configurations are observable or not, and under what conditions it is likely to converge, is worth performing, and the results could be added to the code as built-in warnings (e.g., when a case is unobservable) for future users.

The extended Kalman Filter can have difficulties when there are "corners" in the system relating to the state. One example is when the faces affected by drag and SRP change through rotation. One approach to deal with these sharp changes is to approximate the torques using spherical harmonics, based on the orientation of the ram (for drag) and sunward (for SRP) vectors in the body frame.

There is the question of what happens if the disturbance models (or the inertia matrix and other properties) are incorrect. If the estimator has enough "slack," small modeling errors or inaccuracies should be handled well. However, wild mis-estimates, or those that interact with other aspects of the dynamics, may have a greater effect. Further research to understand when these cases occur and what their impact is would be well-warranted. Similar questions exist for mis-estimating variances and noise on sensors, actuators, disturbances, and other parameters. This includes more complex or specific questions such as how solar panel degradation can be modeled as bias and the effect of a spacecraft's own dipole (inherent, or from magnetorquers, and potentially electrical activity) on magnetometer bias. All of this also ties into classic estimation questions regarding choosing parameters, covariances, and noise, and how to tune these values for a given application, as well as more modern questions regarding choosing correct error and evolution models, and cases of dependency.

There are many questions relating to varying parameters in this approach as well. For example, how well can it track multiple time-varying disturbance torques? There is the possibility of including an all-in-one disturbance torque (estimated and time-varying, as described in Section 3.4.4.c) instead of each torque individually; this could reduce computational complexity and rely less on specific knowledge of the system.

There is room to extend this work in new directions. An expectation-maximization approach is worth considering, switching between estimating the state based on dynamics and estimating system properties–like mass, inertia matrix properties, and actuator misalignment, as well as properties currently covered in this system, like residual dipole or biases. Specifically,

looking at combining the modularity of this approach (and the SRUKF/UKF formulations) with the APKF in Yoon, Riesing, and Cahoy [35]. There are many other approaches to parameter estimation that could influence extensions of this work. A particle filter-based approach would also be interesting to see, as it may better capture the nonlinearities in this system.

There are many extensions beyond this as well. The modular and generalizable nature of this framework makes it applicable to almost any satellite attitude determination problem, so there are many avenues for future work and applications.

4.6 Conclusion

This chapter presents a framework for the modeling and estimation of spacecraft attitude, addressing the critical challenges of adaptability, precision, and autonomy in Attitude Determination and Control Systems (ADCS). Starting with a modular system definition detailed in Section 4.2, the chapter develops a flexible representation of satellite dynamics, including mass properties, actuator characteristics, sensor models, and environmental disturbances. This modularity ensures that the framework can be seamlessly adapted to various satellite architectures, from underactuated CubeSats to large, high-inertia spacecraft, without extensive customization. Specific examples include modeling coarse sun sensors and magnetometers, which are common in resource-limited CubeSats, as well as capturing dynamic effects such as residual dipoles and propulsion misalignment.

Section 4.3 introduces the computational implementation of the framework, focusing on scalability and integration with estimation techniques. This implementation leverages modular class structures to simulate real-world conditions, enabling the evaluation of sensor inaccuracies, actuator biases, and dynamic disturbances. The software's ability to flexibly incorporate diverse components highlights its utility as a tool for both design and real-time operations. The dynamic-aware estimation methods discussed in Section 4.3.5, such as the Unscented Kalman Filter (UKF) and its square root variant, are then integrated into this framework to address the challenge of achieving high-precision attitude determination. These methods prove effective at dynamically adjusting to environmental disturbances and noisy sensor data, even in scenarios with significant initial attitude errors or actuator limitations.

The framework is validated in Section 4.4 through a series of test cases that span diverse mission profiles. For instance, the TRMM satellite scenario demonstrates the framework's ability to reduce angular estimation errors to less than 0.01° in low Earth orbit (LEO) conditions, even when subjected to significant initial attitude and bias errors. Specifically, dynamics-aware estimation using this modular system outperforms USQUE by up to orders of magnitude in terms of angular and angular velocity errors when applied to a system with high-quality sensors (e.g., TRMM in Cases A and B, Sections 4.4.2 and 4.4.3). CubeSat scenarios highlight the adaptability of the framework, showcasing its capability to track disturbances dynamically and correct biases without requiring extensive operator intervention. These results confirm the framework's capacity to handle a variety of satellite configurations and operational conditions, bridging the gap between traditional, rigid ADCS designs and the need for mission-agnostic, scalable solutions.

Despite these significant advancements, several opportunities for future work remain. A

simple approach of separating low-frequency bias and high-frequency noise on sensors and actuators works in many cases, but a more detailed method of system identification would be useful. This could also be applied to separate out specific disturbance torques from an all-in-one torque or to determine which corrections (such as dipole) are necessary. There are sensors and a few actuators that are not yet implemented, such as star cameras, control moment gyroscopes, and Earth horizon sensors. Extending the system to cover these (which should be simple, given the class-based structure) would increase the number of use cases for this system.

Improving the real-time computational efficiency of the estimation algorithms is critical for enabling rapid responses to changes during high-demand missions. As this framework is tested in Python, it can likely be made much faster using C++ or other compiled languages. Hardware-in-the-loop (HIL) testing could provide valuable insights by validating the system in more realistic operational conditions, while the integration of machine learning techniques offers potential for enhancing adaptability in highly uncertain environments.

This work complements and extends existing state-estimation methodologies for nanosatellites with uncertain disturbances and imperfect actuators and sensors. It does so while emphasizing flexibility, modularity, and the incorporation of high-noise data. While approaches like Yoon et al.'s APKF demonstrate the utility of detailed actuator modeling and high-accuracy sensors (such as star trackers), this method prioritizes adaptability and generalization. By leveraging advanced filtering techniques such as the UKF and SRUKF, this framework provides improved nonlinearity handling and numerical stability, making it broadly applicable across diverse mission scenarios.

The concepts and work laid out in this chapter tie to the key components of this thesis: starting from the areas of work in Section 3.2, the contributions in Section 3.3 are built to address the technical gaps in Section 2.3.2. The content of this chapter comes directly from the **Improved Attitude State Estimation** area of work (Section 3.2.1) to create an estimator that is dynamics- and disturbance-aware. By building from a unified modeling framework capable of supporting diverse satellite architectures (part of the **Generalized ADCS** area of work in Section 3.2.5), this chapter describes the **Generalized Disturbance- and Dynamics-Aware Estimator** (Section 3.3.1), which helps close the **Generalized ADCS** technical gap in Section 2.3.2.a. Tracking and measuring disturbances is a necessary step to include them in control efforts, forming a basis for the **Direct Countering of Disturbances** area of work (Section 3.2.3) and the **Disturbance Modeling for Control Inclusion** contribution (Section 3.3.2), further explored in Chapter 6. The modular system definition and dynamic-aware estimation techniques enable satellites to adapt autonomously to changing conditions, reducing the need for mission-specific customization, and help close the **Autonomous Rapid Retargeting** technical gap, described in Section 2.3.2.d. This chapter lays the foundation for the generalized control laws discussed in Chapter 5 and the trajectory planning methods explored in Chapter 7 by creating a unified modular framework. The modeling and estimation techniques presented here represent a critical step toward achieving the dissertation's overarching vision of a fully autonomous, adaptable, and mission-agnostic ADCS. These contributions provide robust solutions for addressing the challenges of modern satellite operations and open new avenues for innovation in the field.

Chapter 5

Generalizing ADCS Goals and Control Laws

5.1 Introduction

Most of the rest of this thesis focuses on specific components or algorithms that can be used to improve ADCS. These are each (comparatively) easy to describe and to define–each can be conceptualized as a discrete module. However, throughout their development, there has been an emphasis on a different, overarching approach to ADCS. This chapter describes that approach and its significance.

As outlined in Chapter 3, a key goal and theme of this dissertation is to improve ADCS autonomy. While ADCS is already largely autonomous—pointing estimation and control algorithms typically run onboard, slews are often pre-planned and executed automatically, and spacecraft ideally operate with minimal human involvement—there are still limitations, and much could be made more autonomous. For example, the direction and method of slews are often planned ahead of time, on the ground. Many satellites still have a division between "slew" mode and "pointing" mode, with separate control algorithms (or method of guide-point setting) for each. This contrasts with a unified approach that could dynamically handle both pointing goals at various times and slewing to a new target. Similarly, active disturbance correction, as described in Chapter 6, if included at all, is typically static, predefined at the mission's outset for specific disturbances, and seldom adaptive to changing conditions.

There are good reasons for the current state of ADCS Autonomy. It is common in controls engineering to prefer simple laws that can be proven to work. This is doubly true in space engineering, with its high cost, long flight times, value in flight heritage, and lack of second chances. Similar patterns exist in robotics, as Russ Tedrake illustrates in his exploration of early walking robots [129]. The first (and for decades, only) walking robots were based on strict control algorithms that maintained constant control availability, at the cost of increased energy (20x!) and slower, less-natural motions.[1] Yet, as in robotics, a more flexible, generalizable, and adaptable approach holds immense value for ADCS.

The vision for an advanced ADCS system (specifically the **General ADCS Approach**

[1]The author would like to highly recommend reading the "Motivation" section of the first chapter of that text–it beautifully describes the value in more general approaches to dynamic control.

and **Autonomous Rapid Retargeting** technical gaps in Sections 2.3.2.a and 2.3.2.d) is a system with the ability to:

- estimate and account for its own biases, uncertainties, and noise based on knowledge of mass properties, actuators, and sensors;
- incorporate environmental factors, such as disturbances and magnetic fields, into its operations (such as the local B-field for magnetic control);
- apply a unified control framework (or select from options!) that is suitable for both pointing and slews, regardless of the satellite architecture or environment;
- automatically combine goals, such as reaction wheel desaturation during slews or continuously during pointing [127]; and
- balance control efforts dynamically across actuators, using systems like magnetorquers in conjunction with reaction wheels.

This dissertation contributes to this vision and attempts to address the technical gaps in Section 2.3.2. Specifically, this work

- creates a **Generalized Disturbance- and Dynamics-Aware Estimator** (see contribution in Section 3.3.1) in Chapter 4 with disturbance incorporation
- explores methods for the **Direct Countering of Disturbances** (see contribution in Section 3.3.2) in Chapter 6 in a variety of control laws,
- furthers **Attitude Trajectory Planning** (see contribution in Section 3.3.3) in Chapter 7 which works across many sets of goals, satellites, and environments.

This chapter explores several complementary developments that enhance autonomy and generalizability, largely coming from the **ADCS Operations and Goals** (Section 3.2.2) and **Generalized ADCS** (Section 3.2.5) areas of work. By tying together the other components, and providing extensions beyond them, the work contained herein contributes to the **Improved Autonomy for ADCS** (Section 3.3.4) and **ADCS Architecture for Weak or Underactuated Satellites** (Section 3.3.5) contributions. Section 5.2 introduces methods for encoding various goal types, applicable to all control laws. Section 5.3 examines reducing fully specified orientation goals to simpler two-vector alignment, enabling broader applicability. Finally, Section 5.4 discusses how control laws can be adapted for general use across different satellites and goal types.

5.2 Specifying Time-Varying Goals Succinctly

One significant way to improve ADCS autonomy is to not specify full slews or even specific orientations as satellite goals. Instead, the satellite can be given a set of goals and time periods. The goals fall into one of several categories, also shown in Table 5.1:

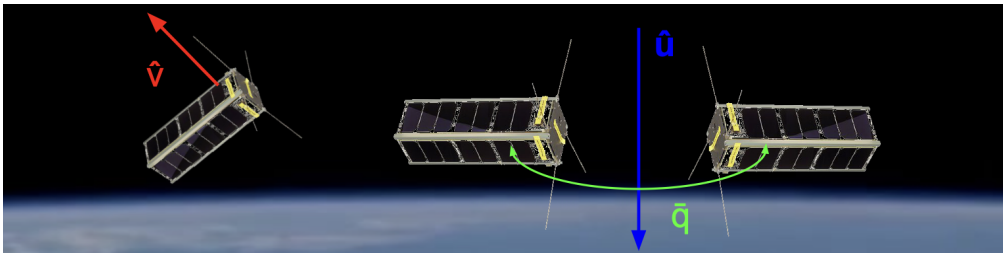


Figure 5.1: If the goal is to point the camera ($\hat{v}$) in the nadir direction ($\hat{u}$), any of the rotations shown are acceptable. Fully specifying the orientation as a goal over-constrains the problem.

- No goal: no specific goal during this time period (this is useful when the planner has "free time" to optimize for goals at other times in the trajectory).
- Align body frame vector and generic world frame vector: this provides a satellite vector body frame (propulsion, camera axis, solar panel normal, etc.) and one of several directions in the world frame that are defined by the environment: nadir, zenith, ram, anti-ram, positive orbit normal, negative orbit normal, towards sun, away from sun. The goal is to align these vectors.
- Align body frame vector and specified world frame direction: this provides a satellite vector body frame and a specified direction in the world frame. The goal is to align these vectors.
- Point body frame vector at specific world frame location: this provides a satellite vector body frame and a specified position in the world frame. This can either be in the inertial global frame or an ECEF position. (It could also be a specified latitude, longitude, and altitude.) The goal is to align the body frame vector with the direction between the spacecraft and the specified position.
- Specified orientation: This passes a desired orientation as an MRP (this allows a constant vector size, valuable for plugging into trajectory planners in C++, for example). The goal is to match this position.

The cases where the body frame vector aligns with a vector or position in the world frame include most cases where the satellite has a goal to meet. These cases can generally be used (along with keep-out zones, where a body frame vector is prevented from pointing within a certain angle of the sun) to describe the desired goals. Describing goals in this way has an advantage over specifying a full orientation, because it allows the planner (or control law) flexibility in how to meet those goals, as shown in Figure 5.1 [130].

5.3 Aligning Two Vectors

Many of the goals described in this chapter are about aligning two vectors. This section describes a derivation of the ways to define all quaternions that meet this goal. This allows

Table 5.1: Control Goal Type Summary

Property	**No Goal**	**Align with Generic Direction**	**Align with Specific Vector**	**Point at Specific Location**	**Specific Orientation**
Goal	None	$A(\bar{q})\mathbf{b} = \mathbf{d}$	$A(\bar{q})\mathbf{b} = \mathbf{w}$	$A(\bar{q})\mathbf{b} = \frac{\mathbf{p}-\mathbf{r}}{\|\mathbf{p}-\mathbf{r}\|}$	$\bar{q} = \bar{q}_{\text{desired}}$
Solution Set	Entire orientation space in SO(3)	Rotations on alignment axis (S^1)	Rotations on alignment axis (S^1)	Rotations on alignment axis (S^1)	One attitude (point in SO(3))
Dimension	3	1	1	1	0
Body Value	None	Body frame vector **b**	Body frame vector **b**	Body frame vector **b**	Spacecraft attitude $\bar{q}$
External Value	None	Generic direction in world frame	Specified vector **w** in world frame	Target direction $\frac{\mathbf{p}-\mathbf{r}}{\|\mathbf{p}-\mathbf{r}\|}$	Desired orientation $\bar{q}_{\text{desired}}$
Reference Frames	N/A	· ECI · Orbital	· ECI · ECEF · Orbital	· ECI · ECEF · LLA	N/A
Required Inputs	None	· Body frame vector **b** · Generic external direction (e.g., nadir, sunward)	· Body frame vector **b** · Specified world frame vector **w**	· Body frame vector **b** · Target position **p** · Spacecraft position **r**	Desired orientation[b] $\bar{q}_{\text{desired}}$
Examples	· Free drift[a] · RW momentum desaturation	· Point solar panel at Sun · Point camera nadir	· Point camera celestially · Point antenna arbitrarily	· Point camera at a satellite · Track ground-based target	· Hold inertial orientation for imaging · Docking
Subset of:	N/A	· No Goal · Align w/ Specific Vector	No Goal	· No Goal · Align w/ Specific Vector	Subset of all others

[a] Useful in trajectory planning for non-important time spans, allowing this time to be used to setup or conserve resources for more critical phases of a trajectory.

[b] Specified in code as Modified Rodrigues Parameters (MRPs) [18], [125] for consistent vector size in C++.

for quaternion-based control algorithms to be modified to vector-pointing control algorithms simply by choosing a method to select a goal quaternion from the set of possible options.

5.3.1 Derivation

Consider a unit vector $\hat{\mathbf{v}}$, representing a direction in the spacecraft body frame. Simple examples include the central vector of a directed antenna, the boresight of a camera, or the normal vector of a solar panel. This vector represents a direction in the satellite frame that we wish to point in the world frame.

We have an additional unit vector $\hat{\mathbf{u}}$. This represents a direction in the world frame. This could be a vector from the antenna to a ground station, a vector from the spacecraft camera to an imaging target, or a vector from the solar panel to the Sun.

Let the minimum angle between $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$ be

$$\theta = \arccos\left(\hat{\mathbf{v}} \cdot \hat{\mathbf{u}}\right). \tag{5.1}$$

Also note that $|\theta| \leq \pi$.

We wish to find a formula that will describe the set of all unit quaternions that would rotate $\hat{\mathbf{v}}$ so that $\hat{\mathbf{u}} = \bar{q}^{-1} \circ \hat{\mathbf{v}}$, as stated in Equation 3.17. Applying Equation 3.10, this means we wish to find all $\bar{q}$ such that

$$\hat{\mathbf{u}} = \bar{q}^{\,-1} \circ \hat{\mathbf{v}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\bar{q} \begin{Bmatrix} 0 \\ \hat{\mathbf{v}} \end{Bmatrix} \bar{q}^{\,-1} \right]. \tag{5.2}$$

Rearranging (for clarity) and applying Equation 3.6 on the last two terms in the multiplication, we find

$$\begin{bmatrix} 0 \\ \hat{\mathbf{u}} \end{bmatrix} = \left[\bar{q} \begin{Bmatrix} \hat{\mathbf{v}} \cdot \mathbf{q}_v \\ q_0 \hat{\mathbf{v}} - \hat{\mathbf{v}} \times \mathbf{q}_v \end{Bmatrix} \right]. \tag{5.3}$$

Applying Equation 3.6 once more, as well as Equation 3.9, and relying on our requirement that $\bar{q}$ is a unit quaternion, we arrive at

$$\begin{bmatrix} 0 \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} (\hat{\mathbf{v}} \cdot \mathbf{q}_v) q_0 - \mathbf{q}_v \cdot (q_0 \hat{\mathbf{v}} - \hat{\mathbf{v}} \times \mathbf{q}_v) \\ q_0 (q_0 \hat{\mathbf{v}} - \hat{\mathbf{v}} \times \mathbf{q}_v) + \mathbf{q}_v (\hat{\mathbf{v}} \cdot \mathbf{q}_v) + \mathbf{q}_v \times (q_0 \hat{\mathbf{v}} - \hat{\mathbf{v}} \times \mathbf{q}_v) \end{bmatrix}. \tag{5.4}$$

Through cancellation and triple product identities, we simplify to

$$\begin{bmatrix} 0 \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} 0 \\ q_0^2 \hat{\mathbf{v}} + \mathbf{q}_v (\hat{\mathbf{v}} \cdot \mathbf{q}_v) - 2 q_0 \hat{\mathbf{v}} \times \mathbf{q}_v - \mathbf{q}_v \times (\hat{\mathbf{v}} \times \mathbf{q}_v) \end{bmatrix} \tag{5.5}$$

and finally to

$$\hat{\mathbf{u}} = q_0^2 \hat{\mathbf{v}} + \mathbf{q}_v (\hat{\mathbf{v}} \cdot \mathbf{q}_v) - 2 q_0 \hat{\mathbf{v}} \times \mathbf{q}_v - \mathbf{q}_v \times (\hat{\mathbf{v}} \times \mathbf{q}_v). \tag{5.6}$$

By the vector triple product identity, this can also be written as

$$\hat{\mathbf{u}} = \hat{\mathbf{v}} \left(q_0^2 - \|\mathbf{q}_v\|^2 \right) + 2 \mathbf{q}_v (\hat{\mathbf{v}} \cdot \mathbf{q}_v) - 2 q_0 \hat{\mathbf{v}} \times \mathbf{q}_v. \tag{5.7}$$

Our set of unit quaternions that rotate $\hat{\mathbf{v}}$ onto $\hat{\mathbf{u}}$ are all unit quaternions that satisfy Equation 5.7. Any quaternion in this set would satisfy the pointing goals of a spacecraft,

allowing more flexibility and smaller orientation change than requiring a specific 3D orientation. However, we would like a formula that will express the entire set (instead of just naming them as the solutions to a complicated equation), so that all relevant quaternions can be described. This formula will rely only on $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$, and a parametric variable that indexes the set. We will derive this formula from Equation 5.7 by finding other relationships between $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$.

First, we find $\hat{\mathbf{v}} \cdot \hat{\mathbf{u}}$. Using Equation 5.7, we find

$$\hat{\mathbf{v}} \cdot \hat{\mathbf{u}} = \hat{\mathbf{v}} \cdot \hat{\mathbf{v}} \left(q_0^2 - \|\mathbf{q}_v\|^2\right) + 2(\hat{\mathbf{v}} \cdot \mathbf{q}_v)^2 - 2q_0 \hat{\mathbf{v}} \cdot (\hat{\mathbf{v}} \times \mathbf{q}_v), \tag{5.8}$$

which simplifies to

$$\hat{\mathbf{v}} \cdot \hat{\mathbf{u}} = q_0^2 - \|\mathbf{q}_v\|^2 + 2(\hat{\mathbf{v}} \cdot \mathbf{q}_v)^2. \tag{5.9}$$

We also find

$$\begin{aligned} \hat{\mathbf{v}} \times \hat{\mathbf{u}} =& \hat{\mathbf{v}} \times \hat{\mathbf{v}} \left(q_0^2 - \|\mathbf{q}_v\|^2\right) \\ &+ 2(\hat{\mathbf{v}} \cdot \mathbf{q}_v)(\hat{\mathbf{v}} \times \mathbf{q}_v) - 2q_0 \hat{\mathbf{v}} \times (\hat{\mathbf{v}} \times \mathbf{q}_v), \end{aligned} \tag{5.10}$$

which simplifies to

$$\hat{\mathbf{v}} \times \hat{\mathbf{u}} = 2(\hat{\mathbf{v}} \cdot \mathbf{q}_v)(\hat{\mathbf{v}} \times \mathbf{q}_v) - 2q_0 \left(\hat{\mathbf{v}} \left(\hat{\mathbf{v}} \cdot \mathbf{q}_v\right) - \mathbf{q}_v\right). \tag{5.11}$$

Rearranging Equation 5.11 to solve for $\hat{\mathbf{v}} \times \mathbf{q}_v$ and substituting it into Equation 5.7, we find

$$\begin{aligned} \hat{\mathbf{u}} =& \hat{\mathbf{v}} \left(q_0^2 - \|\mathbf{q}_v\|^2\right) + 2\mathbf{q}_v(\hat{\mathbf{v}} \cdot \mathbf{q}_v) \\ &- 2q_0 \left(q_0 \hat{\mathbf{v}} + \frac{\frac{1}{2}\left(\hat{\mathbf{v}} \times \hat{\mathbf{u}}\right) - q_0 \mathbf{q}_v}{\hat{\mathbf{v}} \cdot \mathbf{q}_v}\right) \end{aligned} \tag{5.12}$$

and simplify it to

$$\hat{\mathbf{u}} = -\hat{\mathbf{v}} \|\bar{q}\|^2 + \frac{-q_0 \hat{\mathbf{v}} \times \hat{\mathbf{u}} + 2\left(q_0^2 + (\hat{\mathbf{v}} \cdot \mathbf{q}_v)^2\right) \mathbf{q}_v}{\hat{\mathbf{v}} \cdot \mathbf{q}_v}. \tag{5.13}$$

As we are only considering unit quaternions, we know $\|\bar{q}\|^2 = 1$, so we find

$$\hat{\mathbf{u}} = -\hat{\mathbf{v}} + \frac{-q_0 \hat{\mathbf{v}} \times \hat{\mathbf{u}} + 2\left(q_0^2 + (\hat{\mathbf{v}} \cdot \mathbf{q}_v)^2\right) \mathbf{q}_v}{\hat{\mathbf{v}} \cdot \mathbf{q}_v}. \tag{5.14}$$

We now rearrange Equation 5.9 to solve for $\hat{\mathbf{v}} \cdot \mathbf{q}_v$ (while preserving the sign ambiguity on the square root) and substitute it into Equation 5.14. This gives us

$$\hat{\mathbf{u}} = -\hat{\mathbf{v}} + \frac{-q_0 \hat{\mathbf{v}} \times \hat{\mathbf{u}} + \left(q_0^2 + \|\mathbf{q}_v\|^2 + \hat{\mathbf{v}} \cdot \hat{\mathbf{u}}\right) \mathbf{q}_v}{\pm\sqrt{\frac{1}{2}\left(\hat{\mathbf{v}} \cdot \hat{\mathbf{u}} - q_0^2 + \|\mathbf{q}_v\|^2\right)}}, \tag{5.15}$$

which (remembering $\|\bar{q}\|^2 = 1$) simplifies to

$$\hat{\mathbf{u}} = -\hat{\mathbf{v}} \pm \frac{-q_0 \hat{\mathbf{v}} \times \hat{\mathbf{u}} + (1 + \hat{\mathbf{v}} \cdot \hat{\mathbf{u}}) \mathbf{q}_v}{\sqrt{\frac{1}{2}(1 + \hat{\mathbf{v}} \cdot \hat{\mathbf{u}} - 2q_0^2)}}. \tag{5.16}$$

Rearranging and simplifying, we can express $\mathbf{q}_v$ in terms of $\hat{\mathbf{v}}$, $\hat{\mathbf{u}}$, and q_0:

$$\mathbf{q}_v = \frac{\left(q_0\left(\hat{\mathbf{v}}\times\hat{\mathbf{u}}\right) \pm \frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\sqrt{2}}\sqrt{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}-2q_0^2}\right)}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}. \tag{5.17}$$

By applying Equation 5.17 and replacing q_0 with a parametric variable s, we can define a formula, $\bar{f}^s(s)$ for the set of all unit quaternions that rotate $\hat{\mathbf{v}}$ onto $\hat{\mathbf{u}}$:

$$\bar{f}^s(s) = \left\{ \begin{matrix} s \\ \frac{\left(s(\hat{\mathbf{v}}\times\hat{\mathbf{u}}) \pm \frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\sqrt{2}}\sqrt{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}-2s^2}\right)}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}} \end{matrix} \right\}. \tag{5.18}$$

Inserting Equation 5.18 into Equation 5.7 and simplifying shows that it is correct.

It should be noted that Equation 5.18 will not accept all values of s. It can be seen that $|s| \leq \sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}$, or else the vector portion of $\bar{f}^s(s)$, $\mathbf{f}_v^s(s)$, will have an imaginary component.

This bound makes intuitive sense. Using Equation 5.1 and the cosine half-angle identity, we can rewrite the bounds on s to be $|s| \leq \sqrt{\frac{1+\cos\theta}{2}} = \cos\frac{\theta}{2}$. Considering Equation 3.13, we see that $s = \cos\frac{\phi}{2}$, if ϕ is the angle of rotation of the associated quaternion. The bounds on s indicate that $\cos\frac{\phi}{2} \leq \cos\frac{\theta}{2}$, or that $|\phi| \geq |\theta|$ (assuming that both ϕ and θ have been unwrapped to fall on $[-\pi, \pi]$), where θ is the minimum angle between $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ and is thus the minimum possible rotation angle to meet our requirements.

The values of $\bar{f}^s(s)$ at the bounds of s are worth noting. We see that

$$\bar{f}^s\left(\pm\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}\right) = \left\{ \begin{matrix} \pm\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}} \\ \pm\frac{1}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}\left(\hat{\mathbf{v}}\times\hat{\mathbf{u}}\right) \end{matrix} \right\}. \tag{5.19}$$

This simplifies (noting that $\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\| = \sin\theta$) to

$$\pm\bar{x} = \bar{f}^s\left(\pm\sqrt{\frac{1+\cos\theta}{2}}\right) = \left\{ \begin{matrix} \pm\cos\frac{\theta}{2} \\ \pm\hat{\mathbf{x}}\sin\frac{\theta}{2} \end{matrix} \right\}, \tag{5.20}$$

where $\hat{\mathbf{x}} = \frac{(\hat{\mathbf{v}}\times\hat{\mathbf{u}})}{\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\|}$, so that the quaternion associated with $s = \sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}$ is the smallest possible rotation, rotated about the vector perpendicular to both $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$.

We should also consider the case when $s = 0$, where (noting that $\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\| = \sqrt{2(1+\cos\theta)}$ and $\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\| = 2\cos\frac{\theta}{2}$) we find

$$\pm\bar{y} = \bar{f}^s(0) = \left\{ \begin{matrix} 0 \\ \pm\frac{\left(\frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\sqrt{2}}\sqrt{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}\right)}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}} \end{matrix} \right\} = \left\{ \begin{matrix} 0 \\ \pm\hat{\mathbf{y}} \end{matrix} \right\}, \tag{5.21}$$

where $\hat{\mathbf{y}} = \frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\|}$, so that the quaternion associated with $s = 0$ is the rotation about the unit vector halfway between $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$.

Table 5.2: Quaternion Set Formulation Summary.

Variable	Definition	Reference	Bounds	Summary
s	$\left\{ \begin{array}{c} s \\ \frac{\left(s(\hat{\mathbf{v}}\times\hat{\mathbf{u}})\pm\frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\sqrt{2}}\sqrt{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}-2s^2}\right)}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}} \end{array} \right\}$	Eq. 5.18	$-\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}} \leq s \leq \sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}$	$s = q_0$
t	$\left\{ \begin{array}{c} t\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}} \\ \frac{t(\hat{\mathbf{v}}\times\hat{\mathbf{u}})\pm(\hat{\mathbf{v}}+\hat{\mathbf{u}})\sqrt{1-t^2}}{\sqrt{2(1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}})}} \end{array} \right\}$	Eq. 5.24	$-1 \leq t \leq 1$	Simplified bounds
β	$\bar{y}\sin\beta + \bar{x}\cos\beta$	Eq. 5.26	$\beta \in \mathbb{R}$	Continuous, no $\pm$ sign
ϕ	$\left\{ \begin{array}{c} \cos\frac{\phi}{2} \\ \frac{\frac{\cos\frac{\phi}{2}}{\cos\frac{\theta}{2}}(\hat{\mathbf{v}}\times\hat{\mathbf{u}})+(\hat{\mathbf{v}}+\hat{\mathbf{u}})\sqrt{1-\left(\frac{\cos\frac{\phi}{2}}{\cos\frac{\theta}{2}}\right)^2}}{2\cos\frac{\theta}{2}} \end{array} \right\}$	Eq. 5.31	$\arccos(\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}) \leq \lvert\phi\rvert \leq \pi$	ϕ is the rotation angle
α	$\frac{\bar{x}\cos\alpha+\bar{y}\sin\alpha\cos\frac{\theta}{2}}{\sqrt{\cos^2\alpha\cos^2\frac{\theta}{2}+\sin^2\frac{\theta}{2}}}$	Eq. 5.38	$\alpha \in \mathbb{R}$	α indexes rotation axis

5.3.2 Alternate Formulations

While we have found a formula to explore the set of rotations that rotate $\hat{\mathbf{v}}$ onto $\hat{\mathbf{u}}$, it has certain problems. The bound $|s| \leq \sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}$ is not elegant. More significantly, Equation 5.18 has a $\pm$ sign in it, so that $\bar{f}^s(s)$ is not unique. (While indeed an issue in Equation 5.18, the $\pm$ signs in Equations 5.20 and 5.21 are irrelevant, because $\bar{q}$ and $-\bar{q}$ represent the same rotation.) Lastly, the parametric variable, s, is notable as the scalar component of the quaternion, but does not have greater meaning. This section finds several alternate formulations for the acceptable quaternion set, summarized in Table 5.2.

5.3.2.a Simplifying Bounds

One simple adjustment is to simplify the bounds of the parametric variable. As s runs from 0 to $\pm\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}$, we perform a variable substitution with

$$s = t\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}. \tag{5.22}$$

The parametric variable t can then run from -1 to 1. This gives us the function

$$\bar{f}^t(t) = \left\{ \begin{array}{c} t\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}} \\ \frac{\left(t\sqrt{\frac{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}}{2}}(\hat{\mathbf{v}}\times\hat{\mathbf{u}})\pm\frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\sqrt{2}}\sqrt{(1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}})(1-t^2)}\right)}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}} \end{array} \right\}, \tag{5.23}$$

which can simplify to a new definition

$$\bar{f}^t(t) = \left\{ \begin{array}{c} t\cos\frac{\theta}{2} \\ \frac{1}{2\cos\frac{\theta}{2}}\left(t\left(\hat{\mathbf{v}}\times\hat{\mathbf{u}}\right)\pm\left(\hat{\mathbf{v}}+\hat{\mathbf{u}}\right)\sqrt{1-t^2}\right) \end{array} \right\} \tag{5.24}$$

with $-1 \leq t \leq 1$. The $\pm$ sign remains, but this provides a good start to a better formulation.

5.3.2.b Removing Ambiguous Sign

We can remove the $\pm$ sign by implementing a variable substitution of $t = \cos\beta$. If t runs from -1 to 1, we can let β run from $-\pi$ to π (equivalent to any real value of β as we only use trigonometric functions) and then substitute $\pm\sqrt{1-t^2}$ with $\sin\beta$.

This gives us

$$\bar{f}^{\beta}(\beta) = \bar{f}^{t}(\cos\beta) = \left\{ \begin{array}{c} \cos\beta\cos\frac{\theta}{2} \\ \frac{\cos\beta(\hat{\mathbf{v}}\times\hat{\mathbf{u}})\pm(\hat{\mathbf{v}}+\hat{\mathbf{u}})\sqrt{1-\cos^2\beta}}{2\cos\frac{\theta}{2}} \end{array} \right\} \tag{5.25}$$

which can simplify (once again noting that $\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\| = \sqrt{2(1+\cos\theta)} = 2\cos\frac{\theta}{2}$ and $\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\| = \sin\theta$) to a new definition

$$\bar{f}^{\beta}(\beta) = \left\{ \begin{array}{c} \cos\beta\cos\frac{\theta}{2} \\ \hat{\mathbf{x}}\sin\frac{\theta}{2}\cos\beta + \hat{\mathbf{y}}\sin\beta \end{array} \right\} \tag{5.26}$$

for any real β.

This expression is worth noting because it indicates that the space of acceptable rotations is a unit combination of two specific, simple rotations. Specifically, we have

$$\bar{f}^{\beta}(\beta) = \left\{ \begin{array}{c} \cos\frac{\theta}{2} \\ \hat{\mathbf{x}}\sin\frac{\theta}{2} \end{array} \right\} \cos\beta + \left\{ \begin{array}{c} 0 \\ \hat{\mathbf{y}} \end{array} \right\} \sin\beta. \tag{5.27}$$

The components in this equation are the bounds seen in Equations 5.20 and 5.21. In other words, we can write the space of rotations as

$$\bar{f}^{\beta}(\beta) = \bar{y}\sin\beta + \bar{x}\cos\beta. \tag{5.28}$$

5.3.2.c Physical Significance of Parametric Variable

We can also use formulations which more directly give the parametric variable significance. For example, in Equation 3.13, we see that the scalar element of the quaternion (equal to variable s in Equation 5.18) is equal to $\cos\frac{\phi}{2}$, where ϕ is the angle rotated if the rotation is defined around a single axis. We can apply the variable substitution $s = \cos\frac{\phi}{2}$ to Equation 5.18 and (using similar logic as before to remove the $\pm$ sign) we find

$$\bar{f}^{\phi}(\phi) = \left\{ \begin{array}{c} \cos\frac{\phi}{2} \\ \frac{\cos\frac{\phi}{2}(\hat{\mathbf{v}}\times\hat{\mathbf{u}})+\frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{2}\sqrt{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}-2\cos^2\frac{\phi}{2}}}{1+\hat{\mathbf{v}}\cdot\hat{\mathbf{u}}} \end{array} \right\}. \tag{5.29}$$

Using our definition for θ as the angle between $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$ (Equation 5.1) and simplifying, we arrive at

$$\bar{f}^{\phi}(\phi) = \left\{ \begin{array}{c} \cos\frac{\phi}{2} \\ \frac{\cos\frac{\phi}{2}(\hat{\mathbf{v}}\times\hat{\mathbf{u}})+(\hat{\mathbf{v}}+\hat{\mathbf{u}})\sqrt{\cos^2\frac{\theta}{2}-\cos^2\frac{\phi}{2}}}{2\cos^2\frac{\theta}{2}} \end{array} \right\}, \tag{5.30}$$

which can simplify to a new definition

$$\bar{f}^{\phi}(\phi) = \left\{ \begin{array}{c} \cos\frac{\phi}{2} \\ \frac{\frac{\cos\frac{\phi}{2}}{\cos\frac{\theta}{2}}(\hat{\mathbf{v}}\times\hat{\mathbf{u}})+(\hat{\mathbf{v}}+\hat{\mathbf{u}})\sqrt{1-\left(\frac{\cos\frac{\phi}{2}}{\cos\frac{\theta}{2}}\right)^2}}{2\cos\frac{\theta}{2}} \end{array} \right\} \tag{5.31}$$

for any valid ϕ, which is $\theta \leq |\phi| \leq \pi$ (or any value that meets these conditions when unwrapped to the interval $[-\pi, \pi]$). Equation 5.31 indexes the set of acceptable rotations based on ϕ, the angle of the rotation.

The formulation in Equation 5.31 uses a meaningful parametric variable, where the variable defines the angle of rotation in axis-angle formulation. There is a corresponding formulation where the parametric variable defines the axis of rotation. By comparing the vector component of Equation 5.18 with the axis-angle definition of the quaternion in Equation 3.13, we see that the axis of rotation is a weighted vector sum of $\hat{\mathbf{v}} \times \hat{\mathbf{u}}$ and $\hat{\mathbf{v}} + \hat{\mathbf{u}}$. We will define the axis of rotation as a unit vector falling in the unit circle defined by these two vectors, and will use the variable α to describe the position of the rotation axis inside the unit circle. Specifically, α represents the angle of the rotation axis with respect to $\hat{\mathbf{x}}$ (measured around $\hat{\mathbf{x}} \times \hat{\mathbf{y}}$). This is made clearer in Table 5.2, but it is important to note that α is **not** the angle of rotation. Using our definition in Equation 3.13, this can be written as

$$\bar{f}^{\alpha}(\alpha) = \begin{Bmatrix} \cos\frac{\gamma(\alpha)}{2} \\ \sin\frac{\gamma(\alpha)}{2}\left(\hat{\mathbf{x}}\cos\alpha + \hat{\mathbf{y}}\sin\alpha\right) \end{Bmatrix} \tag{5.32}$$

where $\gamma(\alpha)$ (the rotation angle around our defined axis) is an as-yet unknown function of α. This relationship can be found in a number of ways, including solving the axis-angle rotation using Rodruigues' rotation formula or drawing the vectors on the unit sphere and applying Napier's rules for right spherical triangles, but, in the interest of consistency, we will instead find the expression from our existing Equation 5.31. We are looking for an expression defining $\gamma(\alpha)$. From the scalar component we see that $\cos\frac{\gamma(\alpha)}{2} = \cos\frac{\phi}{2}$ and from the vector component we have

$$\frac{\sin\frac{\gamma(\alpha)}{2}\cos\alpha}{\sin\theta} = \frac{\cos\frac{\phi}{2}}{2\cos^2\frac{\theta}{2}} \tag{5.33}$$

and

$$\frac{\sin\frac{\gamma(\alpha)}{2}\sin\alpha}{\sqrt{2}\sqrt{1+\cos\theta}} = \frac{1}{2\cos\frac{\theta}{2}}\sqrt{1-\left(\frac{\cos\frac{\phi}{2}}{\cos\frac{\theta}{2}}\right)^2} \tag{5.34}$$

Equation 5.33 simplifies to

$$\sin\frac{\gamma(\alpha)}{2} = \frac{\tan\frac{\theta}{2}}{\cos\alpha}\cos\frac{\phi}{2}, \tag{5.35}$$

so, substituting in the scalar component expression, we have

$$\tan\frac{\gamma(\alpha)}{2} = \frac{\tan\frac{\theta}{2}}{\cos\alpha}. \tag{5.36}$$

Substituting this into Equation 5.34 will verify that it is also a solution of that equation. Therefore, Equation 5.36 meets our requirements. It is also worth noting that because $\cos\frac{\phi}{2} \geq 0$, that $\gamma(\alpha) = 2\arctan\left(\frac{\tan\frac{\theta}{2}}{\cos\alpha}\right)$ is a unique solution. However, to ensure that $\gamma(\alpha) > 0$ in all cases (and thus have a consistently positive angle of rotation to fully explore the rotation space), we must unwrap around the unit circle, with the final formula being

$\gamma(\alpha) = \pi - 2\arctan\left(\cos\alpha \cot\frac{\theta}{2}\right)$. Our new axis-based formulation can thus be written (keeping in mind that $\bar{q}$ and $-\bar{q}$ represent the same rotation) as

$$\bar{f}^{\alpha}(\alpha) = \left\{ \begin{array}{c} \frac{1}{\sqrt{1+\left(\frac{\tan\frac{\theta}{2}}{\cos\alpha}\right)^2}} \\ \frac{\frac{\tan\frac{\theta}{2}}{\cos\alpha}}{\sqrt{1+\left(\frac{\tan\frac{\theta}{2}}{\cos\alpha}\right)^2}} \left(\hat{\mathbf{x}}\cos\alpha + \hat{\mathbf{y}}\sin\alpha\right) \end{array} \right\} \tag{5.37}$$

which can simplify to a new definition

$$\bar{f}^{\alpha}(\alpha) = \frac{\bar{x}\cos\alpha + \bar{y}\sin\frac{\theta}{2}\sin\alpha}{\sqrt{\cos^2\alpha\cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}}} \tag{5.38}$$

for any real α. Equation 5.38 indexes the set of acceptable rotations based on α, which defines the axis of rotation.

5.3.3 Minimum Slew from Arbitrary Quaternion

The formulations expressed in Equations 5.18, 5.24, 5.26, 5.31, and 5.38 provide several ways to span the set of unit quaternions that align the provided vectors. However, this set is most useful if related to an arbitrary unit quaternion.

Here, we will identify which quaternion in the set is closest (using our distance metric in Equation 3.14) to an input quaternion. For example, if the input quaternion is the current orientation of a spacecraft, there is a whole set of potential orientations $\hat{\mathbf{v}}$ with $\hat{\mathbf{u}}$, and one orientation in that set is the smallest rotational distance from the input.[2] These formulae would return the orientation in that set which is the nearest to (minimum angle of a single-axis rotation between it and) the current orientation. We will also derive the distance between the input quaternion and its closest match, or the minimum distance between the input quaternion and the set. In the process, we will find several other extrema. Identifying rotations that are a maximal angle away could be useful in some control operations, such as avoiding pointing a camera towards the sun.

Let our arbitrary input unit quaternion be $\bar{p} = p_0 + \mathbf{p}_v$. We take our distance metric from Equation 3.14 and combine it with $\bar{p}$ and our expression from Equation 5.26. This gives us

$$\Delta\left(\bar{p}, \bar{f}^{\beta}(\beta)\right) = \arccos\left(2\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)^2 - 1\right). \tag{5.39}$$

By setting $\frac{d\Delta}{d\beta} = 0$, we can find the value of s that minimizes or maximizes the distance Δ. (As the domain of $\bar{f}^{\beta}(\beta)$ is unbounded, we do not need to check boundary points of β.) We see

$$\frac{d\Delta}{d\beta} = -\frac{4\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)\left(\bar{p}\cdot\frac{d\bar{f}^{\beta}(\beta)}{d\beta}\right)}{\sqrt{1-\left(2\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)^2-1\right)^2}}. \tag{5.40}$$

[2]There is the possibility of there being two or infinite equally-close points, but these are edge cases of the formulae derived here.

Table 5.3: Summary of Possibilities for Quaternion in Set Closest to $\bar{p}$.

Case	Solutions	Result
$\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)=0$	$\tan\beta_0=\frac{-\bar{p}\cdot\bar{x}}{\bar{p}\cdot\bar{y}}$	Orientations with maximal rotations from $\bar{p}$
$\left(\bar{p}\cdot\frac{d\,\bar{f}^{\beta}(\beta)}{d\,\beta}\right)=0$	$\tan\beta_1=\frac{\bar{p}\cdot\bar{y}}{\bar{p}\cdot\bar{x}}$	Orientations with minimal rotations from $\bar{p}$
$1-\left(2\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)^2-1\right)^2=0$	$\tan\beta_0=\frac{-\bar{p}\cdot\bar{x}}{\bar{p}\cdot\bar{y}},\ \bar{f}^{\beta}(\beta)=\pm\bar{p}$	Orientations with maximal rotations from $\bar{p}$, $\pm\bar{p}$ itself when $\bar{f}^{\beta}(\beta)$ contains it

Points to check include the values of β for which $\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)=0$ and $\left(\bar{p}\cdot\frac{d\,\bar{f}^{\beta}(\beta)}{d\,\beta}\right)=0$. Also, the singularities when

$$1-\left(2\left(\bar{p}\cdot\bar{f}^{\beta}(\beta)\right)^2-1\right)^2=0 \tag{5.41}$$

must also be considered. These cases are detailed in this section and are summarized in Table 5.3.

We first check the zero-values for $\frac{d\Delta}{d\beta}$. The first is when

$$\bar{p}\cdot\bar{f}^{\beta}(\beta)=0. \tag{5.42}$$

By putting this condition back into our metric, Equation 3.14, we see that when the case in Equation 5.42 occurs, the angle is $\Delta=\arccos(-1)=\pi$. Therefore, the values of β for which $\bar{p}\cdot\bar{f}^{\beta}(\beta)=0$ make $\bar{f}^{\beta}(\beta)$ lie π radians of a rotation from $\bar{p}$ and they are maximally distant. By plugging Equation 5.26 into $\bar{p}\cdot\bar{f}^{\beta}(\beta)=0$ and solving for β we see that this occurs when $\beta=\beta_0$ and β_0 satisfies

$$\tan\beta_0=\frac{-\mathbf{p}_v\cdot\hat{\mathbf{x}}\sin\frac{\theta}{2}-p_0\cos\frac{\theta}{2}}{\mathbf{p}_v\cdot\hat{\mathbf{y}}}=\frac{-\bar{p}\cdot\bar{x}}{\bar{p}\cdot\bar{y}} \tag{5.43}$$

where $\hat{\mathbf{x}}=\frac{\hat{\mathbf{v}}\times\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\|}$ and $\hat{\mathbf{y}}=\frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\|}$. Using Equation 5.28, the quaternion in the set that this corresponds to (that is π radians away from $\bar{p}$ and is maximally distant) is thus

$$\bar{f}^{\beta}(\beta_0)=\pm\frac{\bar{y}\left(\frac{-\mathbf{p}_v\cdot\hat{\mathbf{x}}\sin\frac{\theta}{2}-p_0\cos\frac{\theta}{2}}{\mathbf{p}_v\cdot\hat{\mathbf{y}}}\right)+\bar{x}}{\sqrt{1+\left(\frac{\mathbf{p}_v\cdot\hat{\mathbf{x}}\sin\frac{\theta}{2}+p_0\cos\frac{\theta}{2}}{\mathbf{p}_v\cdot\hat{\mathbf{y}}}\right)^2}} \tag{5.44}$$

or

$$\bar{f}^{\beta}(\beta_0)=\pm\frac{\bar{x}\left(\bar{p}\cdot\bar{y}\right)-\bar{y}\left(\bar{p}\cdot\bar{x}\right)}{\sqrt{\left(\bar{p}\cdot\bar{x}\right)^2+\left(\bar{p}\cdot\bar{y}\right)^2}}. \tag{5.45}$$

This works for almost any $\bar{p}$, which can be verified by trying $\bar{p}\cdot\bar{f}^{\beta}(\beta_0)$. If $\bar{p}$ is a member of the set $\bar{f}^{\beta}(\beta)$ such that $\left(\bar{p}\cdot\bar{x}\right)^2+\left(\bar{p}\cdot\bar{y}\right)^2=1$, we find that Equations 5.43 and 5.45 still work.

The exception is when $\bar{p} \cdot \bar{x} = \bar{p} \cdot \bar{y} = 0$. In that case, all quaternions in the set $\bar{f}^{\beta}(\beta)$ are maximally distant from $\bar{p}$ with distance π and there is no closest or furthest quaternion.

The other case to check is if $\left(\bar{p} \cdot \frac{d\,\bar{f}^{\beta}(\beta)}{d\,\beta}\right) = 0$. We differentiate Equation 5.28 and find

$$\frac{d\,\bar{f}^{\beta}(\beta)}{d\,\beta} = \bar{y}\cos\beta - \bar{x}\sin\beta. \tag{5.46}$$

Thus, our zero-value case to check is when

$$\bar{p} \cdot \bar{y}\cos\beta = \bar{p} \cdot \bar{x}\sin\beta \tag{5.47}$$

. This simplifies to

$$\tan\beta_1 = \frac{\bar{p} \cdot \bar{y}}{\bar{p} \cdot \bar{x}} = \frac{\mathbf{p}_v \cdot \hat{\mathbf{y}}}{p_0\cos\frac{\theta}{2} + \mathbf{p}_v \cdot \hat{\mathbf{x}}\sin\frac{\theta}{2}}. \tag{5.48}$$

When $\bar{f}^{\beta}(\beta)$ uses this value of $\beta = \beta_1$, it can be expressed as

$$\bar{f}^{\beta}(\beta_1) = \pm\frac{\bar{y}\left(\frac{\mathbf{p}_v \cdot \hat{\mathbf{y}}}{p_0\cos\frac{\theta}{2} + \mathbf{p}_v \cdot \hat{\mathbf{x}}\sin\frac{\theta}{2}}\right) + \bar{x}}{\sqrt{1 + \left(\frac{\mathbf{p}_v \cdot \hat{\mathbf{y}}}{p_0\cos\frac{\theta}{2} + \mathbf{p}_v \cdot \hat{\mathbf{x}}\sin\frac{\theta}{2}}\right)^2}} \tag{5.49}$$

or

$$\bar{f}^{\beta}(\beta_1) = \pm\frac{(\bar{p} \cdot \bar{x})\,\bar{x} + (\bar{p} \cdot \bar{y})\,\bar{y}}{\sqrt{(\bar{p} \cdot \bar{x})^2 + (\bar{p} \cdot \bar{y})^2}}. \tag{5.50}$$

As referenced in Equation 3.12, the $\pm$ sign is irrelevant, as these represent the same rotation. For the purposes of clarity, we will use the $+$ sign option, assuming that the angle of rotation is positive and between 0 and 2π. Care should be taken if this is used in a PID controller or another control algorithm, which may assume linearity and where a sign could be more significant. (Typically, this is dealt with by modifying the sign of the desired orientation quaternion based on the sign and value of the state orientation quaternion.)

The distance between this nearest quaternion in the set and $\bar{p}$ in radians is

$$\Delta = \arccos\left(-1 + 2\left((\bar{p} \cdot \bar{x})^2 + (\bar{p} \cdot \bar{y})^2\right)\right). \tag{5.51}$$

This is the minimum distance between $\bar{p}$ and the set $\bar{f}^{\beta}(\beta)$.

This works for almost any $\bar{p}$, which can be verified by trying $\bar{p} \cdot \bar{f}^{\beta}(\beta_1)$. If $\bar{p}$ is a member of the set $\bar{f}^{\beta}(\beta)$ (such that $(\bar{p} \cdot \bar{x})^2 + (\bar{p} \cdot \bar{y})^2 = 1$), we find that Equation 5.48 still works and Equation 5.50 simplifies to $\bar{f}^{\beta}(\beta_1) = \pm\bar{p}$. The exception is when $\bar{p} \cdot \bar{x} = \bar{p} \cdot \bar{y} = 0$. In that case, all quaternions in the set $\bar{f}^{\beta}(\beta)$ are maximally distant from $\bar{p}$ with distance π and there is no closest or furthest quaternion.

We also must check the singularities. Equation 5.41 can be simplified to

$$2\left(\bar{p} \cdot \bar{f}^{\beta}(\beta)\right)^2 - 1 = \pm 1. \tag{5.52}$$

This leaves two cases, which can be written as $\bar{p} \cdot \bar{f}^{\beta}(\beta) = 0$ and

$$\bar{p} \cdot \bar{f}^{\beta}(\beta) = \pm 1. \tag{5.53}$$

The former was already explored in the case relating to Equation 5.42. This is therefore a maxima, when $\bar{p}$ and $\bar{f}^{\beta}(\beta_0)$ are π radians apart. The nearest quaternion in the set can be found in Equation 5.45 and the corresponding value of $\beta = \beta_0$ can be found in Equation 5.43.

For the latter case, in Equation 5.53, the angle is $\Delta = \arccos(1) = 0$. Therefore, the values of β for which $\bar{p} \cdot \bar{f}^{\beta}(\beta) = \pm 1$ are those when $\bar{f}^{\beta}(\beta) = \pm\bar{p}$, meaning $\bar{p}$ and $\bar{f}^{\beta}(\beta)$ represent the same rotation. This would mean a distance of 0 radians, and these are thus minima. This can only happen when the set $\bar{f}^{\beta}(\beta)$ contains $\bar{p}$, which requires that $(\bar{p} \cdot \bar{x})^2 + (\bar{p} \cdot \bar{y})^2 = 1$. If that test is passed, then the solutions are a subset of those found in the case associated with Equation 5.47. The minimal β can be found by $\tan\beta_1 = \frac{\bar{p}\cdot\bar{y}}{\bar{p}\cdot\bar{x}}$, as in Equation 5.48. Similarly, Equation 5.50 still works and simplifies to $\bar{f}^{\beta}(\beta_1) = \pm\bar{p}$.

5.3.4 Slewing with "Keep-out" Zone

The nearest acceptable orientation to a given orientation, as laid out in Section 5.3.3, is useful for finding the minimum orientation change required to point a spacecraft. This can be used to find the appropriate slew from the arbitrary position, as shown in Section 5.3.5. However, the use of a set of orientations as a goal provides additional flexibility in motion planning, and allows for more complex goals to be met.

In this section, we consider single-axis slews that meet pointing goals, with the constraint of having a "keep-out" which some part(s) of the spacecraft may not point towards.

In addition to the previously-defined $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$ vectors, suppose that there is another unit vector $\hat{\mathbf{b}}$, defined in the body frame. This vector represents a direction in the satellite frame that we wish to avoid pointing in certain directions in the world frame. The simplest example would be an unprotected imager which we do not want to point at the sun. There is another unit vector $\hat{\mathbf{a}}$, defined in the world frame. This is the center of the "keep-out" zone that we wish o avoid pointing $\hat{\mathbf{b}}$ towards. Specifically, there is some angle η, and we do not want to bring $\hat{\mathbf{b}}$ within η of $\hat{\mathbf{a}}$.

As we are concerned not only with the final orientation but also with the entire slew, we need to consider the starting orientation as well. This could be done using spherical linear interpolation of quaternions (SLERP),[24], [116] but it is mathematically simpler to assume that the starting orientation of the spacecraft is the simple case, with $\bar{p} = \begin{Bmatrix} 1 & \mathbf{0}^T \end{Bmatrix}^T$, so the world and body frames are aligned at the start of the slew. (This approach can be applied to an arbitrary starting orientation $\bar{p}$ if the world-frame vectors ($\hat{\mathbf{u}}$, $\hat{\mathbf{a}}$) are first rotated by $\bar{p}^{-1}$ using Equation 3.10.)

First, we consider the space of acceptable orientations, $\bar{f}$. Due to our simple starting orientation, we know that the necessary slew $\bar{p}^{-1}\bar{f} = \bar{f}$. Therefore, we will use Equation 5.38, because the rotation variable selects the axis of rotation, which will be helpful in this problem.

Our axis of rotation will be $\hat{\mathbf{k}}(\alpha) = \hat{\mathbf{x}}\cos\alpha + \hat{\mathbf{y}}\sin\alpha$. From Equation 5.36, we know that the total angle of rotation will be $\gamma(\alpha)$, which is the solution to $\tan\frac{\gamma(\alpha)}{2} = \frac{\tan\frac{\theta}{2}}{\cos\alpha}$. During the slew, our rotation angle ρ will go from 0 to $\gamma(\alpha)$. Using Equation 3.13, we know that our orientation midway through the slew at angle ρ is

$$\bar{p}(\rho) = \begin{Bmatrix} \cos\frac{\rho}{2} \\ \hat{\mathbf{k}}(\alpha)\sin\frac{\rho}{2} \end{Bmatrix}. \tag{5.54}$$

We can ensure that the "keep-out" zone is respected if

$$\hat{\mathbf{a}} \cdot \left(\bar{p}^{-1}(\rho) \circ \hat{\mathbf{b}} \right) \leq \cos \eta.$$

From Equations 3.10 and 5.54, and trigonometric identities, this equation simplifies to

$$\begin{aligned} \cos \eta \geq & \left(\hat{\mathbf{a}} \cdot \hat{\mathbf{k}} \right) \left(\hat{\mathbf{b}} \cdot \hat{\mathbf{k}} \right) \\ & + \left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) \cos \rho - \left(\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}} \right) \sin \rho. \end{aligned} \tag{5.55}$$

Our slew options, indexed by α, are only acceptable if Equation 5.55 is true for all $0 \leq \rho \leq \gamma(\alpha)$. We first check the boundaries, when $\rho = 0$ and $\rho = \gamma(\alpha) = \pi - 2 \arctan(\sin \alpha \cot \frac{\theta}{2})$.

With $\rho = 0$, we have

$$\left(\hat{\mathbf{a}} \cdot \hat{\mathbf{k}} \right) \left(\hat{\mathbf{b}} \cdot \hat{\mathbf{k}} \right) + \left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \leq \cos \eta. \tag{5.56}$$

With $\rho = \pi - 2 \arctan(\sin \alpha \cot \frac{\theta}{2})$, the equation simplifies to

$$\frac{\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}}{2} - \frac{\left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) + \left(\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}} \right) \sin \alpha \cot \frac{\theta}{2}}{1 + \sin^2 \alpha \cot^2 \frac{\theta}{2}} \leq \frac{\cos \eta}{2}. \tag{5.57}$$

The other cases that could violate the keep-out zone are the values of ρ which maximize the left-hand side of Equation 5.55. As that equation is of the form $a + b \cos x + c \sin x$ (which, confined to $0 \leq x < 2\pi$, has its maximum at $\operatorname{atan2}(c, b)$), we know our maxima are at

$$\rho_{\max} = \operatorname{atan2}\left(\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}}, \left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) \right) \mod 2\pi \tag{5.58}$$

with the maximum values at $\rho = \rho_{\max}$ simplifying to

$$\begin{aligned} \cos \eta \geq & \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} - \left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) \\ & + \sqrt{\left(\left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) \right)^2 + \left(\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}} \right)^2}. \end{aligned} \tag{5.59}$$

For the trajectory to be acceptable, Equations 5.56 and 5.57 must be satisfied, as well as either Equation 5.59 or $\rho_{\max} > \gamma(\alpha) = \pi - 2 \arctan\left(\sin \alpha \cot \frac{\theta}{2} \right)$, which simplifies to

$$\begin{aligned} \sin \alpha > & \tan \frac{\theta}{2} \frac{\sqrt{\left(\left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right) \right)^2 + \left(\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}} \right)^2}}{\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}}} \\ & + \tan \frac{\theta}{2} \frac{\left(\hat{\mathbf{a}} \times \hat{\mathbf{k}} \right) \cdot \left(\hat{\mathbf{b}} \times \hat{\mathbf{k}} \right)}{\hat{\mathbf{k}} \cdot \hat{\mathbf{a}} \times \hat{\mathbf{b}}}. \end{aligned} \tag{5.60}$$

Thus, the values of α which satisfy both Equations 5.56 and 5.57 and either Equation 5.59 or Equation 5.60 are acceptable. These equations can be theoretically expanded and simplified before solving for α, but graphical or numerical solutions are simpler. Example code is included in the GitHub Repo at https://github.com/patrickmckeen/quaternion-pointing.

5.3.5 Application Cases

The simplest example case using this formulation is to point a spacecraft at a target using the minimum possible slew. To accomplish this, we use Equation 5.50.

5.3.5.a Basic Minimum Slew

As an example, consider a satellite with a directed antenna with a boresight along the body x-axis which needs to point towards a ground station, located in the direction of $\frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & -1\end{bmatrix}^T$ in the global ECI coordinate frame. We will consider two values for the satellite's current orientation: simple case $\bar{p}_1 = \begin{Bmatrix}1 & \mathbf{0}^T\end{Bmatrix}^T$ and the more complicated $\bar{p}_2 = \frac{1}{5}\begin{Bmatrix}4 & 1 & -2 & 2\end{Bmatrix}^T$.

With this information, we have $\hat{\mathbf{v}} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ and $\hat{\mathbf{u}} = \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & -1\end{bmatrix}^T$. Using Equations 5.1, 5.20, and 5.21, we first find $\theta = \arccos 0 = \frac{\pi}{2}$,

$$\bar{x} = \left\{\begin{array}{c}\frac{1}{\sqrt{2}} \\ \frac{1}{2}\begin{bmatrix}0\\1\\1\end{bmatrix}\end{array}\right\} \quad \text{and} \quad \bar{y} = \left\{\begin{array}{c}0 \\ \frac{1}{2}\begin{bmatrix}\sqrt{2}\\1\\-1\end{bmatrix}\end{array}\right\} \tag{5.61}$$

The set of orientations which will point $\hat{\mathbf{v}}$ along $\hat{\mathbf{u}}$ are thus

$$\bar{f}^{\beta}(\beta) = \left\{\begin{array}{c}\frac{1}{\sqrt{2}}\cos\beta \\ \frac{1}{2}\begin{bmatrix}\sqrt{2}\sin\beta \\ \sin\beta + \cos\beta \\ \cos\beta - \sin\beta\end{bmatrix}\end{array}\right\}. \tag{5.62}$$

We are now ready to consider our two cases. First, the simple case, with $\bar{p}_1 = \begin{Bmatrix}1 & \mathbf{0}^T\end{Bmatrix}^T$. Using Equation 5.50 (and taking the positive branch, without loss of generality), we find that the acceptable orientation closest to $\bar{p}_1$ is

$$\frac{\bar{x}\left(\frac{1}{\sqrt{2}}\right) + \bar{y}\left(0\right)}{\sqrt{\left(\frac{1}{\sqrt{2}}\right)^2 + 0^2}} = \bar{x}. \tag{5.63}$$

The associated β is $\beta = 0$. The rotation required to get there (from the current orientation) is

$$\bar{p}_1^{-1}\bar{x} = \bar{x} = \left\{\begin{array}{c}\frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{2} \\ \frac{1}{2}\end{array}\right\}. \tag{5.64}$$

which, using Equation 5.51, has a rotation angle of $\frac{\pi}{2}$ radians and is about the axis $\frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & 1\end{bmatrix}^T$ in both the body and global frame.

We now consider the more complicated case, with $\bar{p}_2 = \frac{1}{5}\begin{Bmatrix}4 & 1 & -2 & 2\end{Bmatrix}^T$. Using Equation 5.50 once more, we find that the acceptable orientation closest to $\bar{p}_2$ is

$$\frac{\bar{x}\left(\frac{2\sqrt{2}}{5}\right) + \bar{y}\left(\frac{\sqrt{2}-4}{10}\right)}{\sqrt{\left(\frac{2\sqrt{2}}{5}\right)^2 + \left(\frac{\sqrt{2}-4}{10}\right)^2}} = \frac{4\bar{x} + (1-2\sqrt{2})\bar{y}}{\sqrt{25-4\sqrt{2}}}. \tag{5.65}$$

The associated β is $\beta \approxeq -1.142$. The rotation required to get to one (from the current orientation) is

$$\bar{p}_2^{-1}\left(\frac{4\bar{x}+(1-2\sqrt{2})\bar{y}}{\sqrt{25-4\sqrt{2}}}\right) = \frac{1}{10\sqrt{25-4\sqrt{2}}}\begin{Bmatrix}25\sqrt{2}-8\\0\\31\\15\end{Bmatrix}. \tag{5.66}$$

which, using Equation 5.51, has a rotation angle of $\arccos\left(\frac{-4\sqrt{2}}{25}\right) \approx 1.799$ radians and is about the axis $\frac{1}{\sqrt{1186}}\begin{bmatrix}0 & 31 & 15\end{bmatrix}$ in the body frame.

5.3.5.b Keep-out Zone

The ability to use a variety of goal orientations can be useful for other trajectory goals. Consider the same trajectory problem, with a "keep-out" zone. The imager boresight is $\hat{\mathbf{b}} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$. Finally, assume the sun is in the direction of $\hat{\mathbf{a}} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & \sqrt{3}/2 & 1/2\end{bmatrix}^T$ in the global ECI coordinate frame, and there is a $\eta = 30°$ "keep-out" zone for pointing the imager towards the sun.

Our axis of rotation will be $\hat{\mathbf{k}}(\alpha) = \hat{\mathbf{x}}\cos\alpha + \hat{\mathbf{y}}\sin\alpha$, where

$$\hat{\mathbf{x}} = \frac{\hat{\mathbf{v}}\times\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\|} = \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\\1\end{bmatrix} \quad \text{and} \quad \hat{\mathbf{y}} = \frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\|} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{2}\\-\frac{1}{2}\end{bmatrix}. \tag{5.67}$$

The angle of rotation is $\gamma(\alpha) = \pi - 2\arctan(\cos\alpha\cot\frac{\theta}{2})$, or $\gamma(\alpha) = \pi - 2\arctan(\cos\alpha)$.The necessary equations for our trajectory to satisfy are Equations 5.56 and 5.57 and either Equation 5.59 or 5.60 (or both). We now consider these for our vectors.

These are best solved graphically or numerically, as in the code in the associated GitHub repo. Figure 5.2 shows the plot of the relevant inequality bounds for this case. Eqs. 5.56 and 5.57 are shown as solid lines, both of which must be less than zero, as the rotation cannot violate the sun-pointing constraint at the start or end of the slew. Eqs. 5.59 and 5.60 are shown as dash-dot lines, and at least one must be less than zero, meaning that either the most-sun-pointing part of the rotation is still outside our limit or that this extreme point will not occur within the interval of the slew (or both). With these constraints, Figure 5.2 shows the acceptable values for α are $0.48 \lessapprox \alpha \lessapprox 0.66$ and $1.74 \lessapprox \alpha \lessapprox 3.60$. Note that the minimum slew is not in the acceptable rotation set, because $\rho_{\max} < \gamma$ (the point on the rotation that most aligns the sun and imager is inside the range of this slew) and at $\rho = \rho_{\max}$, the keep-out zone is violated (that most-aligned point is too close for our 30° limit). In Figure 5.2, this can be seen by noting that both the magenta and purple dash-dot lines are greater than zero.

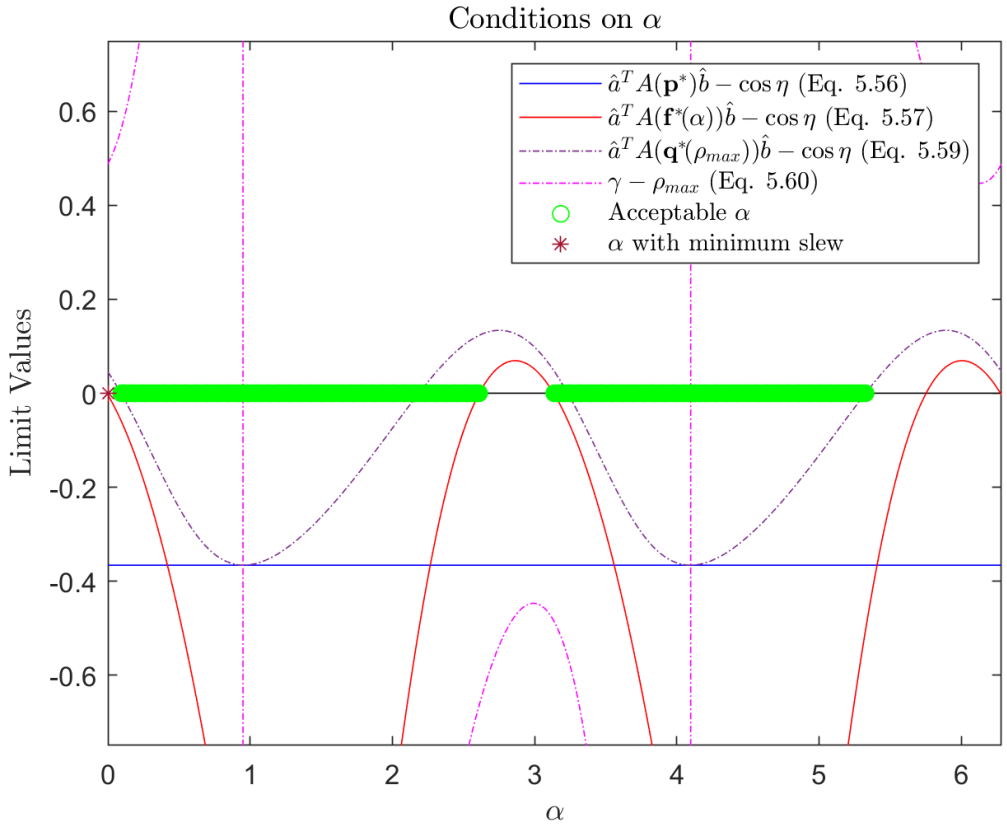


Figure 5.2: Plot of constraints that must be satisfied to avoid keep-out zone. Lines represent inequality bounds: both of the solid lines and one or more of the dash-dot lines must be negative for acceptable rotations. The lines are unitless, representing inequalities which are functions of of α (shown in Equation 5.38), which is the angle between potential rotation axes $\hat{\mathbf{x}} = \frac{\hat{\mathbf{v}}\times\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}\times\hat{\mathbf{u}}\|}$ and $\hat{\mathbf{y}} = \frac{\hat{\mathbf{v}}+\hat{\mathbf{u}}}{\|\hat{\mathbf{v}}+\hat{\mathbf{u}}\|}$). For a given rotation (indexed by α) to be acceptable, the corresponding slew must not intersect the keep-out zone at the beginning (true if the blue line is less than zero, Equation 5.56), ending (true if the red line is less than zero, Equation 5.57), or closest point (true if one or both of the purple and pink dot-dash lines, Equations 5.59 and 5.60, are less than zero). Green circles (forming a solid thick line) mark acceptable values of α for which this is true, and a dark red star indicates the acceptable value of α with the minimum slew.

5.4 Modifying and Generalizing Control Laws

One difficulty in approaching this issue is using various published control laws on satellites with different properties or different goals than the case for which they were originally published. Sometimes this is for good reason–the algorithms are specialized and are effective primarily due to mission-specific details (such as a long, thin satellite relying partially on gravity gradient to remain in nadir-zenith alignment). However, many control laws should (at least in theory) be generalizable to alternate orientations and different sets of actuators. Adjusting the control laws, however, has often been more difficult than expected. Here are a variety of modifications that are required to make control laws more general. For reference, consider a basic control law that generates a desired torque $\boldsymbol{\tau}$ via a command $\mathbf{u}$, for the actuators based on the system state $\mathbf{x}$ and goal $\mathbf{x}_g$

$$\boldsymbol{\tau} = f\left(\mathbf{x}, \mathbf{x}_g\right) = g\left(\mathbf{u}, \mathbf{x}\right), \tag{5.68}$$

where f represents the control law formulation and g represents the system's model of its own dynamics. Many control laws specifically have a value for $\boldsymbol{\tau}$ that they find in their calculations, which may then be adjusted to meet constraints, add terms (such as disturbance countering or an integral term), and be applied to each actuator. This allows for each modification. More rarely, control laws do not have a step which finds the desired torque and instead arrive at a command for each actuator independently. To modify such a control law, propagate the control law's command through the expected dynamics of the system via g, and this will describe the "desired" torque. The other adjustments in this section can then be applied.

5.4.1 Adjusted Command

Perhaps the simplest adjustment that can be made is to the command, adjusting it to directly counter a known (or estimated, or generalized) bias. In that case, the command is adjusted by the expected bias, β, giving

$$\mathbf{u}' = \mathbf{u} - \beta. \tag{5.69}$$

5.4.2 Adjusted Torque

A similar adjustment can be made to the control torque to directly counter a known (or estimated, or all-in-one) disturbance, as discussed further in Chapter 6. In that case, the control is adjusted by the expected disturbance to give

$$\boldsymbol{\tau}' = \boldsymbol{\tau} - \boldsymbol{\tau}_{\text{disturbance}}. \tag{5.70}$$

Care should be taken not to double-up on disturbance countering, such as through an integral term, or when used in a system whose stability, passively or actively, relies on a specific disturbance.

Some control laws, such as PD control, do not directly account for the evolution of the system. To speed convergence on these (and deal with weird dynamic effects), some of the system's nonlinear dynamics can be accounted for directly as a "disturbance torque." This is

valid because the rotational nonlinearities are added directly to torque in the dynamics (see Equation 2.2) and can thus be directly subtracted off. This gives

$$\boldsymbol{\tau}' = \boldsymbol{\tau} + \boldsymbol{\omega} \times J\boldsymbol{\omega}, \tag{5.71}$$

where ω is the body-frame angular velocity and J is the spacecraft body-frame inertia matrix. If N reaction wheels are present, the control law modifies to account for the additional angular momentum, giving

$$\boldsymbol{\tau}' = \boldsymbol{\tau} + \boldsymbol{\omega} \times \left(J_{\text{noRW}}\omega + \sum_{i=1}^{N_{\text{act}}} \hat{\alpha}_{\text{act},i} h_i \right), \tag{5.72}$$

where J_{noRW} represents the inertia matrix without the axis-aligned components of the reaction wheels, and $\hat{\alpha}_{\text{act},i}$ and h_i represent the i-th wheel's axis and stored angular momentum.

This should not be applied if the control law already counters this effect.

5.4.3 Momentum Desaturation

A method to constantly desaturate reaction wheels (including when they are redundant) via onboard magnetorquers is described in [127]. Specifically, if $\mathbf{u}_{\text{mtq}}$ is the command of magnetorquers and $\mathbf{u}_{\text{rw}}$ is the command for reaction wheels, the command is adjusted as

$$\Delta h = c\left(\mathbf{h} - \mathbf{h}_g\right) \tag{5.73}$$

$$\Delta \mathbf{u}_{\text{mtq}} = \left(\mathbf{B}^{\times} A_{\text{mtq}}\right)^{+} A_{\text{rw}} \Delta h \tag{5.74}$$

$$\Delta \mathbf{u}_{\text{rw}} = \Delta h + A_{\text{RW}}^{+}\left(\mathbf{B}^{\times} A_{\text{mtq}} \Delta \mathbf{u}_{\text{mtq}} - A_{\text{RW}} \Delta h\right) \tag{5.75}$$

$$\mathbf{u}'_{\text{mtq}} = \mathbf{u}_{\text{mtq}} + \Delta \mathbf{u}_{\text{mtq}} \tag{5.76}$$

$$\mathbf{u}'_{\text{rw}} = \mathbf{u}_{\text{rw}} + \Delta \mathbf{u}_{\text{rw}}, \tag{5.77}$$

where c is a gain controlling the rate of momentum dumping, $\mathbf{h}$ is the vector of stored angular momentum on each reaction wheel, $\mathbf{h}_g$ is the "desired" stored angular velocity,[3] $\mathbf{B}$ is the local external magnetic field expressed in the body frame, $A_{\text{mtq}} = \begin{bmatrix}\hat{\alpha}_{\text{mtq},1} & \hat{\alpha}_{\text{mtq},2} & \dots & \hat{\alpha}_{\text{mtq},N_{\text{mtq}}}\end{bmatrix}$ is the matrix composed of the unit-vector directions for the magnetorquers, $A_{\text{RW}} = \begin{bmatrix}\hat{\alpha}_{\text{RW},1} & \hat{\alpha}_{\text{RW},2} & \dots & \hat{\alpha}_{\text{RW},N_{\text{RW}}}\end{bmatrix}$ is the same for the reaction wheels, M^{+} is the Moore-Penrose pseudoinverse of M, and $\mathbf{v}^{\times}$ is the skew-symmetric matrix of vector $\mathbf{v}$.

This approach can be extended somewhat to account for other actuators or to allow a specific input torque. Such an approach is expanded in the next subsection, Section 5.4.4.

5.4.4 Command from Torque

A simple generalization of the Hogan-Schaub approach can be thought of as minimizing

$$\frac{1}{2}\mathbf{u}^T W \mathbf{u} - (\mathbf{h} - \mathbf{h}_g)^T W_{\text{momentum}} u_{\text{RW}}, \ \mathbf{u} = \begin{bmatrix} u_{\text{RW}} \\ u_{\text{mtq}} \\ u_{\text{other}} \end{bmatrix} \tag{5.78}$$

[3] This should not be zero to avoid stiction.

subject to

$$\boldsymbol{\tau} = M\mathbf{u},\ M = \begin{bmatrix} A_{\text{RW}} & \mathbf{B}^{\times} A_{\text{mtq}} & G_{\text{other}} A_{\text{other}} \end{bmatrix} \tag{5.79}$$

where u_{other} represents any additional control method, A_{other} represents their axes, and $G_{\text{other}} = \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{u}_{\text{other}}}$ is a matrix representing the (linearized) effect of each of the "other" actuators on torque. Additionally, W is a symmetric, positive-definite matrix describing the comparative cost of using each actuator and W_{momentum} describes the relevant cost of storing the angular momentum–adjusting these values changes what the solution minimizes.

The solution to such a problem is

$$A = MW^{-\frac{1}{2}} \tag{5.80}$$

$$\mathbf{u}' = W^{-\frac{1}{2}} \left(A^{+}\boldsymbol{\tau} + \left(I - A^{+}A\right) W^{-\frac{1}{2}} \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix} W_{\text{momentum}} (\mathbf{h} - \mathbf{h}_g) \right) \tag{5.81}$$

.

This could be extended further to better capture the desaturation requirement, but generally applies to an arbitrary set of actuators that have linear torque.

5.4.5 Bounds

The methods described in this section can find a solution that works for all sets of actuators. However, they can lead to desired commands that are not possible for the satellite. There may be constraints–equality constraints can be countered as shown in Section 5.4.4, but nonlinear constraints do not permit an analytical solution. The most likely constraint to be encountered is bounds–maximum and minimum values on each actuation effort.

This can be described as a minimization problem

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T W \mathbf{u} - (\mathbf{h} - \mathbf{h}_g)^T W_{\text{momentum}} u_{\text{RW}} + \gamma \left(\frac{1}{2} \left| M\mathbf{u} - \boldsymbol{\tau} \right|^2 \right) \\ &\text{s.t. } \ell_- \leq \mathbf{u} \leq \ell_+, \end{aligned}$$

where γ has a very large value.

The typical approach (for a small number of actuators) is to take the ideal solution and then clip the values, though this raises questions. If correcting for biases, should clipping take place after or before–and should the biases be included in the clipping? Should the clip preserve direction (for example, in an MTQ-only satellite) and thus clip so that the final control is parallel to the desired control, or should it clip each axis separately, minimizing the vector distance between the desired and final control? And is there a method to see if there is *any* value of $\mathbf{u}$ satisfying $M\mathbf{u} = \boldsymbol{\tau}$ within the desired bounds, even if the "ideal" value from Equation 5.81 does not? These choices should be further investigated, as well as how such a nonlinearity then combines with the other factors in this section–for example, is it better to fully correct for disturbance and clip the baseline desired torque?

This approach could be extended to work with other nonlinear constraints, such as limits on the next-step stored angular momentum (when gain/weight selection cannot fully avoid it).

5.4.6 Thoughts on Gains and Other Parameters

Further exploration is needed, but control torque overall should scale with J. The aspect that corresponds to angular correction (as opposed to angular velocity/momentum/dynamics correction) should scale with a fraction of this for stability–perhaps $\sqrt{J}$, or not at all.

5.4.7 Goal-Selection on a Set

One of the most complex modifications is to modify an orientation-based control law–one that is attempting to control the system towards a specific quaternion q_g–to instead attempt to align two vectors, such as the pointing goals in Section 5.2. There are control laws that do this already[105], but a general implementation for other laws is difficult.

The most straightforward approach is to take the orientation-based control laws and to use them as-is, but adjust the goal, q_g, to some value on the quaternion goal set of the type described in Section 5.3. For some laws, there may also then need to be an adjustment to the controlled angular velocity $\boldsymbol{\omega}_g$, to account for the rate at which q_g changes. The question, then, is the method by which to select the goal amidst the set. There are several strategies to do so, though the efficacy of the method depends on the control method used (sliding surface control, for example, has more complexity than PD). These strategies include:

- Aim for the nearest quaternion–select β for the set of goals so that the angular distance between q and q_g is minimized. This is simple, but does not take advantage of existing dynamics, such as angular velocity or any constraints.
- Create a Lyapunov function and select β such that the predicted rate of change will be minimized–this could include scaling it with the current Lyapunov score
- Select something that minimizes distance to some internal variable–such as the surface distance $\mathbf{s}$ in a sliding surface problem
- Take advantage of the variability to deal with constraints. For example, with magnetic control, select β such that the torque that is actually expected to be generated from the magnetic field is as close to the ideal desired torque as possible.

5.5 Conclusion

This chapter presents a framework for generalizing ADCS goals and control laws, addressing the critical challenges of adaptability, precision, and scalability in satellite operations. Beginning with the goal specification detailed in Section 5.2, the chapter introduces a versatile method for defining time-varying pointing goals that can adapt to dynamic mission requirements. These goals are represented in a mathematical structure that supports a wide range of operational scenarios, such as aligning a satellite's orientation with specific celestial or terrestrial targets, specific orbital frame directions (such as zenith or anti-ram), or specified vectors in various frames. By using these reduced goals (when appropriate), we expand the traditional notion of pointing goals to encompass broader use cases, including vector alignment and goal sets, which are vital for missions requiring simultaneous multi-objective

operations and can allow for more versatile and lower-effort solutions. The derivation of these goal sets and their use to identify slews that are closest or respect keep-out zones is described in Section 5.3.

Section 5.4 extends this adaptability by generalizing control laws to accommodate diverse satellite configurations and operational conditions. This is achieved through the introduction of dynamic adjustments to control strategies, such as multiple actuators and momentum desaturation, ensuring robust performance across many types of spacecraft, including those that are underactuated. The incorporation of disturbance-aware adjustments allows the framework to handle real-world challenges like torque saturation and misalignment errors, addressing critical gaps in traditional ADCS systems. These advancements enable satellites to autonomously transition between goals and adapt to changes in their environment or behavior, significantly reducing the need for manual intervention.

Despite these advancements, several areas of future work remain. Enhancing the optimization methods for goal prioritization under complex constraints, such as dynamically changing environments or overlapping operational requirements, could further improve the framework's flexibility. Expanding the range of constraints, such as non-linear keep-out zones or time-dependent obstacles, would also increase its applicability to more complex missions. Additionally, the integration of high-performance computational methods or hardware-in-the-loop testing would bridge the gap between theoretical validation and real-world implementation. Incorporating machine learning techniques could further enhance adaptability by enabling the system to learn from historical data and predict future disturbances or operational needs.

The concepts and work laid out in this chapter tie to the key components of this thesis: starting from the areas of work in Section 3.2, it builds the contributions in Section 3.3 to address the technical gaps in Section 2.3.2. Much of this chapter's content is a direct output of the **ADCS Operations and Goals** area of work (Section 3.2.2). By specifically formalizing goals (allowing adaptation between goal types) and generalizing control laws, the chapter also stems from the **Generalized ADCS** area of work (Section 3.2.5), which helps address the **General ADCS Approach** technical gap (Section 2.3.2.a). This is part of the foundation for the **ADCS Architecture for Weak or Underactuated Satellites** contribution (Section 3.3.5). It also pays dividends by simplifying work in other areas. The formalization of goal types lays the foundation for the **Trajectory Planning** contribution (Section 3.3.3) described primarily in Chapter 7, enabling seamless integration between goal specification, control, and long-term planning, helping to address the **ADCS Trajectory Planning** technical gap (2.3.2.c). By modifying control laws to account for goal sets or reduced goals and by including methods to counter disturbances or balance between actuators, it also leads to improved magnetic control (addressing the **Effective Magnetic Control** technical gap in Section 2.3.2.b). Lastly, by increasing the solution space with goal sets and modifying control laws to allow for real-time adaptation for things like biases, disturbances, desaturation, and balancing between actuators, this is part of the **Improved Autonomy for ADCS** contribution (Section 3.3.4) and makes progress on the **Autonomous Rapid Retargeting** technical gap (Section 2.3.2.d). This work represents a significant step forward in achieving a generalizable, autonomous, and adaptable ADCS, providing robust solutions for the challenges of modern satellite operations and paving the way for future advancements in multi-mission satellite frameworks.

Chapter 6

Active Disturbance Rejection

6.1 Introduction

Satellite attitude control is critical for missions requiring precise pointing, such as directing solar panels at the sun, imagers at a target, or radios at ground stations. Achieving these goals involves controlling the system's state–typically attitude and angular velocity or momentum–and rejecting disturbances from both the spacecraft and its environment. Disturbances, if left unaddressed, can disrupt the state and degrade performance. While control strategies vary widely–ranging from passive stabilization to active methods using reaction wheels, magnetorquers, thrusters, or control moment gyroscopes–they are typically tailored to mission-specific requirements, including goals, orbit, actuation methods, and spacecraft properties.

At their core, all attitude control methods share a fundamental objective: rejecting disturbances that apply unwanted torques to the spacecraft. These disturbances arise from diverse sources, including atmospheric drag, solar radiation pressure, the spacecraft's residual magnetic dipole interacting with external magnetic fields, and gravity gradient effects. Many active attitude control approaches treat disturbances effectively as random noise, relying on feedback control laws to mitigate their impact. Less commonly, satellite attitude control systems have incorporated disturbances into their stability calculations and even their control modes. For example, missions like GOCE [49], [131] have used drag as a stabilizing influence, and others have relied on permanent magnets or the gravity gradient [39], [51], [53]–[56].

Building on this foundation, this work explores an alternative approach: modeling disturbances as part of the dynamic system state and incorporating them directly into control laws. This method includes disturbances as feedforward terms, enhancing the control law's predictive capability. By doing so, it bridges the gap between traditional feedback-dominated strategies and more adaptive systems that explicitly account for environmental forces. This approach stems directly from the **Direct Countering of Disturbances** (Section 3.2.3) area of work and is highly generalizable across spacecraft, missions, and disturbances, connecting it to the **Generalized ADCS** (Section 3.2.5) area of work as well. Unlike techniques such as system identification (SysID) or adaptive control, which focus on refining models or dynamically adjusting control responses, this method emphasizes real-time estimation of disturbance dynamics. By modeling disturbances and integrating them into

the estimation process, it continuously adapts to changing conditions without requiring explicit parameter adjustments or control law modifications. It delivers the **Disturbance Modeling for Control Inclusion** contribution (Section 3.3.2), and its flexibility and utility for underactuated satellites (as explored thoroughly in this chapter) connect it to the **Improved Autonomy for ADCS** (Section 3.3.4) and **ADCS Architecture for Weak or Underactuated Satellites** (Section 3.3.5) contributions.

The primary efforts of this chapter include adjusting (as described in Section 5.4) existing control laws to incorporate disturbances and demonstrating their performance improvements, even under significant environmental variability. Several existing control laws, described in Section 6.2, are considered: a simple proportional-derivative (PD) control law for thrusters [42], a PD control law for magnetic control [59], and a sliding surface controller for magnetic actuation [128]. Building on the satellite modeling framework in [124], these control laws are modified to account for disturbances. Their performance with and without disturbance modeling is shown. Additionally, the estimation method (see Chapter 4) is used to generate an "all-in-one disturbance" torque, described in Section 6.3, rather than modeling each one specifically, and control laws incorporating the all-in-one disturbance torque are also considered. The performance of the anti-disturbance control laws (with both fully-specified models and the all-in-one disturbance term) is compared against the original laws in a disturbed environment in Section 6.5. A summary of these test cases can be found in Table 6.1.

Table 6.1: Summary of Disturbance Control Test Cases

Test Case	**Satellite Details**	**Section**	**Purpose**	**Results and Notes**
Wie Comparison	Space Shuttle	6.5.1.a	Verify implementation of Wie's control law and assess disturbance inclusion	Disturbance inclusion has a small but measurable effect. Errors remain comparable across cases. Implementation verified.
Lovera Comparison	Lovera Test Case	6.5.1.b	Verify implementation of Lovera's magnetic PD control law and evaluate effect of disturbance handling	Implementation verified. Explicit disturbance modeling reduces errors to $< 1°$. All-in-one estimate stabilizes between 0.5–2°.

Continued on next page

Table 6.1 Continued

Test Case	Satellite Details	Section	Purpose	Results and Notes
Wisniewski Comparison	Ørsted	6.5.1.c	Verify implementation of Wisniewski's sliding mode control law and determine the impact of disturbances	Implementation verified. Disturbance-aware case stabilizes errors to $< 3°$. Ignoring disturbances leads to errors of 10–60°.
Modified Sliding Test	Ørsted	6.5.1.d	Assess robustness of modified sliding surface control law	Errors stabilize at 2–8° (disturbance-aware) and 3–6° (all-in-one). Disturbed case stabilizes at 10–50°.
Lovera CubeSat Test	3U CubeSat	6.5.2.b	Assess Lovera's control law on a CubeSat platform	Errors stabilize to $< 5°$ with disturbance inclusion. All-in-one estimate performs comparably.
Wisniewski CubeSat Test	3U CubeSat	6.5.2.a	Evaluate Wisniewski's sliding law performance for CubeSat case	Explicit modeling achieves $< 4°$. Disturbed case fails (errors $> 180°$).
Modified Sliding CubeSat Test	3U CubeSat	6.5.2.c	Test modified sliding surface control law on CubeSat under disturbances	Errors stabilize to 2–8° (disturbance-aware) and 3–6° (all-in-one). Disturbed case "stabilizes" at 10–50°.
Sliding versus Modified Comparison	Ørsted, 3U CubeSat	6.5.3	Compare Wisniewski and modified sliding surface control laws.	Wisniewski's law achieves the best accuracy ($< 3°$), but the modified surface shows improved robustness to unmodeled disturbances and different platforms.

6.2 Comparison Cases

Three published control laws, and the test cases from the original publications, are selected for comparison. The notation between works differs, but we use a consistent notation for all three. Two cases feature control laws that control the satellite to the orientation $q = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$. The other case controls to this orientation as well, but in an orbit-based frame with the z-axis pointing zenith and the x-axis pointing to the orbit normal.

All control laws are slightly modified to account for disturbance torques, as described in the following subsections, as well as actuator biases.[1] Actuator biases are implemented by taking the control law output $\mathbf{u}$ and creating the command $\mathbf{u}_{\text{out}} = \text{clip}(\mathbf{u} - \beta)$, where β are the actuator biases (estimated by the system estimator), and the clipping function is applied axis-by-axis.

6.2.1 Proportional-Derivative (Idealized) Thruster Control

This control law comes from [42], Equation 4. In that work, Wie and Barba create several control laws based on body frame angular velocity and angular error. Their control law is governed by a proportional gain K_p and a derivative gain K_d. With small changes to notation, the law is

$$\mathbf{u} = \boldsymbol{\tau} = -K_p\delta_{1:3} - K_d\boldsymbol{\omega}, \tag{6.1}$$

where δ is the difference between the current orientation and the desired orientation, both represented as quaternions, so

$$\delta = q_g^{-1}q, \tag{6.2}$$

with q_g as the goal quaternion. Additionally, $\mathbf{q}_{1:3}$ is the vector component of the quaternion q.

The law in Equation 6.1 is simply modified to include disturbance torque as

$$\mathbf{u} = \boldsymbol{\tau} = -K_p\delta_{1:3} - K_d\boldsymbol{\omega} - \boldsymbol{\tau}_{\text{dist}}. \tag{6.3}$$

The test case comes from the paper as well. A 28.5° inclination circular orbit at 450 km is used, with a satellite with an inertia matrix $J = \text{diag}\left(\begin{bmatrix}10000, & 9000, & 12000\end{bmatrix}\right)$ kgm^2. The control parameters are set to $K_d = 200I_{3\text{x}3}$ and $K_p = 50I_{3\text{x}3}$.

6.2.2 Proportional-Derivative Magnetic Control

The law applied in this section is from [59]. In that work, Lovera et al. create several magnetic PD control laws. We focus on the body-frame control law in Equation 21:

$$\tau_{\text{des}} = -J^{-1}\left(\epsilon^2 k_p\delta_{1:3} + \epsilon k_v\omega\right). \tag{6.4}$$

The variables k_p and k_v are the proportional and derivative gains, respectively, while ϵ is a scaling factor applied in the original text to make the system stable.

[1] Actuator axes are also accounted for, but that is not relevant in this work.

This equation produces the desired torque. However, these laws are for magnetically-controlled systems. Magnetically-controlled satellites cannot provide torque around the direction of the ambient B-field, and the control law must be converted to the magnetic moment applied to each torque coil. Fortunately, a simple approach can be used to achieve both. This is commonly used in magnetic attitude control and can be found in Equation 5 in [59]. The desired control torque τ_{des} is converted to a magnetic moment through the cross product formula

$$\mathbf{m}_{\text{coils}} = \frac{1}{|\mathbf{B}|^2}\mathbf{B} \times \boldsymbol{\tau}_{\text{des}}. \tag{6.5}$$

Modifying these equations to account for disturbance torque (and combining them) gives

$$\mathbf{m}_{\text{coils}} = \frac{1}{|\mathbf{B}|^2}\mathbf{B} \times (\boldsymbol{\tau}_{\text{des}} - \boldsymbol{\tau}_{\text{dist}}) = \frac{1}{|\mathbf{B}|^2}\mathbf{B} \times \left(-J^{-1}\left(\epsilon^2 k_p \delta_{1:3} + \epsilon k_v \omega\right) - \boldsymbol{\tau}_{\text{dist}}\right). \tag{6.6}$$

The test case comes from the paper as well. An 87° inclination circular orbit at 450 km is used, with a satellite with an inertia matrix $J = \text{diag}\left(\begin{bmatrix}27, & 17, & 25\end{bmatrix}\right)$ kgm^2. The control parameters are set to $k_p = k_v = 50$ and $\epsilon = 0.01$.

6.2.3 Sliding Scale Magnetic Control

The law applied in this section is from [128]. In that work, Wisniewski creates a magnetic control law based on the sliding surface. This control law is rather intensive and tailored to the application. The sliding surface used for the control law is based on an orientation goal, q_{des}, that varies with the orbit, defined in a right-hand frame with the z-axis pointing zenith and the x-axis pointing along the orbit normal. There is also a desired angular velocity to maintain this frame's rotation in the inertial frame throughout the orbit. We will call this ω_{des} and, in the global frame, it is equal to the angular velocity of the satellite around the Earth's center (we will treat it as being rotated into the body frame). In the original case described, with this motion and gravity gradient taken into account, the control law is described as follows:

The sliding surface is (Equation 7 in [128])

$$\mathbf{s} = J\left(\omega - \omega_{\text{des}}\right) + K_q \delta_{1:3}, \tag{6.7}$$

where K_q defines a constant of proportionality between the angular momentum error and the angular error.

This surface variable evolves according to (Equation 15 in [128])

$$\dot{\mathbf{s}} = J\left(\dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\omega}}_{\text{des}} - \boldsymbol{\omega} \times \boldsymbol{\omega}_{\text{des}}\right) + \frac{1}{2}K_q\left(\delta\left(\omega - \omega_{\text{des}}\right)\right)_{1:3}. \tag{6.8}$$

The first term is easily converted from (our) Equation 2.2. (It should be noted that in addition to the control torque $\boldsymbol{\tau}_{\text{ctrl}}$, Wisniewski already also includes gravity gradient torque, but it is not shown in this derivation–including disturbances has benefits!) Our other disturbance torques can also be added here. This gives us the sliding variable evolution as

$$\dot{\mathbf{s}} = -\boldsymbol{\omega} \times J\boldsymbol{\omega} + \boldsymbol{\tau}_{\text{ctrl}} + \boldsymbol{\tau}_{\text{dist}} - J\left(\dot{\boldsymbol{\omega}}_{\text{des}} + \boldsymbol{\omega} \times \boldsymbol{\omega}_{\text{des}}\right) + \frac{1}{2}K_q\left(\delta\left(\omega - \omega_{\text{des}}\right)\right)_{1:3}. \tag{6.9}$$

To keep forcing $\mathbf{s}$ towards zero, we set the desired control torque equal to a value that would keep $\mathbf{s}$ constant, plus a term (in this case, $K_s\mathbf{s}$, where K_s is a gain governing the decrease in the magnitude of $\mathbf{s}$) that forces it to have a decreasing magnitude (see Eqs. 17-18 in [128]). This gives us

$$\boldsymbol{\tau}_{\mathrm{s}} = -\boldsymbol{\tau}_{\mathrm{dist}} + \boldsymbol{\omega} \times J\boldsymbol{\omega} + J\left(\dot{\boldsymbol{\omega}}_{\mathrm{des}} + \boldsymbol{\omega} \times \boldsymbol{\omega}_{\mathrm{des}}\right) - \frac{1}{2}K_q\left(\delta\left(\omega - \omega_{\mathrm{des}}\right)\right)_{1:3} - K_s\mathbf{s}. \tag{6.10}$$

However, as in the Lovera case, this torque cannot be fully realized due to the magnetic control constraints. The same approach is applied, using Equation 6.6 with

$$\boldsymbol{\tau}_{\mathrm{des}} = \mathbf{s}\frac{\mathbf{s}^T\boldsymbol{\tau}_{\mathrm{s}}}{|\mathbf{s}|^2}. \tag{6.11}$$

The factors in the simulation are based on those in [128]. A 96° inclination orbit at 450 km is used, with a satellite inertia matrix $J = \mathrm{diag}\left(\begin{bmatrix}3.4278, & 2.9038, & 1.2750\end{bmatrix}\right)$ kgm^2. As in the source, the control parameters are set to $K_s = \mathrm{diag}\left(\begin{bmatrix}0.003, & 0.003, & 0.003\end{bmatrix}\right)$ and $K_q = \mathrm{diag}\left(\begin{bmatrix}0.002, & 0.002, & 0.002\end{bmatrix}\right)$. This sliding scale law is used in its own right, but it struggles with some cases. Specifically, it does not work consistently across different goals or when stabilizing disturbances are removed. A modified version is developed, described further in Section 6.4.

6.3 All-in-One Disturbance Torque

When including the disturbance torques in the control laws in Section 6.2, the more accurate, the better. However, many torques are difficult to estimate directly. The framework in [124] allows for estimating time-varying disturbances like a residual magnetic dipole or off-axis thrust from propulsion, but when including disturbances like gravity gradient, drag, and SRP, there is the risk of the estimated torques being incorrect due to the system's properties (like reflectance or inertia matrix) changing over time, or parameters being mis-modeled. To account for that, as well as to reduce computational complexity and include unknown disturbances, the estimation package laid out in [124] allows for creating a time-varying "all-in-one disturbance" that effectively acts as a feed-forward integral term in the control. Effectively, inside the estimator (square root unscented Kalman filter (SRUKF), in these cases, but applicable to others), the all-in-one disturbance is found from the residuals of the expected dynamics (based on estimated parameters, including biases) and estimated dynamics (from sensor fusion). The residual torque is similar to that in [35]. Control using such an estimated disturbance is tested as well as control using full models of the disturbances. Typically, the all-in-one disturbance torque does not perform as well as a full disturbance model, but it performs better than ignoring disturbances. It is perhaps a more practical approach in cases where the disturbances may be unpredictable. Detailed results can be found in Section 6.5.

6.4 Modified Sliding Scale Control Law

During various tests and explorations for this work, some difficulties are found using the sliding-scale control described in Section 6.2.3. This approach works very well for the inertia matrix and pointing goals of its design mission (the Ørsted Satellite), but, when tested in simulation, its performance does not generalize to other satellites or orientation goals. A modified version is developed here, based on ideas described in [64]. A different sliding surface is defined:

$$\mathbf{s} = J\left(\omega - \omega_{\text{des}} + K_q \delta_{1:3} \left\|\delta_{1:3}\right\|\right). \tag{6.12}$$

For a system that is traveling perfectly along the sliding surface, this effectively changes the relationship between velocity and orientation from linear to quadratic–forcing a higher restorative angular velocity when further out, and a slower angular velocity when near the goal. The sliding variable evolution is

$$\begin{aligned}\dot{\mathbf{s}} = &-\boldsymbol{\omega} \times J\boldsymbol{\omega} + \boldsymbol{\tau}_{\text{ctrl}} + \boldsymbol{\tau}_{\text{dist}} - J\left(\dot{\boldsymbol{\omega}}_{\text{des}} + \boldsymbol{\omega} \times \boldsymbol{\omega}_{\text{des}}\right) \\ &+ \frac{1}{2} J K_q \left(\left(\left\|\delta_{1:3}\right\| I_{3\text{x}3} + \frac{\delta_{1:3}\delta_{1:3}^T}{\left\|\delta_{1:3}\right\|}\right)\left(\delta\left(\omega - \omega_{\text{des}}\right)\right)_{1:3}\right).\end{aligned} \tag{6.13}$$

The desired control torque is defined similarly to Equation 6.10:

$$\begin{aligned}\boldsymbol{\tau}_{\text{s}} = &-\boldsymbol{\tau}_{\text{dist}} + \boldsymbol{\omega} \times J\boldsymbol{\omega} + J\left(\dot{\boldsymbol{\omega}}_{\text{des}} + \boldsymbol{\omega} \times \boldsymbol{\omega}_{\text{des}}\right) \\ &- \frac{1}{2} J K_q \left(\left(\left\|\delta_{1:3}\right\| I_{3\text{x}3} + \frac{\delta_{1:3}\delta_{1:3}^T}{\left\|\delta_{1:3}\right\|}\right)\left(\delta\left(\omega - \omega_{\text{des}}\right)\right)_{1:3}\right) - K_s \mathbf{s}.\end{aligned} \tag{6.14}$$

The magnetic torque is generated as described in Section 6.2.3.

Comparisons between this method and the original can be found in Section 6.5.3. For the main test case, the same inclusion of disturbances, orbit, and satellite is used as in Section 6.2.3. The control parameters are set to $K_s = \text{diag}\left(\begin{bmatrix}0.003, & 0.003, & 0.003\end{bmatrix}\right)$ and $K_q = \frac{1}{\lambda_{\text{avg}}}\text{diag}\left(\begin{bmatrix}0.004, & 0.004, & 0.004\end{bmatrix}\right)$, where $\lambda_{\text{avg}} = \frac{1}{3}\left(\lambda_1 + \lambda_2 + \lambda_3\right)$, and λ_i are the eigenvalues of the inertia matrix J. (This approach roughly preserves the relationship between angular velocity and orientation from the original sliding control law.)

6.5 Tests and Results

In this work, we modify existing attitude control laws to account for a feedforward disturbance term. Simulations are run to compare: the unaltered control law without disturbances (matching those in the original sources), the unaltered control law with disturbances, the altered control law with disturbances accounted for as calculated (based on an estimator), and the altered control law with disturbances calculated as an all-in-one disturbance torque. For the CubeSat cases, only the latter three are used (as there is no "clean" case to compare against from the original papers).

In each of the cases tested, the control laws are analyzed in a simulation based on the framework and estimators in [124]. It is initialized with correct values for the angular velocity and state quaternion, but estimation of other parameters related to biases and disturbances is initially set to 0 and is estimated over time. Actuator and sensor biases and noise are included.

6.5.1 Disturbance Inclusion

6.5.1.a Wie Comparison

A simulation is generated to imitate the test cases in [42], using the control law in Section 6.2.1. The specific parameters are found in Table 6.2.

Table 6.2: Satellite Properties and Test Parameters for Wie Comparison

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}10000 & 9000 & 12000\end{bmatrix}\right)$ kg·m^2
Thrusters	Three-axis, identical on each Maximum Torque: 20 Nm
Disturbances by Case	**Clean** (blue line in figures): no added disturbances. **Disturbed** (orange line in figures): aerodynamic drag, solar radiation pressure, a randomly- varying residual dipole, and randomly-varying thruster bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. **Disturbance Control** (green line in figures): aerodynamic drag, solar radiation pressure, a randomly- varying residual dipole, and randomly-varying thruster bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control. **"All-in-One" Disturbance** (red line in figures): aerodynamic drag, solar radiation pressure, a randomly- varying residual dipole, and randomly-varying thruster bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator, and accounted for in control. (Thruster bias is not tracked, as it is identical in dynamics equations to an all-in-one disturbance torque.)
Orbit Parameters	Geocentric circular orbit Altitude: 540 km Inclination: 28.5° J2 effects included

Continued on next page

Table 6.2 – *Continued from previous page*

Property	**Value/Description**
Update Period	1 second
Simulation Duration	2 hours
Control Algorithms	No control for the first 200 seconds (spacecraft drifting away from identity quaternion). Wie Control law (Section 6.2.1) initiates at 200 seconds. Wie Control Gains: $K_d = 200I_{3\times3}$, $K_p = 50I_{3\times3}$
Quaternion Pointing Goal	0s to 200s: None (no control) 200s to 7200s: [1,0,0,0]
Initial State	Body-frame Angular Velocity: $[0.01, 0.01, 0.0001]$ rad/s Quaternion (relative to ECI): $[1, 0, 0, 0]^T$

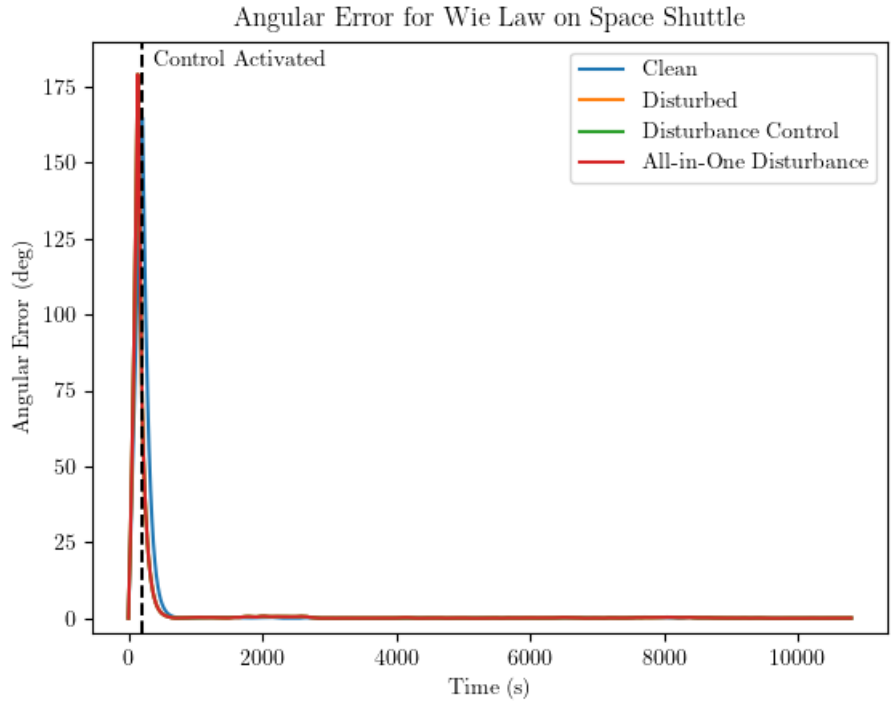


Figure 6.1: Comparison of angular error between different methods of Wie's control law, based on the simulation parameters in Table 6.2. We can see that even with disturbances, whether the control system is aware or not, the error stays on a scale of 0.01 to 0.1 degrees, due to the excellent performance of the baseline control law, the large moment of inertia of the satellite, and full three-axis control. All lines (blue, orange, green, and red) trend together.

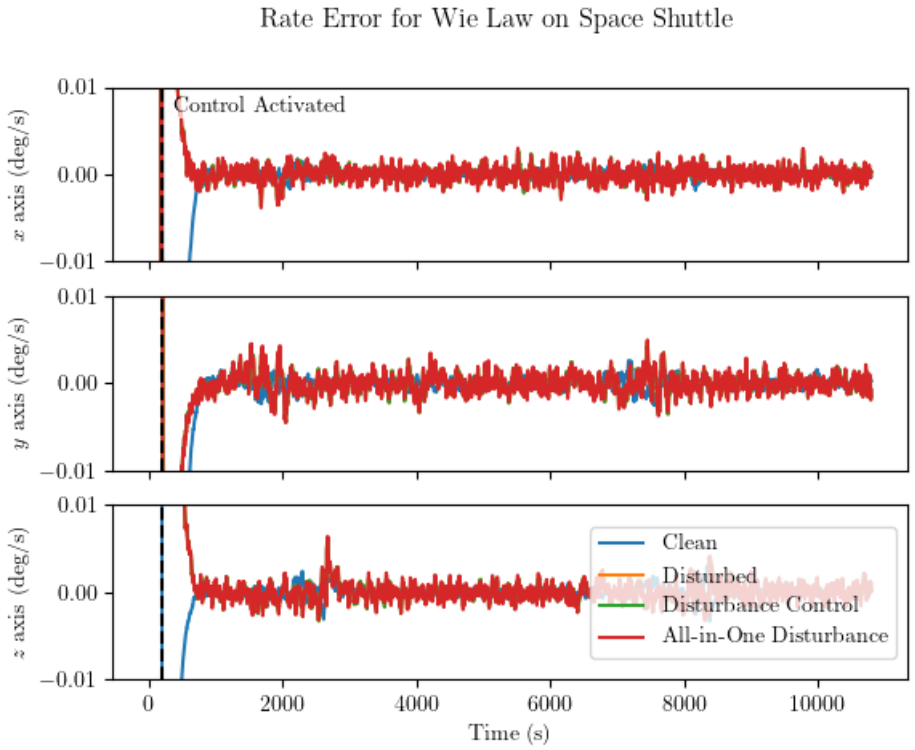


Figure 6.2: Comparison of angular rate error between different methods of Wie's control law by axis, based on the simulation in Table 6.2. We can see that even with disturbances, whether the control system is aware or not, the error stays on a scale of ± 0.001 degrees, due to the excellent performance of the baseline control law, the large moment of inertia of the satellite, and full three-axis control. All lines (blue, orange, green, and red) trend together.

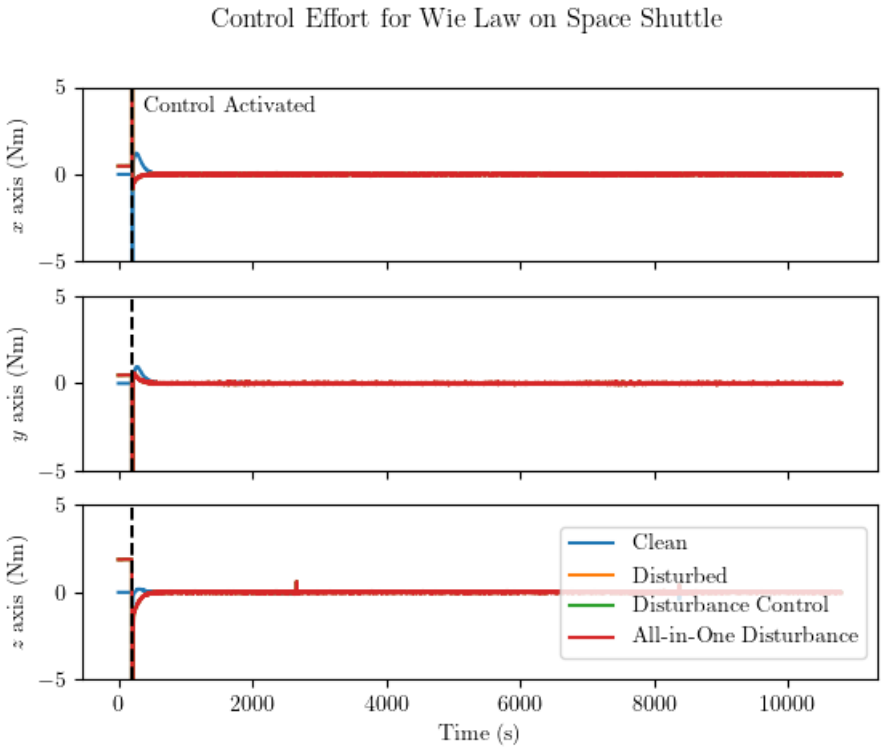


Figure 6.3: Comparison of control effort (with counter-bias adjustment removed) between different methods of Wie's control law by axis, based on the simulation parameters in Table 6.2. It is clear that the control effort required between different methods is small (less than 1 Nm) in all cases, except for the initial slew control. All lines (blue, orange, green, and red) trend together.

As shown in Figure 6.1, with three-axis control and a large moment of inertia, the inclusion of disturbances does not present a significant advantage. The baseline, even when disturbed, already performs very well. In all cases, the system's angular error decays immediately once the control law activates and stays near zero thereafter, on the scale of 0.01 to 0.1 degrees. It can be seen that accounting for the disturbance with the all-in-one disturbance torque creates a bit more error as the system stabilizes (around the 2000-second mark), likely due to the slow adjustment of the disturbance tracking term. Otherwise, it performs as well as the disturbance-aware control. Figures 6.2 and 6.3 show that there is little difference between the cases in angular velocity and control effort as well. Due to the excellent performance of the baseline control law, the addition of disturbance torques into the control algorithm here does not improve performance.

6.5.1.b Lovera Comparison

The inclusion of disturbances has a more significant effect on magnetically controlled systems. The simulation is constructed to imitate the test cases in [59] using the control law described in Section 6.2.2. Detailed information can be found in Table 6.3.

Table 6.3: Satellite Properties and Test Parameters for Lovera Comparison

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}27 & 17 & 25\end{bmatrix}\right)$ kg·m^2
MTQs	Three-axis, identical on each Maximum Magnetic Moment[a]: 50 Am^2
Disturbances by Case	**Clean** (blue line in figures): no added disturbances. **Disturbed** (orange line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. (Dipole is not included as it is dynamically identical to MTQ bias.) **Disturbance Control** (green line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control. (Dipole is not included as it is dynamically identical to MTQ bias.) **"All-in-One" Disturbance** (red line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator and accounted for in control, along with an analytical gravity gradient torque. (Dipole is not included as it is dynamically identical to MTQ bias.)
Orbit Parameters	Geocentric circular orbit Altitude: 450 km Inclination: 87° J2 effects included
Update Period	1 second
Simulation Duration	10 hours
Control Algorithms	No control for the first 200 seconds (spacecraft drifting). Lovera Control law (Section 6.2.2) initiates at 200 seconds. Lovera Control Gains: $k_p = k_v = 50$, $\epsilon = 0.01$

Continued on next page

Table 6.3 – *Continued from previous page*

Property	Value/Description
Quaternion Pointing Goal	0s to 200s: None (no control) 200s to 36000s: [1,0,0,0]
Initial State	Body-frame Angular Velocity: $[1, 1, -1]$ deg/s Quaternion (relative to ECI): $[0, 0, 0, 1]^T$

[a] The comparison case from literature is unrestricted in magnitude.

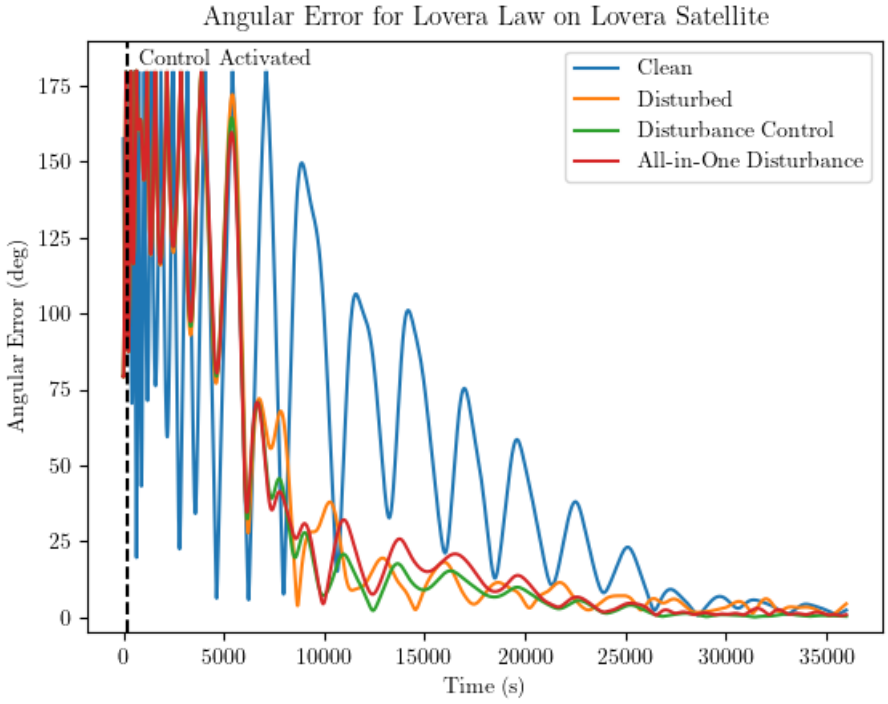


Figure 6.4: Comparison of angular error between different methods of Lovera's control law, based on the simulation parameters in Table 6.3. When there are disturbances and they are not accounted for (orange), the error remains between 1 and 10 degrees. When disturbance is simulated (green), it stabilizes with error between 0.1 and 1 degrees. When disturbance is tracked but not modeled (red), it stays between roughly 0.5 and 2 degrees. The clean case (blue line) is the slowest to converge, perhaps indicating a damping effect by the disturbances.

The results from the Lovera comparison verify the correct implementation of Lovera's magnetic PD control law and highlight the effect of incorporating disturbances. The angular error results in Figure 6.4 reveal clear differences between the approaches. Without accounting for disturbances, the angular error stabilizes between 1 and 10 degrees (orange curve), reflecting the inability to counter persistent external torques. The most significant reduction occurs when disturbances are explicitly modeled and incorporated into the control law (green curve), where errors are reduced to less than 1 degree, demonstrating the value of accurately understanding disturbances, estimating their values, and incorporating them into control.

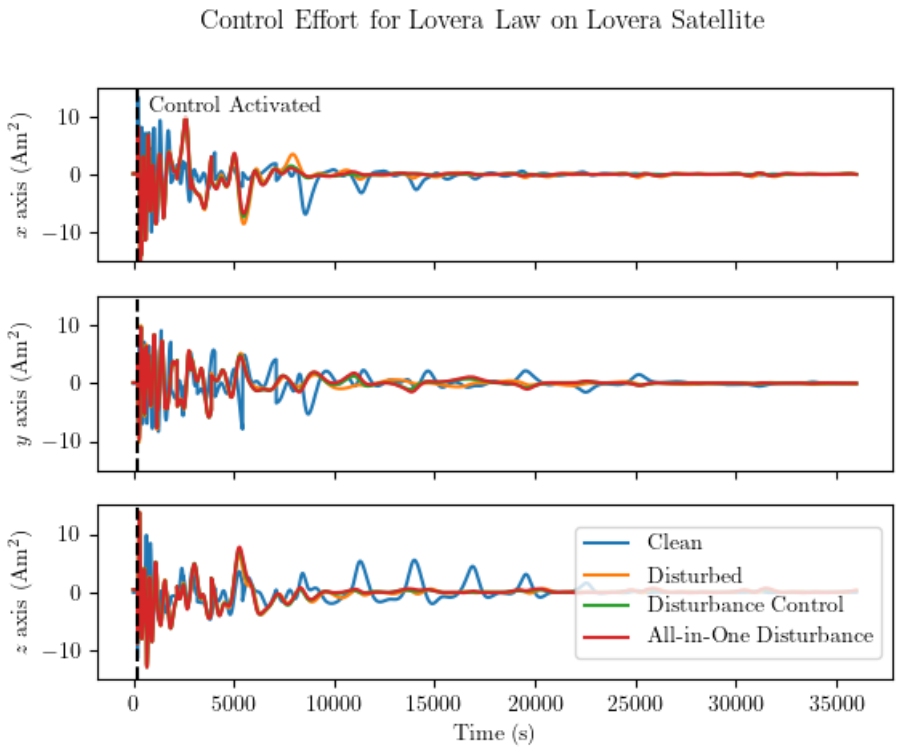


Figure 6.5: Comparison of control effort (with counter-bias adjustment removed) between different methods of Lovera's control law, based on the simulation parameters in Table 6.3. Interestingly, all of the disturbed methods (green, orange, and red) take less control effort than the clean case (blue). The reason for this is unknown but implies some of the disturbance may be stabilizing or interfere constructively with the control law.

Using an all-in-one disturbance estimate provides an intermediate result, with error stabilizing between 0.5 and 2 degrees (red curve). While not perfect, this approach provides a useful trade-off between implementation simplicity and performance improvement, and could be used when disturbances are difficult to predict. The clean case (blue curve)–where no disturbances are applied–surprisingly converges more slowly, suggesting that disturbances may have a minor damping effect on the system dynamics.

The control effort remains consistent across all cases, as shown in Figure 6.5. This indicates that disturbance inclusion does not carry a penalty in terms of control effort. In fact, Figure 6.5 shows a counterintuitive trend: disturbed scenarios require slightly less effort than the clean case. This suggests that the presence of small disturbances may interact with the magnetic control dynamics in a way that reduces the effort needed to stabilize the satellite, a finding that warrants further investigation.

Overall, these results validate the author's implementation of Lovera's law (with the clean case matching results in [59]) and demonstrate the benefits of disturbance-aware control for underactuated systems like magnetically actuated satellites. Explicitly modeling, estimating, and countering disturbances lead to improved pointing accuracy, reducing errors that would otherwise limit mission performance. Even if the disturbance cannot be exactly modeled, using an all-in-one disturbance term reduces pointing error by a factor of 2-5. This makes disturbance-aware strategies particularly valuable for missions requiring operation under disturbance-heavy conditions.

6.5.1.c Wisniewski Comparison

The inclusion of some disturbance is already a part of the formulation of the control law in Section 6.2.3, due to the nature of the sliding surface law. This control law is tested in a simulation based on the test cases in [128]. Details can be found in Table 6.4.

Table 6.4: Satellite Properties and Test Parameters for Wisniewski Comparison

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}3.428 & 2.904 & 1.275\end{bmatrix}\right)$ kg·m^2
MTQs	Three-axis, identical on each Maximum Magnetic Moment: 20 Am^2
Disturbances by Case	**Clean** (blue line in figures): includes gravity gradient[a]. **Disturbed** (orange line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. (Dipole is not included as it is dynamically identical to MTQ bias.) **Disturbance Control** (green line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control. (Dipole is not included as it is dynamically identical to MTQ bias.) **"All-in-One" Disturbance** (red line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator and accounted for in control, along with an analytical gravity gradient torque. (Dipole is not included as it is dynamically identical to MTQ bias.)
Orbit Parameters	Geocentric near-circular orbit Eccentricity: 0.028599 Altitude: 450 km at perigee, 850 km at apogee Inclination: 96.1° RAAN: 105.2° J2 effects included
Update Period	1 second
Simulation Duration	10 hours
Control Algorithms	No control for first 200 seconds (spacecraft drifting). Wisniewski Control law (Section 6.2.3) initiates at 200 seconds.

Continued on next page

Table 6.4 – *Continued from previous page*

Property	**Value/Description**
	Wisniewski Control Gains: $K_s = 0.003I_{3\times3}$, $K_q = 0.002I_{3\times3}$
Pointing Goal	0s to 200s: None (no control) 200s to 36000s: body z-axis pointing zenith and body x-axis aligned with the orbit normal.
Initial State	Body-frame Angular Velocity: $[-0.002, 0.002, 0.002]$ rad/s Orientation: (roll, pitch, yaw) = (60°, 100°, -100°) from desired attitude with body z-axis pointing zenith and body x-axis aligned with the orbit normal.

[a] Gravity gradient included to match original case in [128].

The results from this test demonstrate the behavior of the sliding mode control law under various disturbance scenarios and confirm its correct implementation (compare the clean case to the results in [128]). The angular error results in Figure 6.7 highlight the dramatic impact of unmodeled disturbances and the benefits of disturbance inclusion. When disturbances are ignored (orange curve), the system exhibits large, persistent errors, oscillating between 10 and 60 degrees. This clearly illustrates the destabilizing effects of external torques like gravity gradient or residual dipole when they are not accounted for. By contrast, explicitly modeling, estimating, and incorporating disturbances into the control law (green curve) significantly improves performance, with angular errors stabilizing mostly below 3 degrees. The all-in-one disturbance estimate (red curve) performs very similarly, reducing the errors to 2–5 degrees. While marginally less accurate than explicit disturbance modeling, this approach demonstrates that approximating disturbances as a single term is effective and offers a practical trade-off when detailed disturbance models are unavailable.

The log-scale angular error plot (Figure 6.6) further emphasizes these trends and clarifies the small-angle errors. The disturbance-aware approaches exhibit rapid error reduction and sustained stability, approaching clean-case performance levels. In stark contrast, the disturbed case fails to converge, with errors remaining orders of magnitude larger. This comparison highlights the critical role disturbance compensation plays in stabilizing the system under realistic operating conditions.

The control effort results in Figure 6.8 show that incorporating disturbances does not increase the actuation demand. The disturbance-aware cases maintain consistent and steady control effort, comparable to the clean case. The disturbed case exhibits similar control efforts and is neither more efficient nor dramatically more power-hungry as it works to counter the large oscillations. This may be due to the nature of the sliding-surface control law. This finding underscores that disturbance compensation improves performance without requiring additional actuator effort.

In summary, these results confirm that Wisniewski's sliding mode control law performs well under ideal conditions but degrades significantly when disturbances are ignored. Explicitly including disturbances in the control formulation improves pointing accuracy dramatically, reducing angular errors to below 3 degrees and enabling robust stabilization. The all-in-one disturbance estimate achieves similar improvements, stabilizing errors within 3–5 degrees.

This makes it a valuable alternative when precise disturbance models are unavailable. The analysis highlights the importance of disturbance-aware control strategies for underactuated systems like magnetically controlled satellites, where unaccounted disturbances can severely limit performance.

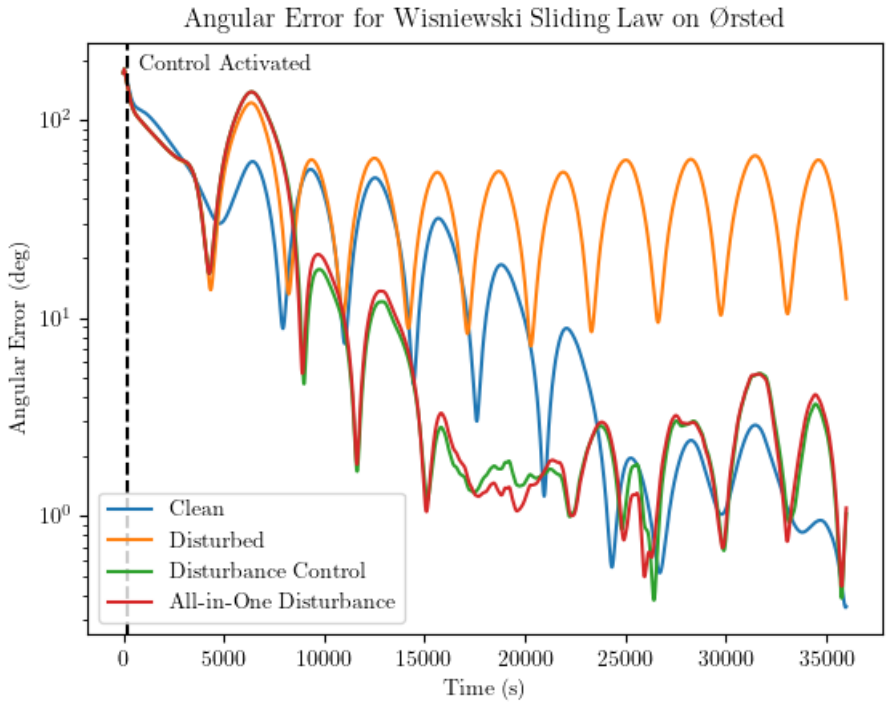


Figure 6.6: Log scale comparison of angular error between different methods of Wisniewski's Sliding control law, based on the simulation parameters in Table 6.4. With disturbances accounted for, either directly (green line) or by tracking an all-in-one disturbance term (red line), the angular error stays between 0.5 and 3 degrees, matching (and converging faster than) the case without extra disturbances (blue). When disturbances are present but not countered (orange line), the angular error remains between 9 and 60 degrees.

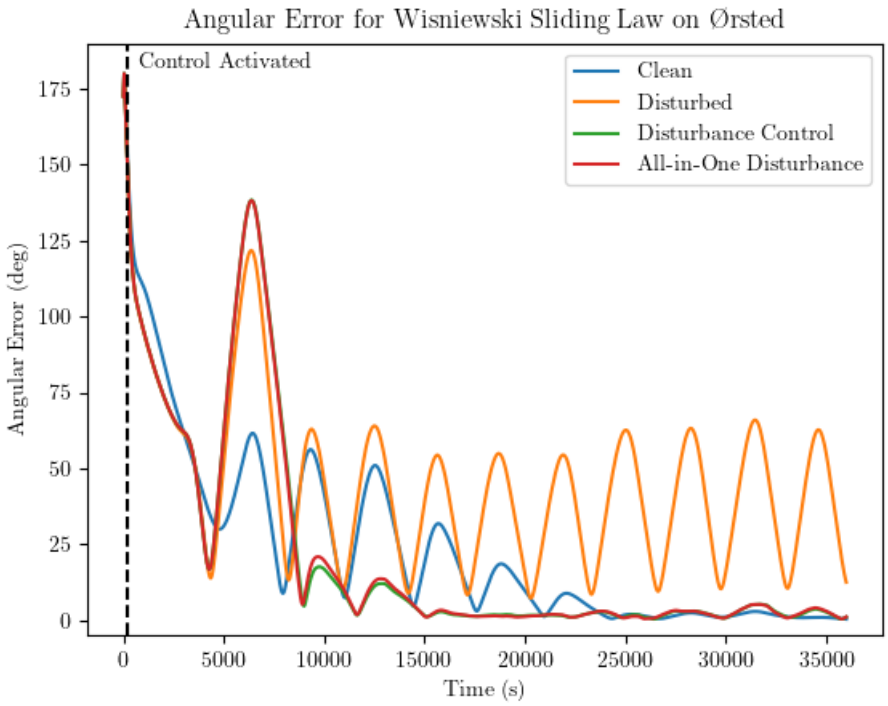


Figure 6.7: Comparison of angular error between different methods of Wisniewski's Sliding control law, based on the simulation parameters in Table 6.4. With disturbances accounted for, either directly (green line) or by tracking an all-in-one disturbance term (red line), the angular error stays between 0.5 and 3 degrees, matching (and converging faster than) the case without extra disturbances (blue). When disturbances are present but not countered (orange line), the angular error remains between 9 and 60 degrees.

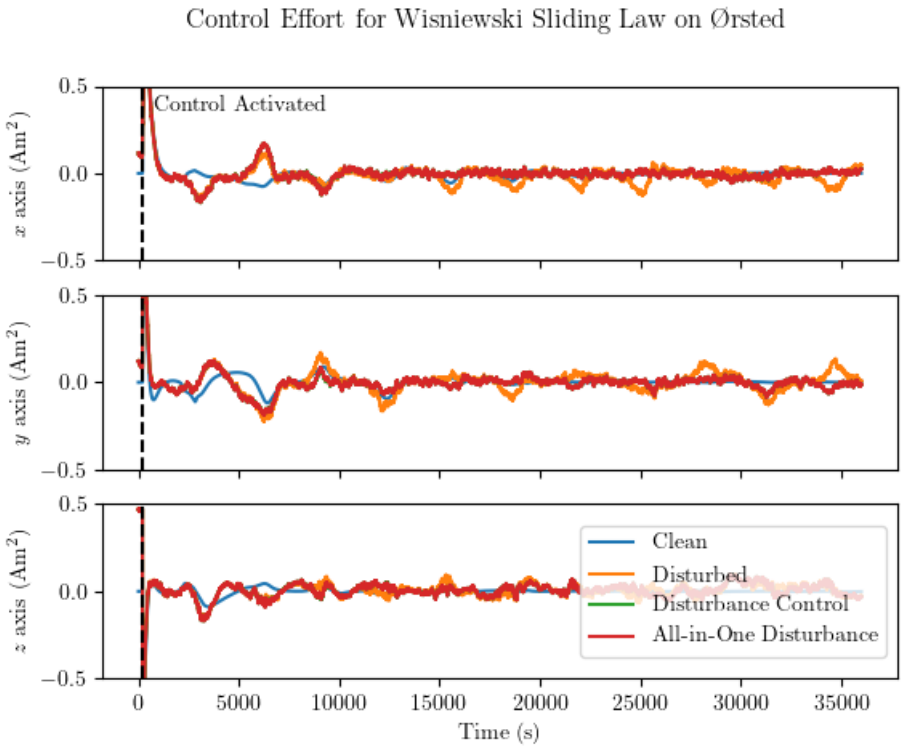


Figure 6.8: Comparison of control effort (with counter-bias adjustment removed) between different methods of Wisniewski's Sliding control law, based on the simulation parameters in Table 6.4. Similar effort (roughly 0.2 Am^2 or less) is required in all cases, independent of whether disturbances are present or not, and whether they are countered or not.

6.5.1.d Modified Sliding Surface Comparison

It is worth considering the performance of the Modified sliding law laid out in Section 6.4. The same simulations as in Section 6.5.1.c are attempted. (See Table 6.4.) The results from this test demonstrate the robustness of the modified sliding surface control law under various disturbance scenarios, with all cases showing similar performance. Unlike prior tests where disturbance inclusion yielded dramatic improvements, the differences here are modest.

The angular error results in Figures 6.10 and 6.9 highlight this trend. In the clean case (blue), errors stabilize around 1–3 degrees, confirming baseline performance with gravity gradient disturbances. When the additional disturbances are introduced without compensation (orange), errors increase slightly to 0.5–4 degrees but remain bounded. Explicit disturbance modeling (green) and the all-in-one disturbance approach (red) show similar behavior, with errors stabilizing between 0.5–6 degrees. They do not show a large improvement on the disturbance-ignorant performance. This indicates that this may be a valuable and robust control law, even when disturbances are ignored, or it may be an effect of this scenario, where the desired orientation goal is stable due to the gravity gradient.

Control effort results in Figure 6.11 demonstrate that all methods require similarly low actuation effort, consistently below 0.1 Am^2. Notably, disturbance-aware approaches (green and red) do not impose any additional actuation demand, underscoring the efficiency of the control law even when compensating for external torques.

In summary, the modified sliding surface control law performs reliably across all scenarios, maintaining stable angular errors and efficient control effort. While disturbance-aware strategies provide slight improvements, errors remain between 0.5–6 degrees in all cases. These results highlight the modified control law's inherent robustness, even when disturbances are unmodeled or only approximately compensated for.

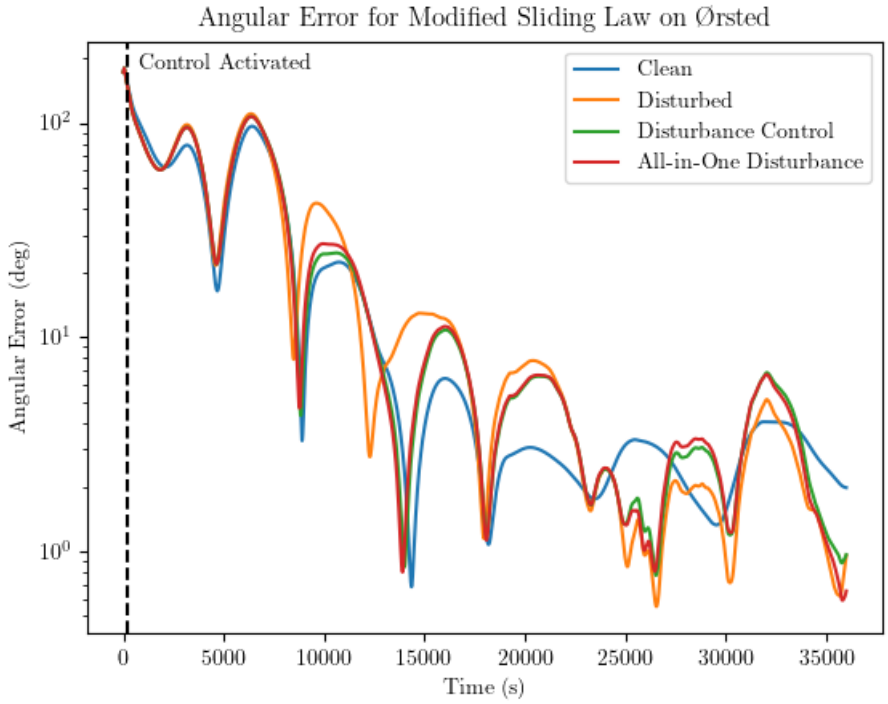


Figure 6.9: Log-scale comparison of angular error between different methods of the Modified Sliding control law, based on the simulation parameters in Table 6.4 with the original sliding control law replaced by the modified version from Section 6.4. All cases, whether disturbances are present or not and accounted for or not, perform similarly, with an error between 1 and 5 degrees.

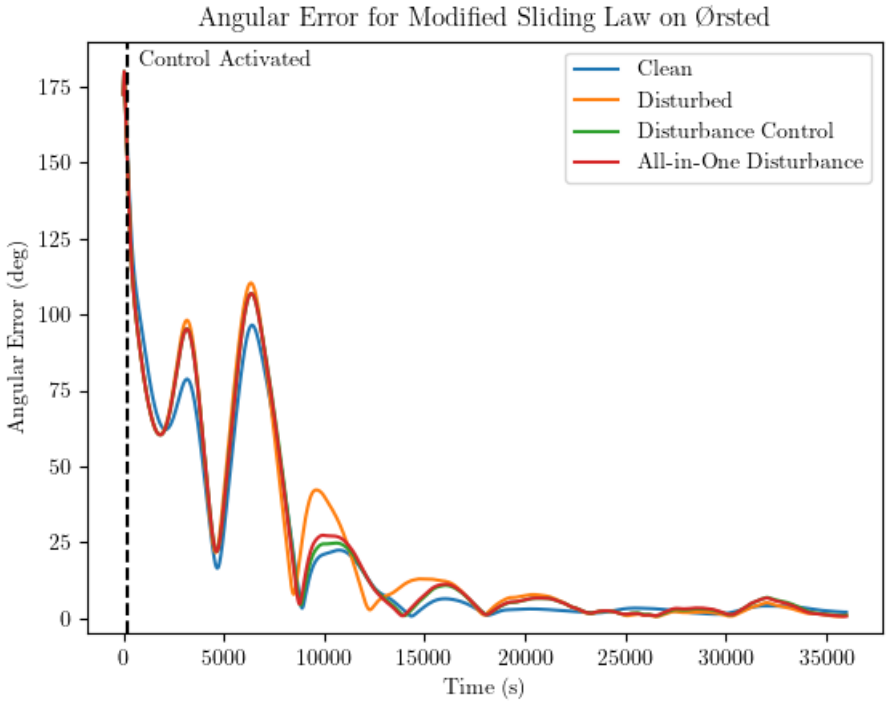


Figure 6.10: Comparison of angular error between different methods of the Modified Sliding control law, based on the simulation parameters in Table 6.4 with the original sliding control law replaced by the modified version from Section 6.4. All cases, whether disturbances are present or not and accounted for or not, perform similarly, with an error between 1 and 5 degrees.

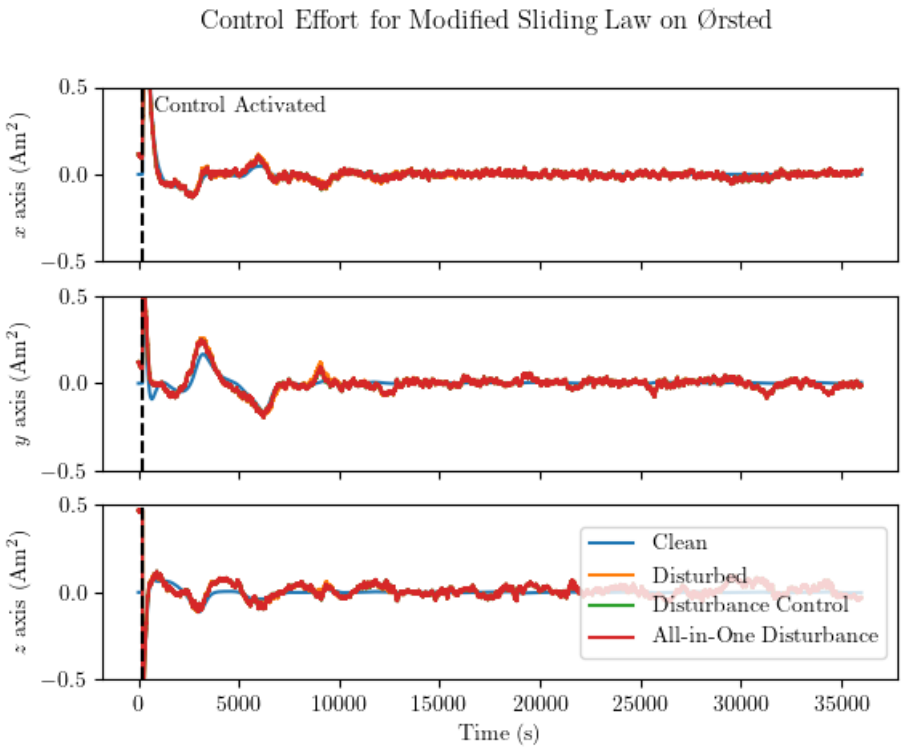


Figure 6.11: Comparison of control effort (with counter-bias adjustment removed) between different methods of Modified Sliding control law, based on the simulation parameters in Table 6.4 with the original sliding control law replaced by the modified version from Section 6.4. Similar effort (roughly 0.1 Am^2 or less) is required whether disturbances are present or not, and whether they are countered or not.

6.5.2 CubeSat Tests

The magnetic control laws are also tested on a simulated CubeSat to determine their applicability to other missions. CubeSats are becoming more common and could benefit from improved magnetic control due to their constraints on size, weight, and power. Making magnetic control more accurate by accounting for disturbances could enable greater utility of CubeSats and smallsats.

6.5.2.a Wisniewski CubeSat Test

The control law from Section 6.2.3 is simulated on a CubeSat. The relevant information can be found in Table 6.5.

Table 6.5: Satellite Properties and Test Parameters for Wisniewski CubeSat Test

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}0.0333 & 0.0341 & 0.0081\end{bmatrix}\right)$ kg·m^2
MTQs	Three-axis, identical on each Maximum Magnetic Moment: 1 Am^2
Disturbances by Case[b]	**Disturbed** (orange line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. **Disturbance Control** (green line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control. **"All-in-One" Disturbance**[c](red line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator and accounted for in control.
Orbit Parameters	Geocentric circular orbit Altitude: 450 km Inclination: 87° J2 effects included
Update Period	1 second
Simulation Duration	10 hours
Control Algorithms	No control for first 200 seconds (spacecraft drifting). Wisniewski Control law (Section 6.2.3) initiates at 200 seconds.

Continued on next page

Table 6.5 – *Continued from previous page*

Property	**Value/Description**
	Wisniewski Control Gains: $K_s = 0.0003 I_{3\times3}$, $K_q = 0.0001 I_{3\times3}$
Pointing Goal	0s to 200s: None (no control) 200s to 36000s: body z-axis pointing zenith and body x-axis aligned with the orbit normal.
Initial State	Body-frame Angular Velocity: $[0.25, 0.25, 0.25]$ deg/s Quaternion (relative to ECI frame): $\frac{1}{\sqrt{2}}[1, 0, 1, 0]$.

[a] There are only three cases, as opposed to the four used in the test for Section 6.5.1. The "clean" case included there is used to verify the implemented control laws against their original publications, which is not necessary here, as they are being applied to a new scenario.

[b] Dipole is not included as it is dynamically identical to MTQ bias.

[c] Unlike larger satellite cases in this work in Subsection 6.5.1, there is no gravity gradient term added analytically, as doing so proved unnecessary, perhaps due to the small inertia matrix.

The benefits in this case are clear. The "disturbed" controller (orange lines), which does not account for disturbance, is wildly off, as shown in Figure 6.12. When accounting for disturbances, an error of just a few degrees is achieved, despite the poor sensors and actuators used on this simulated CubeSat. An all-in-one disturbance controller (red) performs as well as the fully-modeled disturbance control (green) and still manages about 10 degrees of error. Minimal control effort is required in all cases, as shown in Figure 6.13.

The results from this test demonstrate the behavior of Wisniewski's sliding mode control law on a CubeSat under realistic disturbance conditions and show the clear effect of disturbance inclusion. The angular error results in Figure 6.12 highlight the dramatic impact of disturbances and the benefits of disturbance compensation. In the disturbed case (orange curve), the spacecraft is essentially uncontrolled. The angular errors remain extremely high, approaching 180 degrees and failing to stabilize. By contrast, explicitly modeling, estimating, and including disturbances (green curve) achieves excellent performance, with angular errors stabilizing below 4 degrees. The all-in-one disturbance estimate (red curve) provides a practical alternative, reducing errors to between 5 and 20 degrees. While slightly less accurate than full disturbance-aware control, this approach still delivers substantial improvements compared to ignoring disturbances and requires less knowledge of the disturbance environment.

The control effort results in Figure 6.13 reveal that including disturbances improves performance without increasing actuator demands. Both the disturbance-aware and all-in-one approaches maintain smooth and efficient control effort, while the disturbed case exhibits higher-effort and more erratic control behavior as it struggles to counteract external disturbances.

Overall, these results confirm that Wisniewski's sliding mode control law is highly effective when disturbances are accounted for, even on CubeSats. Explicit disturbance inclusion achieves the best performance, reducing angular errors and improving stability while maintaining efficient actuation. The all-in-one disturbance estimate offers a robust and computationally

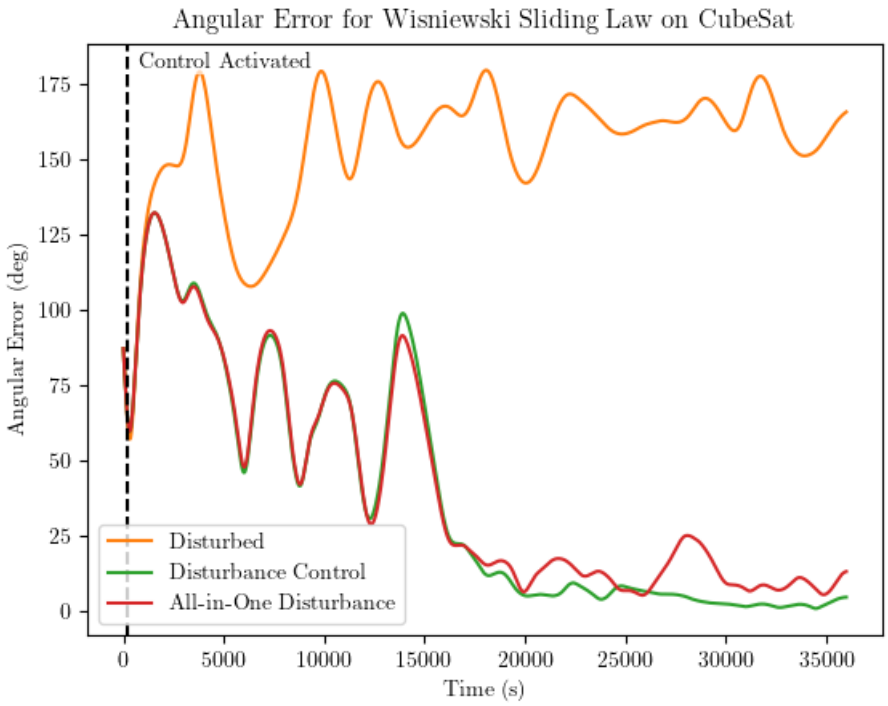


Figure 6.12: Comparison of angular error between different methods of Wisniewski's sliding control law on a disturbed CubeSat, based on test parameters described in Table 6.5. The "Disturbed" case (orange) that does not account for disturbances is basically uncontrolled, remaining almost 180 degrees off. The "General Disturbance" case (red) with an all-in-one estimated disturbance stays between 5 and 20 degrees. The "Disturbance Control" case (green) with a disturbance model stays between 0 and 4 degrees.

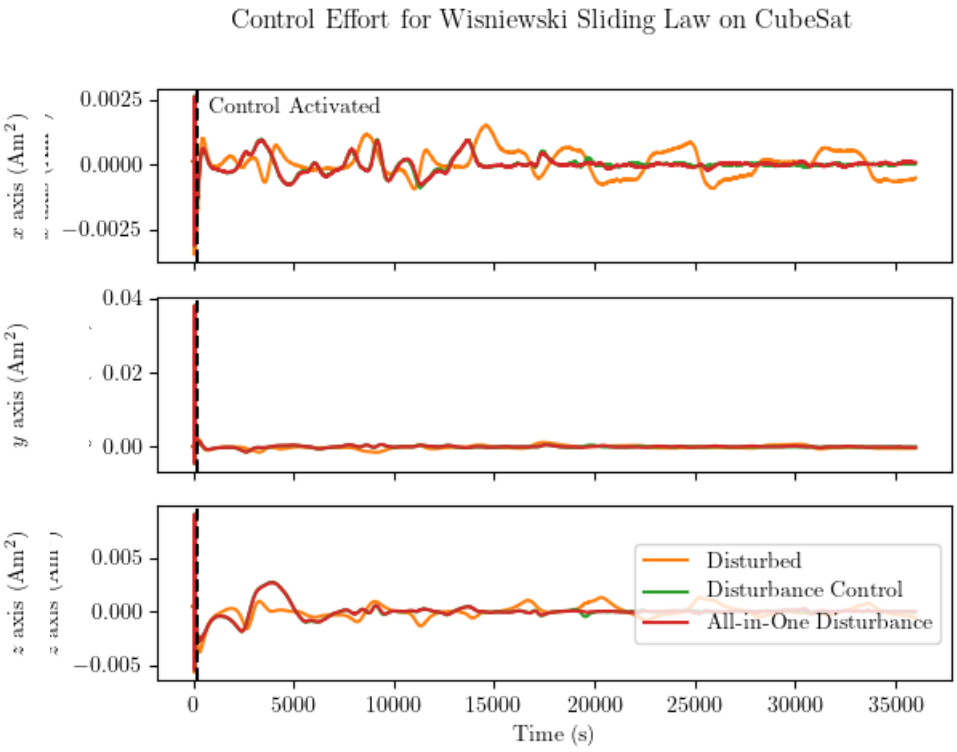


Figure 6.13: Comparison of control effort (with counter-bias adjustment removed) between different methods of Wisniewski's sliding control law on a disturbed CubeSat, based on test parameters described in Table 6.5. The case that does not account for disturbances (orange) requires more effort than the disturbance-aware cases (green and red).

efficient alternative, making it a worthy contender for CubeSat systems operating under disturbance-heavy conditions.

6.5.2.b Lovera CubeSat Test

The control law from Section 6.2.2 is simulated on a CubeSat. The relevant test information can be found in Table 6.6.

Table 6.6: Satellite Properties and Test Parameters for Lovera CubeSat Test

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}0.0333 & 0.0341 & 0.0081\end{bmatrix}\right)$ kg·m^2
MTQs	Three-axis, identical on each Maximum Magnetic Moment: 1 Am^2
Disturbances by Case[b]	**Disturbed** (orange line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. **Disturbance Control** (green line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control. **"All-in-One" Disturbance**[c](red line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator and accounted for in control.
Orbit Parameters	Geocentric circular orbit Altitude: 450 km Inclination: 87° J2 effects included
Update Period	1 second
Simulation Duration	10 hours
Control Algorithms	No control for the first 200 seconds (spacecraft drifting). Lovera Control law (Section 6.2.2) initiates at 200 seconds. Lovera Control Gains: $k_p = 0.01$, $k_v = 0.015$, $\epsilon = 0.001$
Pointing Goal	0s to 200s: None (no control)

Continued on next page

Table 6.6 – *Continued from previous page*

Property	**Value/Description**
	200s to 36000s: body z-axis pointing zenith and body x-axis aligned with the orbit normal.
Initial State	Body-frame Angular Velocity: $[0.25, 0.25, 0.25]$ deg/s Quaternion (relative to ECI frame): $\frac{1}{\sqrt{2}}[1, 0, 1, 0]$.

[a] There are only three cases, as opposed to the four used in the test for Section 6.5.1. The "clean" case included there is used to verify the implemented control laws against their original publications, which is not necessary here, as they are being applied to a new scenario.

[b] Dipole is not included as it is dynamically identical to MTQ bias.

[c] Unlike larger satellite cases in this work in Subsection 6.5.1, there is no gravity gradient term added analytically, as doing so proved unnecessary, perhaps due to the small inertia matrix.

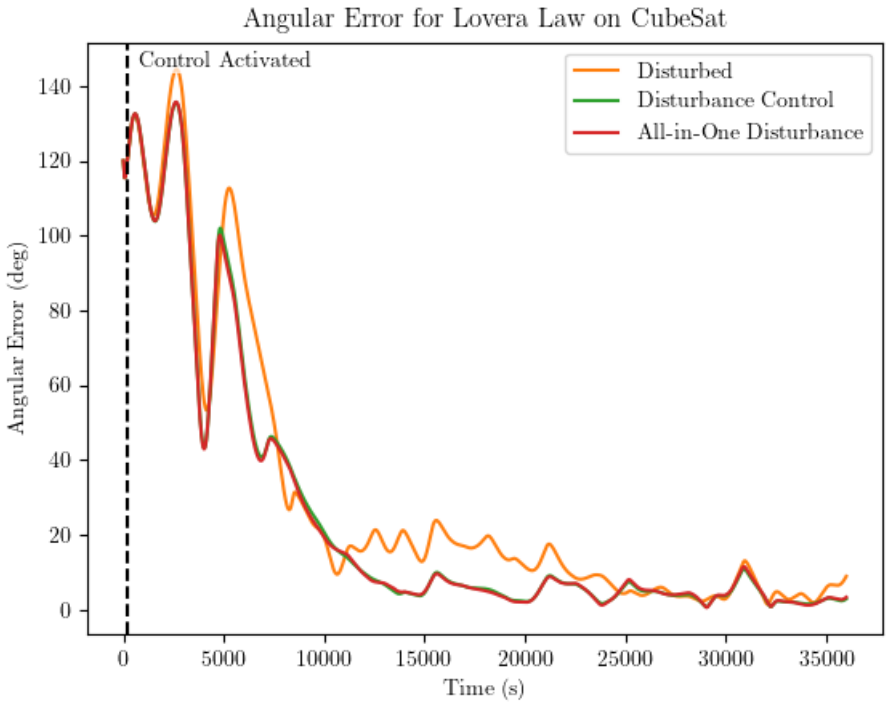


Figure 6.14: Comparison of angular error between different methods of Lovera's control law on a disturbed CubeSat, based on test parameters described in Table 6.6. The cases perform similarly, all reaching within single-degree levels of accuracy, though the cases that are disturbance-aware (green, red) converge somewhat faster.

The results from this test demonstrate the behavior of Lovera's magnetic PD control law under realistic disturbance scenarios and on a CubeSat platform. The angular error results in Figure 6.4 highlight the benefits of disturbance compensation. In the disturbed

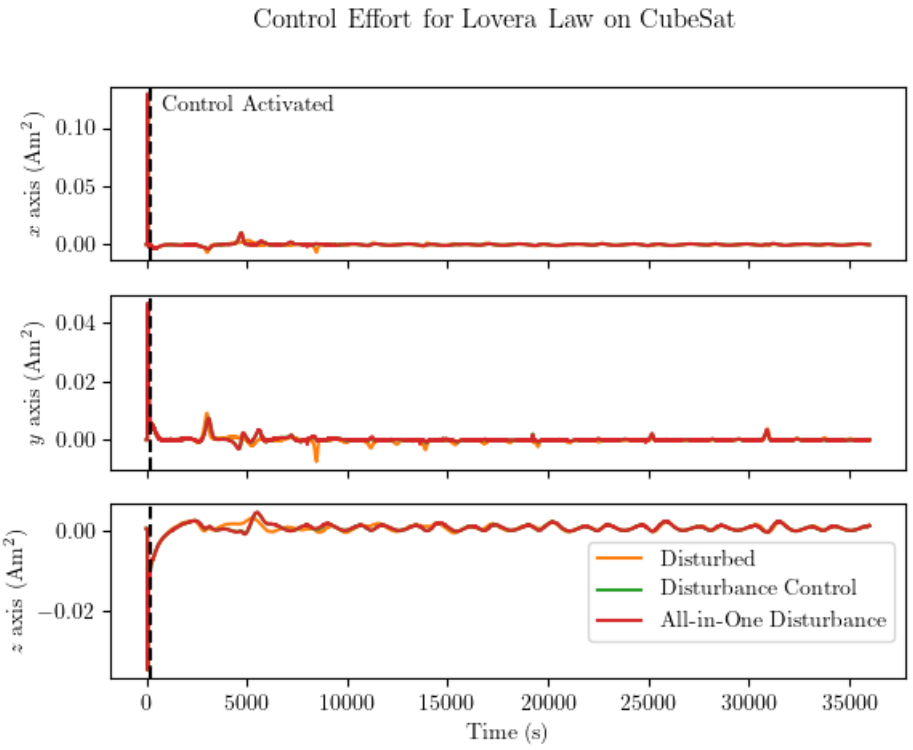


Figure 6.15: Comparison of control effort (with counter-bias adjustment removed) between different methods of Lovera's control law on a disturbed CubeSat, based on test parameters described in Table 6.6. The case that does not account for disturbances (orange) requires slightly more effort than the disturbance-aware cases.

case (orange curve), angular errors converge to roughly 10 degrees, reflecting the system's inability to fully reject disturbances like aerodynamic drag. By contrast, explicitly modeling and incorporating disturbances into the control law (green curve) reduces errors to around 5 degrees, demonstrating the value of understanding and compensating for external disturbances. The all-in-one disturbance estimate (red curve) achieves near-identical performance, stabilizing mostly between 3 and 5 degrees, showing that this approach offers a practical and effective alternative for computationally constrained systems. All methods show a small unknown spike around 32000 seconds, which may come from a specific event connected to mission goals or disturbances. The control effort results in Figure 6.5 show that disturbance-aware methods achieve these improvements without increasing actuator demands. The control effort for both disturbance-aware approaches remains smooth and steady, while the disturbed case exhibits larger (roughly twice the scale) fluctuations.

Overall, these results demonstrate the suitability of Lovera's control law for CubeSat platforms operating under real-world disturbances. Explicit disturbance inclusion leads to faster stabilization, lower angular errors, and improved angular rate control while maintaining efficient actuator usage. The all-in-one disturbance estimate, in particular, emerges as a robust and computationally efficient alternative for achieving precision pointing performance on resource-limited CubeSats.

6.5.2.c Modified Sliding CubeSat Test

The control law from Section 6.4 is simulated on a CubeSat. More details are given in Table 6.7.

Table 6.7: Satellite Properties and Test Parameters for Modified Sliding CubeSat Test

Property	**Value/Description**
Inertia Matrix	$J = \text{diag}\left(\begin{bmatrix}0.0333 & 0.0341 & 0.0081\end{bmatrix}\right)$ kg·m^2
MTQs	Three-axis, identical on each Maximum Magnetic Moment: 1 Am^2
Disturbances by Case[b]	**Disturbed** (orange line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite but are not included in the estimator or accounted for in control. **Disturbance Control** (green line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite, are estimated onboard, and accounted for in control.

Continued on next page

Table 6.7 – *Continued from previous page*

Property	Value/Description
	"All-in-One" Disturbance[c](red line in figures): gravity gradient, aerodynamic drag, solar radiation pressure, and randomly-varying MTQ bias. These are included in the "real" satellite. An all-in-one three-axis disturbance torque term is included in the estimator and accounted for in control.
Orbit Parameters	Geocentric circular orbit Altitude: 450 km Inclination: 87° J2 effects included
Update Period	1 second
Simulation Duration	10 hours
Control Algorithms	No control for the first 200 seconds (spacecraft drifting). Modified Sliding Control law (Section 6.4) initiates at 200 seconds. Modified Sliding Control Gains[d]: $K_s = 0.006 I_{3\times3}$, $K_q = \frac{0.0002}{\lambda_{\text{avg}}} I_{3\times3}$
Pointing Goal	0s to 200s: None (no control) 200s to 36000s: body z-axis pointing zenith and body x-axis aligned with the orbit normal.
Initial State	Body-frame Angular Velocity: $[0.25, 0.25, 0.25]$ deg/s Quaternion (relative to ECI frame): $\frac{1}{\sqrt{2}}[1, 0, 1, 0]$.

[a] There are only three cases, as opposed to the four used in the test for Section 6.5.1. The "clean" case included there is used to verify the implemented control laws against their original publications, which is not necessary here, as they are being applied to a new scenario.

[b] Dipole is not included as it is dynamically identical to MTQ bias.

[c] Unlike larger satellite cases in this work in Subsection 6.5.1, there is no gravity gradient term added analytically, as doing so proved unnecessary, perhaps due to the small inertia matrix.

[d] The value $\lambda_{\text{avg}} = \frac{1}{3}(\lambda_1 + \lambda_2 + \lambda_3)$, where λ_i for $i = 1, 2, 3$ are the eigenvalues of the spacecraft inertia matrix J. (This approach roughly preserves the relationship between angular velocity and orientation from the original sliding control law.)

The results from this test clearly show the effect of disturbance inclusion and demonstrate the behavior of the modified sliding surface control law on a CubeSat platform. The angular error results in Figure 6.16 highlight the importance of disturbance compensation. In the disturbed case (orange curve), angular errors grow wildly–including through 180 degrees of error–but eventually stabilize between 10 and 50 degrees. By contrast, explicitly modeling and incorporating disturbances (green curve) reduces errors significantly, stabilizing between 2 and 8 degrees. The all-in-one disturbance estimate (red curve) achieves similar performance,

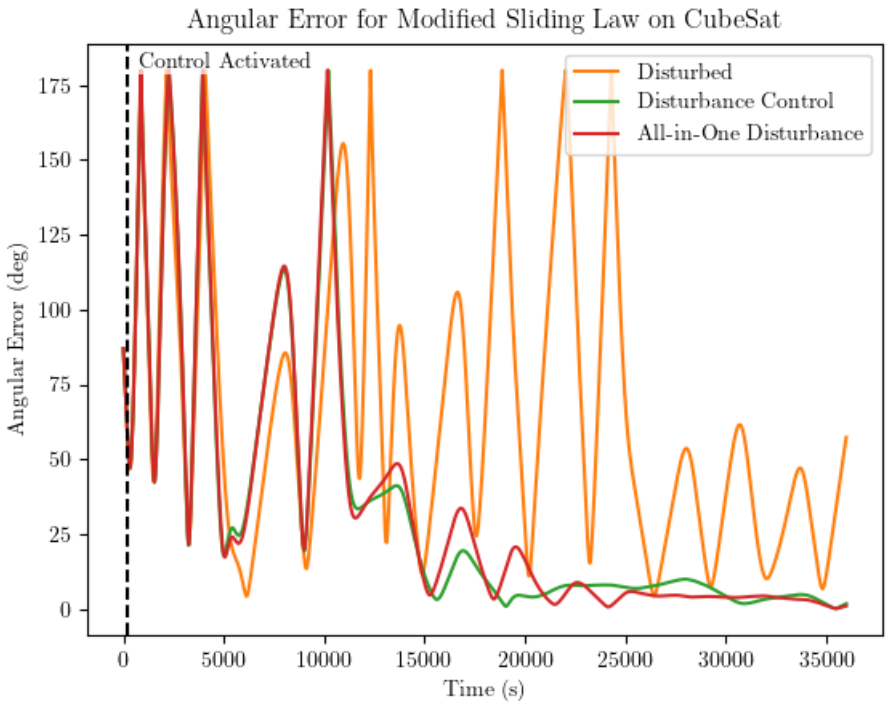


Figure 6.16: Comparison of angular error between different methods of the Modified Sliding control law on a disturbed CubeSat, based on test parameters described in Table 6.7. The case that does not account for disturbances (orange) is basically uncontrolled, remaining almost 180 degrees off, though near the end it stays within 70 degrees. The case with an all-in-one estimated disturbance (red) and a disturbance model (green) perform similarly, staying within 10 degrees.

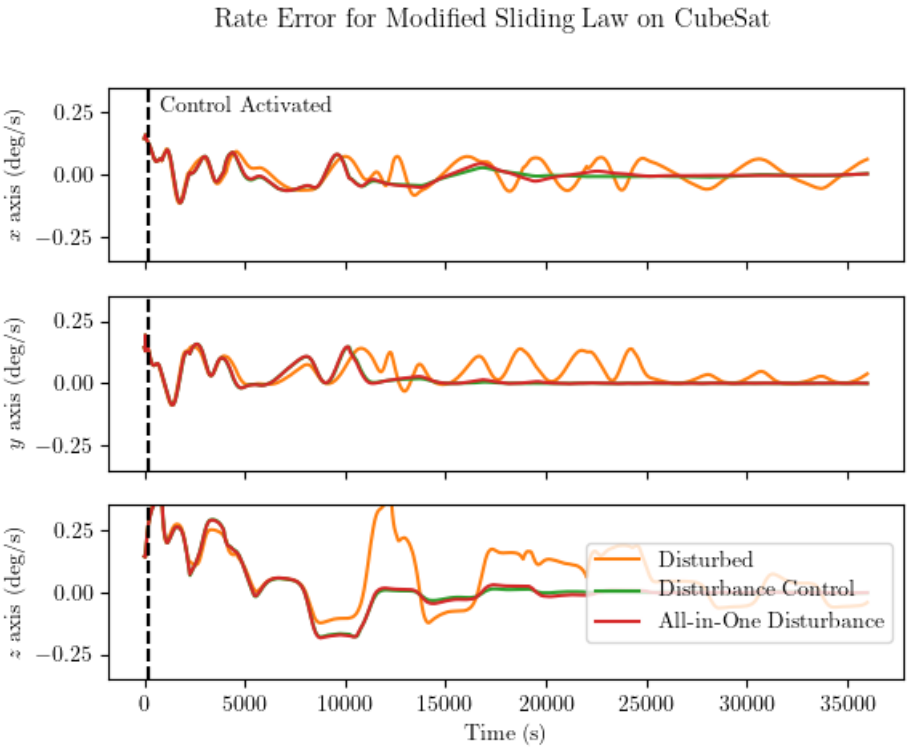


Figure 6.17: Comparison of angular rate error between different methods of the modified sliding control law on a disturbed CubeSat, based on test parameters described in Table 6.7. The case that does not account for disturbances (orange) is basically uncontrolled, with an error that often exceeds 0.05 deg/s. The case with an all-in-one estimated disturbance (red) stays within 0.03 deg/s. The case with a disturbance model (green) stays within roughly 0.01 deg/s.

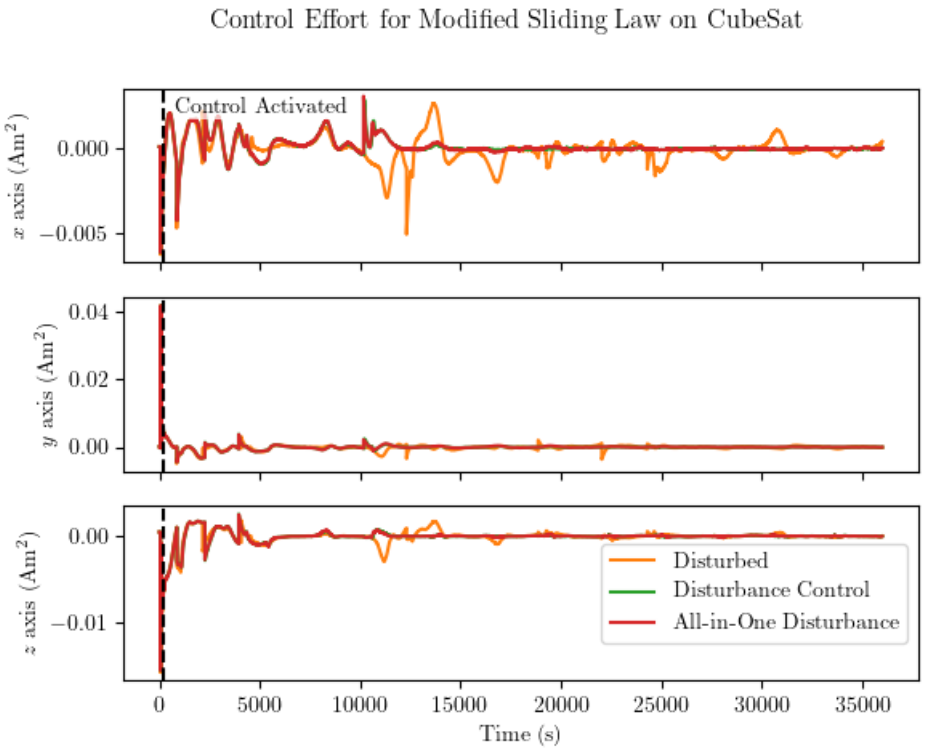


Figure 6.18: Comparison of control effort (with counter-bias adjustment removed) between different methods of the modified sliding control law on a disturbed CubeSat, based on test parameters described in Table 6.7. The case that does not account for disturbances (orange) requires more effort than the disturbance-aware cases (green, red).

with errors stabilizing between 3 and 6 degrees, offering a simpler yet effective alternative to full disturbance modeling. The reasons for the all-in-one disturbance controller equaling or surpassing the fully-disturbance-aware approach are not completely clear. This over-performance may be due to the all-in-one disturbance having some "memory" and acting as a damping factor.

The control effort results in Figure 6.18 show that disturbance inclusion improves performance without increasing actuator demand. Both disturbance-aware approaches reduce control effort fluctuations compared to the disturbed case, achieving smoother and more efficient actuation.

Overall, these results confirm that the modified sliding surface control law achieves robust and efficient stabilization when disturbances are included. Explicit disturbance modeling is effective but not better than the all-in-one disturbance approach. These findings emphasize the value of disturbance-aware control for CubeSats operating in real-world conditions.

6.5.3 Sliding versus Modified Sliding

It is worth directly comparing the modified and original sliding laws, laid out respectively in Sections 6.2.3 and 6.4. Their behavior in the Disturbance Control case (as laid out in Section 6.5.1.c) is directly compared in Figure 6.19. In the original test case, Wisniewski's control law achieves improved performance when disturbances are explicitly modeled, stabilizing angular errors between 0.5 and 3 degrees. The modified sliding surface performs slightly worse in this case, with errors stabilizing between 1 and 5 degrees. However, if disturbances are unknown or ignored, as shown in Figure 6.20, Wisniewski's law fails to converge (errors between 9 and 60 degrees), whereas the modified sliding surface demonstrates much greater robustness, stabilizing errors within 1–5 degrees.

In the CubeSat test case, this trend becomes even more pronounced. As shown in Figure 6.21, when disturbances are unmodeled, errors for Wisniewski's law grow as large as 180 degrees, effectively leaving the system uncontrolled. The modified sliding surface, by contrast, stabilizes errors between 10 and 50 degrees, underscoring its improved ability to handle unaccounted disturbances. When disturbances are modeled, as seen in Figure 6.22, both methods perform well: Wisniewski's law achieves the best performance, with errors stabilizing below 4 degrees, while the modified sliding surface converges slightly less precisely, between 2 and 8 degrees.

These results confirm that while Wisniewski's control law offers better accuracy under ideal conditions or with disturbance modeling, the modified sliding surface has better robustness to unmodeled disturbances, making it particularly valuable for real-world applications where disturbance conditions may be unpredictable. This robustness is especially evident in the CubeSat test case, where the modified sliding surface maintains control even in highly disturbed scenarios.

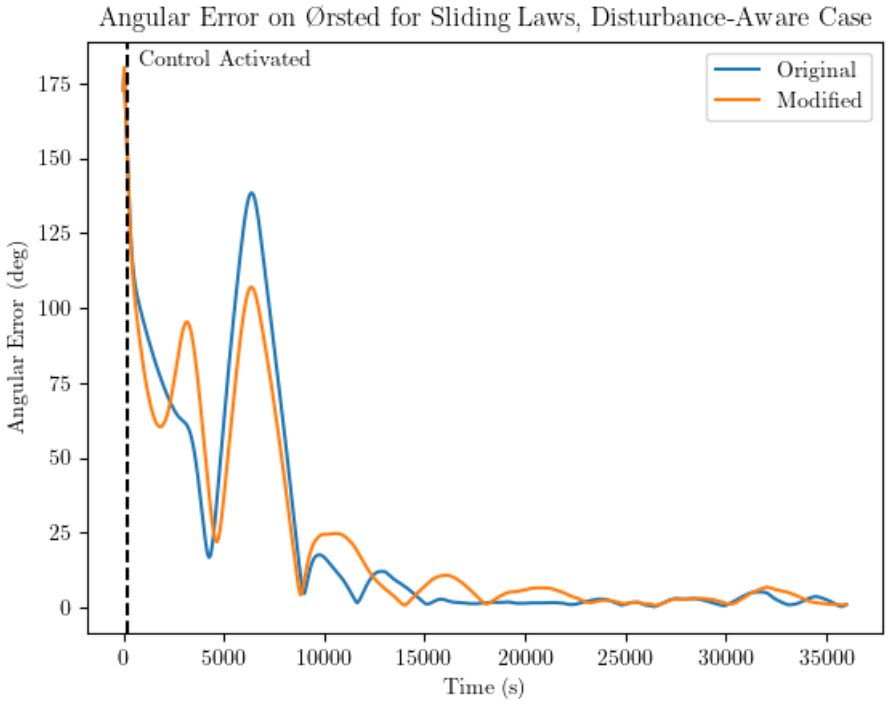


Figure 6.19: Comparison of angular error between Wisniewski's (blue) and Modified (orange) sliding control on a disturbed version of Ørsted. The blue case is based on test parameters in Table 6.4, and the orange case is identical but with the control law changed, as described in Section 6.5.1.d. Both cases use disturbance models. Wisniewski's original law (blue) stays between 0.5 and 3 degrees, while the modified law (orange) stays between 1 and 5 degrees.

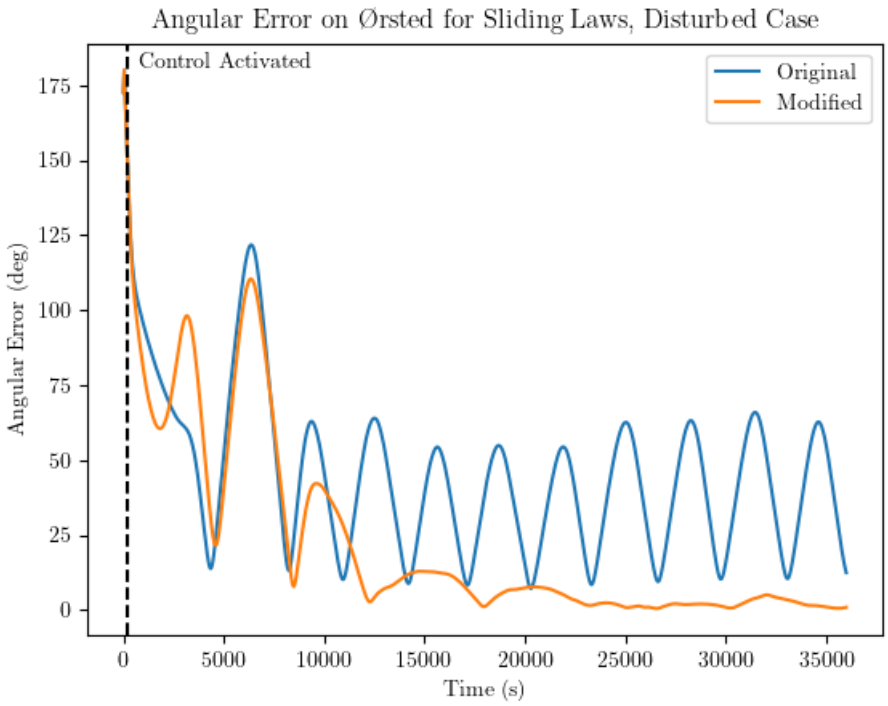


Figure 6.20: Comparison of angular error between Wisniewski's (blue) and Modified (orange) sliding control on a disturbed version of Ørsted. The blue case is based on test parameters in Table 6.4, and the orange case is identical but with the control law changed, as described in Section 6.5.1.d. Neither case accounts for the disturbances. Wisniewski's method does not converge, staying between 9 and 60 degrees, while the modified case stays between 1 and 5 degrees.

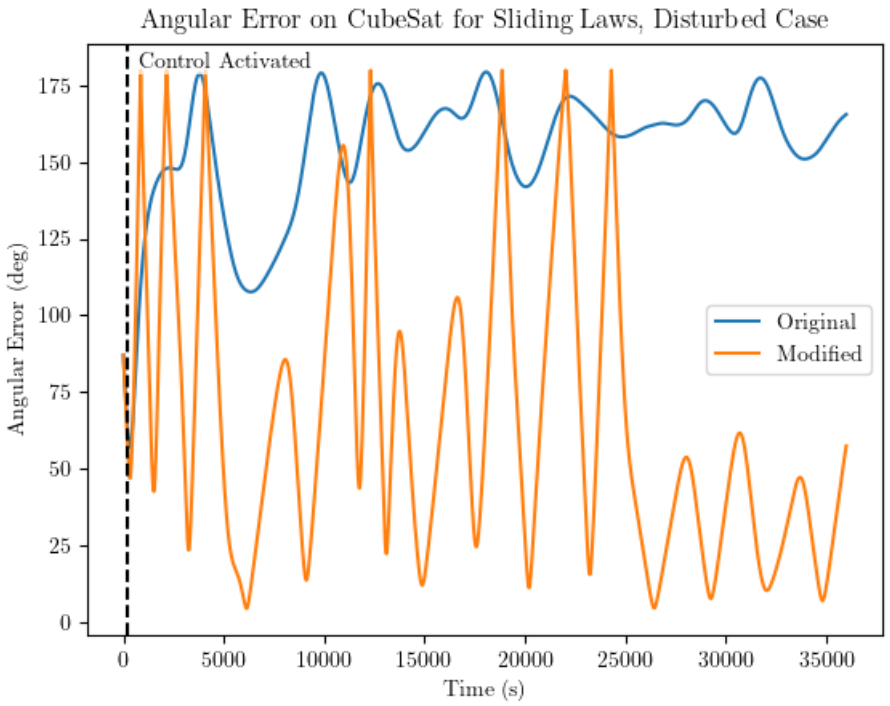


Figure 6.21: Comparison of angular error between Wisniewski's (blue) and Modified (orange) sliding control on a disturbed CubeSat. The blue case is based on test parameters in Table 6.5, and the orange case is identical but with the control law changed, as described in Section 6.5.2.c. Neither case accounts for the disturbances. Wisniewski's method does not converge, staying roughly 180 degrees off, while the modified case stays between 0 and 60 degrees.

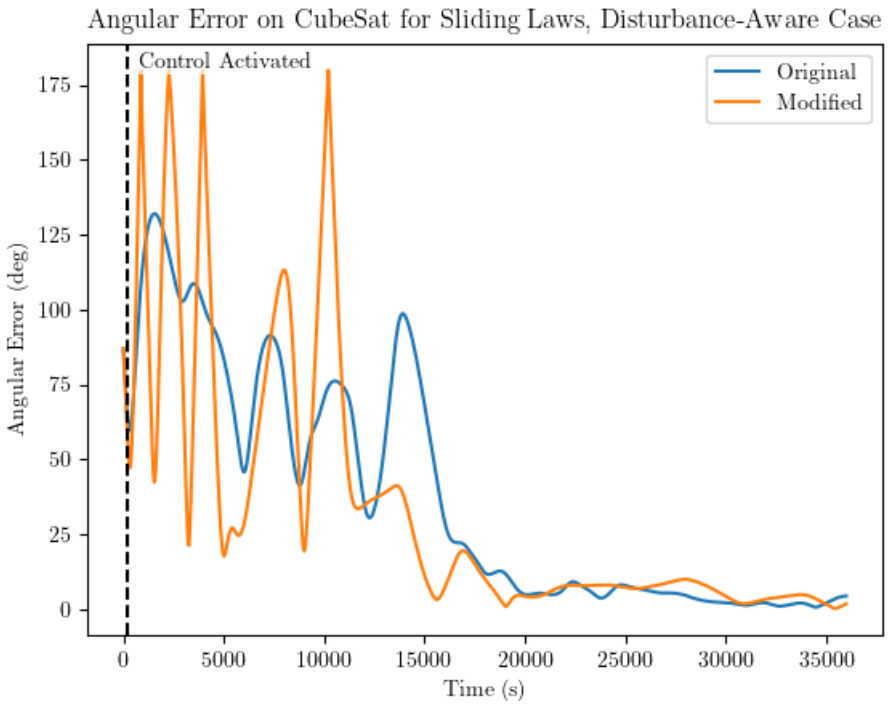


Figure 6.22: Comparison of angular error between Wisniewski's (blue) and Modified (orange) sliding control on a disturbed CubeSat. The blue case is based on test parameters in Table 6.5, and the orange case is identical but with the control law changed, as described in Section 6.5.2.c. Both cases use disturbance models. Both cases converge to single-digit angular errors, with the modified case converging slightly faster, after about 4.2 hours, compared to the original law's 5.6 hours.

6.6 Conclusion

This chapter presents a system for integrating active disturbance rejection into Attitude Determination and Control Systems (ADCS), addressing key challenges in countering environmental and internal disturbances that affect satellite stability and precision. By directly incorporating disturbances into control laws, the work explored in this chapter contributes to creating more robust and adaptable systems capable of functioning across a wide range of satellite architectures and mission scenarios.

Several existing control laws in the literature are adapted to include disturbance modeling and counter those disturbances with a feedforward term, as specified in Section 6.2. This can be applied to both analytical disturbance models and an innovative "all-in-one" disturbance torque estimation method. The all-in-one torque, described in Section 6.3, aggregates disturbances that cannot be individually modeled, offering a flexible approach to mitigating unpredictable effects on satellite attitude. The sliding surface control law is also generalized to better accommodate varying spacecraft (Section 6.4). Section 6.5 validates the framework through various test cases, highlighting significant improvements in disturbance rejection and overall control performance. For example, CubeSat scenarios demonstrate that the inclusion of disturbance models in sliding-mode control laws improves pointing accuracy by up to 20 degrees compared to baseline methods. Additionally, the all-in-one disturbance term allows underactuated systems, such as magnetic-only satellites, to achieve stable operation under high disturbance loads without requiring detailed knowledge of individual disturbance sources. Results from Lovera CubeSat tests further underscore the system's adaptability and resilience, demonstrating its potential for broader application in real-world missions.

While these advancements mark a significant step forward, future work could enhance the framework's applicability and effectiveness. Refining the disturbance modeling methods, particularly for non-linear and time-varying effects, would improve the accuracy and efficiency of the control laws. It is also worth investigating the effect of inaccurate disturbance models to understand the trade-off between model accuracy and system performance and determine if incorrect disturbance models could harm performance. A key aspect of such work would be identifying the cases under which the all-in-one disturbance estimation is ineffective. A potential extension would be relying on two all-in-one disturbances for satellites in a slew or other dynamic condition–one torque slowly varying in the body frame, the other in the world frame. This approach could also be extended beyond the comparably simple all-in-one method explored here. Using system identification and other, more complex methods to better understand the disturbance environment could improve stability when inaccurate disturbance models harm performance. Demonstrating this method on other existing control laws and systems, including reaction wheels or even multiple actuators (explored to some extent in Section 5.4), would be beneficial, as would understanding the effect of disturbance countering on momentum saturation (constantly countering a disturbance could cause a build-up of momentum). Hardware-in-the-loop testing and eventual flight demonstrations would validate the framework under real-world conditions, bridging the gap between simulation and operational readiness.

The concepts and work laid out in this chapter tie into the key components of this thesis. Starting from the areas of work in Section 3.2, the contributions in Section 3.3 build up

to address the technical gaps in Section 2.3.2. This chapter covers most of the **Direct Countering of Disturbance** area of work in Section 3.2.3 and is the most direct application of the **Disturbance Modeling for Control Inclusion** contribution (Section 3.3.2). By allowing satellites to understand and adapt to their disturbance environment in real-time, it plays a large part in the **Improved Autonomy for ADCS** contribution from Section 3.3.4 and could be used to tackle the **Autonomous Rapid Retargeting** technical gap in Section 2.3.2.d. The inclusion of disturbances (particularly if combined with goal sets and other modifications from Chapter 5) improves the performance of underactuated spacecraft or those with weak actuators, earning this work a part in the **ADCS Architecture for Weak or Underactuated Satellites** contribution (Section 3.3.5) and helps specifically for magnetically-controlled satellites by closing the **Effective Magnetic Control** technical gap (Section 2.3.2.b). Lastly, this work enhances the ability of ADCS (with a variety of control laws) to counter diverse disturbances autonomously, reducing reliance on mission-specific customization. This results from the **Generalized ADCS** area of work (Section 3.2.5) and contributes to closing the **General ADCS Approach** technical gap (Section 2.3.2.a).

The contributions in this chapter also support the trajectory planning methods explored in Chapter 7, where the ability to predict and counter disturbances enables more accurate and reliable path planning. By integrating disturbance rejection into both control and planning, this work advances the dissertation's overarching vision of a fully autonomous, generalizable ADCS framework. The methods and results presented here open new avenues for innovation, making satellite missions more robust and efficient, even in the face of challenging operational environments.

Chapter 7

Trajectory Planning

7.1 Introduction

Most current attitude control systems rely heavily on feedback control laws, which determine actuation based solely on the current state. While this approach has proven effective in many applications, it inherently limits autonomy, as discussed in Section 5.1. Feedback-based systems often address only immediate goals, such as maintaining or adjusting orientation, without considering longer-term objectives, like momentum desaturation or balancing competing mission priorities. Additionally, these systems are typically tailored to specific actuators and control modes, making them less generalizable to different satellite architectures, actuator configurations, or mission goals. Efforts to adapt feedback control for broader applications, such as the modifications described in Section 5.4, still fall short of addressing these fundamental limitations.

An alternative approach is trajectory planning, where the satellite's actuation is determined not solely by its current state but through the generation of a time-dependent path, or "trajectory," over a set interval. This trajectory specifies desired positions or orientations at each time step and optimizes the satellite's motion to balance multiple goals, such as pointing, slewing, and momentum desaturation. Such planning inherently improves autonomy, as it allows the satellite to adapt dynamically to mission demands with minimal external input. Moreover, trajectory planning is inherently generalizable: when integrated with a robust dynamic model like the one described in Section 4.3, it can optimize trajectories for virtually any set of actuators or control constraints. This adaptability enables the use of planners in underactuated systems (e.g., those with magnetic control), unconventional actuator combinations, or challenging environments. Trajectory planning also has the potential to leverage disturbances as part of the solution. For instance, a planner could schedule slews to counteract residual dipole torques with solar radiation pressure or account for magnetic underactuation by combining environmental forces intelligently. This expands the possibilities for mission design, making it feasible to achieve complex goals even with limited hardware capabilities.

These advantages directly address the **ADCS Trajectory Planning** (Section 2.3.2.c) and **General ADCS Approach** (Section 2.3.2.a) technical gaps, and enable improvements on the **Effective Magnetic Control** (Section 2.3.2.b) and **Autonomous Rapid Retargeting**

(Section 2.3.2.d) gaps as well.The content of this chapter largely stems from the **Attitude Trajectory Planning** area of work (Section 3.2.4), but there are also key components from the areas of work for **ADCS Operations and Goals** (Section 3.2.2) and **Generalized ADCS** (Section 3.2.5). It most directly creates the **Trajectory Planning** contribution, described in Section 3.3.3. However, it is also a cornerstone of the **Improved Autonomy for ADCS** contribution (Section 3.3.4) and enables the **ADCS Architecture for Weak or Underactuated Satellites** contribution (Section 3.3.5) as well.

The planner used in this work is based on the Augmented Lagrangian TRajectory Optimizer (ALTRO), introduced in Section 7.2. ALTRO provides a robust framework for trajectory optimization by balancing computational efficiency with the ability to enforce complex constraints. It enables the generation of trajectories that respect actuator limits, minimize control effort, and achieve mission goals across diverse environments. Incorporating a planner into an ADCS requires addressing several implementation details, discussed in Section 7.3. These include incorporating state estimation and following the path, addressing the timing of calling the planner and how to estimate information for its trajectory, and convergence questions. Once a trajectory is generated, it must be followed accurately, which introduces additional challenges in control implementation. These challenges, along with solutions, are explored in Section 7.4. Example trajectories demonstrate the system's ability to handle diverse scenarios, including long-duration planning, dynamic goal switching, and underactuation. These results are detailed in Section 7.5, with a summary of the test cases provided in Table 7.1.

7.2 ALTRO

The planner that forms the basis of this work is based on the ALTRO algorithm developed by Zac Manchester and his team. This has been applied to this problem before, as magnetorquer-only control by Gatherer and Manchester [87]. Their approach has been expanded to include disturbance torques and to focus on varying goals and the quaternion set. Their general algorithm was adjusted to focus on this problem and re-coded in C++ for faster operation onboard a spacecraft. A general approach for differing types of satellites and different control architectures is also developed. ALTRO solves for a trajectory on a nonlinear system by using an iterated LQR (iLQR)–linearizing about and optimizing on its trajectory at each point. Constraints are handled with an Augmented Lagrangian.The overall structure of the algorithm is shown in Figure 7.1. It is described in detail in Section 7.2.1, with more information on satellite-specific modifications found in Section 7.2.2.

7.2.1 ALTRO Overview

The Augmented Lagrangian Trajectory Optimization (ALTRO) method has been under development by Zac Manchester and his team for several years, and has gone through a number of iterations. The method used in this work is based on the algorithm described in works [87]–[89], [99]. It has since been supplanted by a more general formulation focused on re-writing the problems as conic problems, called ALTRO-C (the "C" is for conic). The overall method is similar. Large parts of this overview are based on the excellent tutorial [99].

Table 7.1: Summary of Planning Test Cases

Test Case	Satellite Details	Sec.	Purpose	Pointing Goals	Results & Notes
Long CubeSat Two Goal Trajectory	3U CubeSat, 3MTQ	7.5.1	Demonstrate trajectory planning to meet multiple goals in weak & underactuated satellite	Track ground target (960s-1440s), point propulsion anti-ram (2400s on)	Successfully planned a feasible trajectory. Handled underactuation and weak control.
Planning Around Disturbance	3U CubeSat, 3MTQ+1RW Body-fixed Disturbance	7.5.2	Showcase flexible goals & trajectory planning generating novel solutions	Point propulsion anti-ram	Large torque countered effectively via rotation about goal axis.
Monte Carlo Simulations	3U CubeSat, 3MTQ+1RW	7.5.3	Demonstrate advantage of 3MTQ+1RW architecture, test robustness	· Specific orientation · Align body & world vectors · Multi-goal	Faster convergence, lower pointing error due to RW. Improved overall stability.
	3U CubeSat, 3MTQ	7.5.3	Evaluate limits of 3MTQ architecture, test robustness	· Specific orientation · Align body & world vectors · Multi-goal	Slower convergence, higher pointing error due to underactuation.
Sequentially-planned Trajectory	6U CubeSat, 3RW	7.5.4	Demonstrate sequential trajectory planning	· $-x$ anti-ram (150s-1100s) · z nadir (1200s-1500s) · z zenith (1600s-1900s) · z orbit normal (2000s-2400s) · $-x$ anti-ram (2500s on)	Successfully planned successive, overlapping trajectories with stable performance under disturbances, incorporating new information & estimated state.

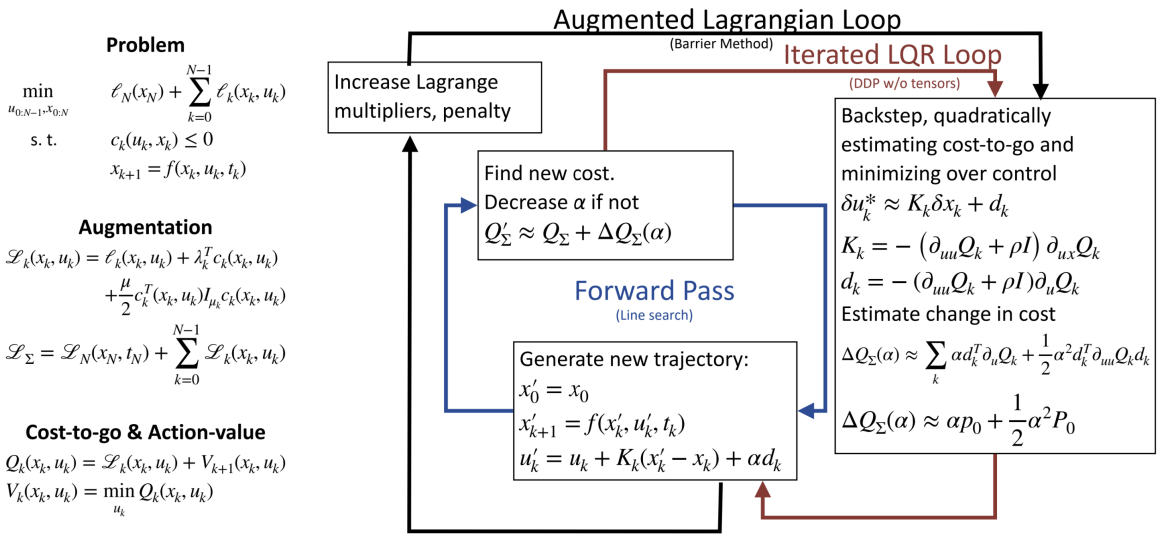


Figure 7.1: Overall description of how ALTRO works. Largely based on [99].

ALTRO provides a systematic framework for solving constrained trajectory optimization problems by combining augmented Lagrangian (AL) techniques with the computational efficiency of the Iterative Linear Quadratic Regulator (iLQR). The algorithm's overall structure is based on three nested loops that work together to minimize a given cost function for a trajectory, satisfy system dynamics, and enforce constraints.

7.2.1.a Problem Formulation

The trajectory optimization problem seeks to minimize a total trajectory cost over the control efforts u_k, subject to system dynamics and inequality constraints. Mathematically, the problem is expressed as:

$$\begin{aligned} \underset{\mathbf{u}_{0:N-1}}{\text{minimize}} \quad & \ell_f(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f_k(x_k, u_k),\ x_0 = x_{\text{init}}, \\ & c_k(x_k, u_k) \leq 0, \quad k = 0, \ldots, N-1. \end{aligned} \tag{7.1}$$

Here, $x_k \in \mathbb{R}^n$ represents the state at time k, $u_k \in \mathbb{R}^m$ is the control input, $f(x_k, u_k)$ describes the system dynamics, and $c(x_k, u_k) \leq 0$ encodes constraints. The cost function comprises the cost at step k, $\ell_k(x_k, u_k)$, and the terminal cost $\ell_N(x_N)$. In other words, the goal is to minimize the total trajectory cost

$$\ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \tag{7.2}$$

over the control vectors (u_k for $k = 0, \ldots, N-1$), subject to the discrete (or discretized via integration) system dynamics

$$x_{k+1} = f_k(x_k, u_k),\ x_0 = x_{\text{init}}, \tag{7.3}$$

and mixed equality and inequality constraints

$$c_k(x_k, u_k) \leq 0, \quad k = 0, \ldots, N-1. \tag{7.4}$$

The constraint function c_k and system dynamics f_k have a direct dependency on k to indicate they can vary over the course of the trajectory duration.

7.2.1.b The Three Nested Loops of ALTRO

The ALTRO algorithm employs a three-loop structure to solve this problem efficiently. This can be seen in Figure 7.1.

1. The Outer Loop: Augmented Lagrangian for Constraint Enforcement The outermost loop enforces the constraints $c_k(x_k, u_k) \leq 0$ using the augmented Lagrangian approach. The "AL" in "ALTRO" stands for "Augmented Lagrangian". From an optimization perspective, this is a barrier-method-type approach to enforcing the constraints. The baseline per-step cost function ℓ_k is augmented with terms involving Lagrange multipliers λ_k and quadratic penalties μ_k based on the constraints $c_k(x_k, u_k)$:

$$\mathcal{L}_k(x_k, u_k, \lambda_k, \mu_k) = \ell_k(x_k, u_k) + \lambda_k^T c_k(x_k, u_k) + \frac{\mu}{2} c_k^T(x_k, u_k) I_{k,\mu} c_k(x_k, u_k), \tag{7.5}$$

where $I_{k,\mu}$ is a diagonal matrix where the ith entry is unity, unless the ith element of $c_k(x_k, u_k) < 0$ and the ith element of $\lambda_k = 0$. Effectively, the penalty cost is not added if constraints are met and the Lagrange multiplier is off. The terminal cost ℓ_N is augmented similarly.

In each outer loop iteration, the black loop in Figure 7.1, the augmented Lagrangian parameters are updated (based on which constraints are still in violation) to progressively enforce constraint satisfaction:

$$\lambda_k^+ = \max(0, \lambda_k + \mu c_k(x_k, u_k)), \tag{7.6}$$

$$\mu^+ = \phi\mu, \quad \phi > 1. \tag{7.7}$$

The middle and inner loops run within this largest loop. They solve the trajectory optimization problem for the current values of λ_k and μ_k, and the process repeats until constraint violations are sufficiently small (or the optimization process terminates for other reasons).

2. The Middle Loop: Iterative LQR for Trajectory Optimization Within the augmented Lagrangian framework, the middle loop (red loop in Figure 7.1) solves the unconstrained trajectory optimization problem using iLQR. It is effectively a version of differential dynamic programming (DDP), though it ignores second-order effects to keep everything in the matrix space without relying on higher-order tensors. The iLQR approach linearizes the dynamics and approximates the cost as a quadratic around a nominal trajectory at each time step. The solution proceeds in two phases: the backward pass to determine the suggested control adjustments from a nominal trajectory, and the forward pass that contains the innermost loop where the control adjustments are applied.

‘

Backward Pass: Local Quadratic Approximation The backward pass starts at the terminal time step N and propagates backward through the trajectory. We define the value function (or "cost-to-go") as the cost of being at our given state and propagating the dynamics forward, choosing our control at each step to minimize the remaining cost-to-go. We start with the terminal cost-to-go of $V_N(x_N) = \mathcal{L}_N(x_n)$.[1] It is defined recursively as minimizing the sum of the current step cost and the cost-to-go of the state (dependent on current control) as

$$V_k(x_k) = \min_{u_k} \left(\mathcal{L}_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k))\right) = \min_{u_k} Q_k(x_k, u_k). \tag{7.8}$$

The value $Q_k(x_k, u_k)$ is the action-value function. It is the value being minimized at each step, describing the value (or cost) of the rest of the trajectory, based on the current state and action (control). We have omitted the augmented Lagrangian parameters from $\mathcal{L}_k(x_k, u_k)$ for clarity, as when performing the iLQR, we are working fully inside a single step of the augmented Lagrangian loop and those parameters are treated as constants.

To minimize the trajectory over potential control efforts, we take each step and approximate the action-value and cost-to-go functions quadratically. Working backwards from the end, we minimize the action-value approximation to define the cost-to-go approximation. At each backwards step, the action-value Q_k is approximated as

$$Q_k(x_k + \delta x_k, u_k + \delta u_k) \approx \frac{1}{2}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} + Q_k(x_k, u_k) \tag{7.9}$$

and the cost-to-go V_k is approximated

$$V_k(x_k + \delta x_k) \approx \frac{1}{2}\delta x_k^T P_k \delta x_k + p_k^T \delta x_k + V_k(x_k). \tag{7.10}$$

The values Q_{xx}, $Q_{ux} = Q_{xu}^T$, and Q_{xx} form the Hessian of next-step action-value Q_{k+1} (indices omitted for clarity) with respect to x_k and u_k, while Q_x and Q_u are the Jacobians of next-step action-value Q_{k+1} with respect to x_k and u_k. The values P_k and p_k are the Hessian and Jacobian, respectively, of the cost-to-go at time k. The action-value derivatives are calculated from Equation 7.5, the system dynamics, and the baseline cost functions ℓ. Specifically,[2] for each time index k (omitted for clarity),

$$\begin{aligned}
Q_{xx} &= \frac{\partial^2}{\partial x_k^2}\left(\mathcal{L}_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k))\right) \\
&= \ell_{xx} + \mu c_x^T I_\mu c_x + f_x^T P_{k+1} f_x && (7.11) \\
Q_{xu^T} = Q_{ux} &= \ell_{ux} + \mu c_u^T I_\mu c_x + f_u^T P_{k+1} f_x && (7.12) \\
Q_{uu} &= \ell_{uu} + \mu c_u^T I_\mu c_u + f_u^T P_{k+1} f_u && (7.13) \\
Q_x &= \ell_x + c_x^T\left(\lambda_k + \mu I_\mu c_k(x_k, u_k)\right) + f_x^T p_{k+1} && (7.14) \\
Q_u &= \ell_u + c_u^T\left(\lambda_k + \mu I_\mu c_k(x_k, u_k)\right) + f_u^T p_{k+1} && (7.15)
\end{aligned}$$

[1] This could be equivalently defined, based on Equation 7.8, as $V_{N+1} = 0$.

[2] Also note the lack of second-order tensor derivatives of vector functions. That is where the iLQR differs from DDP.

where $g_{xx} = \frac{\partial^2}{\partial x_k^2} g_k(x_k, u_k)$ for functions of the form $g_k(x_k, u_k)$, $g_{uu} = \frac{\partial^2}{\partial u_k^2} g_k(x_k, u_k)$, $g_{ux} = g_{ux}^T = \frac{\partial^2}{\partial u_k \partial x_k} g_k(x_k, u_k)$, $g_u = \frac{\partial}{\partial u_k} g_k(x_k, u_k)$, and $g_x = \frac{\partial}{\partial x_k} g_k(x_k, u_k)$. These derivatives can be carefully calculated numerically (or using autodifferentiation), but the approach used in this dissertation calculated them directly (including RK4 dynamics).

To find the cost-to-go (the minimum of the action-value) and its derivatives P_{k+1} and p_k, we find the value of δu_k that minimizes the expansion in Equation 7.9, and label it δu_k^*. By taking the first derivative, we see that the optimal control adjustment (for each time step) to minimize the cost-to-go approximation is given by:

$$\delta u_k^* = -Q_{uu}^{-1}(Q_{ux}\delta x_k + Q_u), \tag{7.16}$$

which consists of a linear feedback gain $-Q_{uu}^{-1}Q_{ux}\delta x_k$ and a feedforward term $-Q_{uu}^{-1}Q_u$. Once backstepping is completed, and the trajectory is re-generated based on this information (in the forward pass), the control effort at each step will be modified from its prior value based on the feedforward term and will vary additionally according to state changes δx_k, through the feedback gain. This accounts for the fact that, during the forward pass, adjustments made at early time steps will result in subsequent times on the trajectory having different states than originally optimized around. The linear term is included to account for this adjustment.

While not mathematically optimal, additional linearizing term, ρ, is added to deal with nonlinearities and noisy behavior. It is increased and decreased based on how effective the algorithm is at each time (increased when diverging to approximate gradient descent, decreased to focus more on quadratic terms). The actual formulation with this included is

$$\delta u_k^* = -(Q_{uu} + \rho I)^{-1}(Q_{ux}\delta x_k + Q_u) \equiv K_k \delta x_k + d_k. \tag{7.17}$$

We can now minimize Equation 7.9 and equate it with Equation 7.10. This gives us

$$\begin{aligned}
V_k(x_k + \delta x_k) &= \min_{\delta u_k} Q_k(x_k + \delta x_k, u_k + \delta u_k) && \text{(See Eq. 7.8)} \\
&= Q_k(x_k + \delta x_k, u_k + \delta u_k^*) && \text{(Def. of } \delta u_k^*\text{)} \\
&\approx \frac{1}{2}\begin{bmatrix}\delta x_k \\ \delta u_k^*\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k^*\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ \delta u_k^*\end{bmatrix} + Q_k(\delta x_k, \delta u_k) && \text{(See Eq. 7.9)} \\
&= \frac{1}{2}\begin{bmatrix}\delta x_k \\ -Q_{uu}^{-1}(Q_{ux}\delta x_k + Q_u)\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}\delta x_k \\ -Q_{uu}^{-1}(Q_{ux}\delta x_k + Q_u)\end{bmatrix} \\
&\quad + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ -Q_{uu}^{-1}(Q_{ux}\delta x_k + Q_u)\end{bmatrix} + Q_k(x_k, u_k) && \text{(See Eq. 7.17)} \\
&= \frac{1}{2}\delta x_k^T \left(Q_{xx} - Q_{ux}^T Q_{uu}^{-1} Q_{ux}\right)\delta x_k + \left(Q_x - Q_{ux}^T Q_{uu}^{-1} Q_u\right)^T \delta x_k \\
&\quad + Q_k(x_k, u_k) - Q_u^T Q_{uu}^{-1} Q_u && (7.18)
\end{aligned}$$

By comparing Equations 7.18 and 7.10, we can find our definitions of P_k and p_k. Specifically,

$$P_k = Q_{xx} - Q_{ux}^T Q_{uu}^{-1} Q_{ux} \tag{7.19}$$

$$p_k = Q_x - Q_{ux}^T Q_{uu}^{-1} Q_u \tag{7.20}$$

and by extension to the final time step,

$$P_N = \ell_{N,xx} + \mu c_{N,x}^T I_{N,\mu} c_{N,x} \tag{7.21}$$

$$p_N = \ell_{N,x} + c_{N,x}^T \left(\lambda_N + \mu I_{N,\mu} c_N(x_N)\right). \tag{7.22}$$

These values can be used in Equations 7.11-7.15, allowing us to complete our minimization and providing all information needed for the backwards pass. The middle loop, the red loop in Figure 7.1, iterates between backward and forward passes (which contains the innermost loop) until the trajectory cost stabilizes, indicating convergence for the current augmented cost function.

There is, however, one last piece of information that is important for the forward pass. When the control adjustments in Equation 7.17 are made and the trajectory is re-propagated, the new trajectory will differ from the old one. We can estimate the cost of the new trajectory (subject to our approximations in Equations 7.9 and 7.10) by looking at the effect of each feedforward gain. If we take Equation 7.9 and substitute in the ideal control adjustment from Equation 7.17, we find

$$\begin{aligned} Q_k(x_k + \delta x_k, u_k + \delta u_k^*) \approx \frac{1}{2} \begin{bmatrix} \delta x_k \\ K_k \delta x_k + d_k \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x_k \\ K_k \delta x_k + d_k \end{bmatrix} \\ + \begin{bmatrix} Q_x \\ Q_u \end{bmatrix}^T \begin{bmatrix} \delta x_k \\ K_k \delta x_k + d_k \end{bmatrix} + Q_k(x_k, u_k). \end{aligned}$$

Because Q_k, as defined in Equation 7.8, includes V_{k+1}, the effects of future δx based on this δu_k^* are already accounted for. The expected change in trajectory cost from the control adjustment for each time step can be found by setting δx_k to zero and subtracting off the baseline original trajectory

$$Q_k(x_k, u_k + \delta u_k^*) - Q_k(x_k, u_k) \approx \frac{1}{2} d_k^T Q_{uu} d_k + d_k^T Q_u.$$

By summing the cost change attributable to each δu_k^*, we can find the expected overall cost change to be

$$\Delta V_0 = V_0 \sum_k Q_k(x_k, u_k + \delta u_k^*) - Q_k(x_k, u_k) \approx \sum_k \frac{1}{2} d_k^T Q_{k,uu} d_k + d_k^T Q_{k,u}. \tag{7.23}$$

This is used to determine if the forward pass was successful (and deal with nonlinearities) in the third and innermost loop.

The iLQR has an estimated change in trajectory cost based on these derivative approximations, which is used to govern the innermost loop–the cost change in the innermost loops should approximate the estimated cost change, within certain bounds, to ensure the system stays within regions that the iLQR reasonably approximates.

3. The Inner Loop: Forward Pass Line Search The third and innermost loop, the blue loop in Figure 7.1, is a forward pass which updates the nominal trajectory using the control corrections obtained in the backward pass from Equation 7.17:

$$u_k^{\text{new}} = u_k + \alpha d_k + K_k \delta x_k, \tag{7.24}$$

$$x_{k+1} = f_k(x_k, u_k^{\text{new}}), \tag{7.25}$$

where α is a scaling factor and δx_k is the difference in the value of x_k between this and the prior iteration (i.e., the difference between the x_k value calculated in the forward pass and the value that it held during the backward pass). The forward pass operates as a loop by running a linesearch over the value of α. This helps stabilize the optimization. The backward pass provides an adjustment direction for the controls, but large corrections (or the stacked effects of corrections over time throughout the trajectory) can cause instability or divergence. To address this, the control update is scaled by α, where:

$$\alpha \in \{1, 0.5, 0.25, \dots\}. \tag{7.26}$$

With this scaling of d_k by α, the expected cost change in Equation 7.23 becomes

$$\Delta V(\alpha) \approx \sum_k \frac{1}{2}\alpha^2 d_k^T Q_{k,uu} d_k + \alpha d_k^T Q_{k,u}. \tag{7.27}$$

Starting with $\alpha = 1$, the algorithm evaluates the updated trajectory and compares the actual cost reduction to the expected decrease computed in the backward pass, as described in Equation 7.27. If the trajectory does not reduce cost sufficiently (or reduces it by too much), α is halved,[3] and the trajectory is re-evaluated. (The limits which define the acceptable bounds are hyperparameters of the optimization that can be adjusted.) This is repeated (blue loop in Figure 7.1) until the new trajectory that is output has a cost change that is approximated correctly (typically with wide bounds) by Equation 7.27. This ensures the updates remain within the region where the iLQR approximations are valid.

7.2.1.c Summary of the Algorithm

The ALTRO algorithm integrates these three loops, shown in Figure 7.1, in a structured manner:

- The Augmented Lagrangian **outer loop** progressively enforces constraint satisfaction by updating the Lagrange multipliers and penalties.
- The iterated LQR **middle loop** minimizes the augmented cost function using iLQR, iteratively refining the state and control trajectories.
- The forward pass **inner loop** performs a line search by uniformly scaling the iLQR feedforward gains to ensure stability and cost reduction during the forward pass.

Together, these loops enable ALTRO to efficiently solve constrained trajectory optimization problems, balancing computational speed with robust constraint enforcement. By leveraging the recursive nature of iLQR and the progressive tightening of constraints through the augmented Lagrangian, ALTRO achieves high performance even for complex, high-dimensional systems [89].

[3] An alternate scaling between 0 and 1 can also be used.

7.2.1.d Initializing ALTRO

ALTRO requires a single starting state and an initial guess for the control sequence over the trajectory. The minimal requirement is that the initial state x_0 be defined, while the control sequence $\mathbf{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ can be initialized arbitrarily—commonly using random values. These initial control inputs do not need to satisfy the system dynamics or constraints, and the resulting state trajectory obtained by propagating these controls forward serves as the starting point for the optimization process. While random initialization is often sufficient, ALTRO can also be warm-started with a more informed trajectory guess. This might include control inputs derived from a heuristic, a prior solution, or even a simple control law, such as a simulation using a feedback controller like B-dot for attitude stabilization.

Although not used in this work, extensions to ALTRO allow for additional flexibility in initialization through the use of slack variables. These slack variables can accommodate dynamically infeasible state trajectories as initialization inputs, enabling the user to specify desired state sequences directly without requiring them to correspond to valid control efforts. By adding these slack inputs to the dynamics, ALTRO can iteratively drive the infeasible initial trajectories toward a dynamically and constraint-compatible solution.

7.2.2 Satellite-Specific ALTRO Details and Modifications

The approach described in Section 7.2.1 can be (and has been) applied to a wide variety of trajectory-based problems. For use in onboard flight software (in the mission described in Section 2.4), some modifications were made, as well as various choices in moving the general optimization algorithm to a specific implementation. This section describes some of these choices. The most important questions are how we define the system, cost, and constraints that the optimizer works on. While these questions are part of the basic algorithm, there are many other choices to be made in the creation of such a system, as well as some potential for modifications.

7.2.2.a Dynamics

The system dynamics are simple enough, as detailed in Sections 3.4.2 and 3.4.1. Which disturbances are included in the dynamics (and whether any are at all) is variable. The equations of motion are turned into state-transition equations using the fourth-order Runge-Kutta (RK4) method. In general, the time steps are small enough and desired rates are slow enough that this is satisfactory. Auto-integration and variable-time-step methods could also work, but using RK4 allows for the dynamics Jacobians to be calculated analytically, ensuring a consistent runtime for the optimizer.

7.2.2.b Constraints

There are a number of constraints. The simplest are those placed on the actuators–any actuator has a maximum effort value, and these are represented in the ALTRO implementation. Generally, however, the planner is given tighter bounds than the actuators would permit–this allows for there to be some "room" for correcting and flying the trajectory. If the optimizer outputs trajectories that require actuators to be at their maximum value, and disturbances

or uncertainty then drive the system, which is trying to follow the trajectory, to a state such that correcting it required further effort in the "maxed-out" direction, there would be no recourse. The scaling of the actuator magnitude is variable (often set in tests to be 0.3 or 0.5 of the maximum value). The other constraints that are already programmed are simple–there is a "keep-out" zone constraint that forbids trajectories that point the camera towards the sun (this could easily be generalized to multiple angular keep-out zones with different body and world directions). Specifically, the constraint is on the orientation quaternion at time step k, $\bar{q}_k$, such that

$$\hat{\mathbf{a}}^T A(\bar{q}_k)\hat{\mathbf{s}}_k \leq \cos\eta, \tag{7.28}$$

where η is the angular limit of the keep-out zone, $\hat{\mathbf{a}}$ is the body-frame unit vector aligned with the camera (or whatever must be kept away from the sun), and $\hat{\mathbf{s}}_k = \frac{\mathbf{S}_{\text{ECI,k}}}{\|\mathbf{S}_{\text{ECI,k}}\|}$ is the normalized unit vector in the direction of the sun (in the ECI frame) at time k, $\mathbf{S}_{\text{ECI,k}}$. Another constraint is a maximum angular velocity, such that $\|\boldsymbol{\omega}\|^2 \leq \omega_{\max}^2$, which keeps the satellite from spinning too quickly. For many of the tests in this work, $1^\circ/\text{s}$ is used, but higher or lower values may be needed based on the spacecraft. Similarly, if the spacecraft has reaction wheels, there are maximum values for the stored momentum on each. Like the control effort bounds, these are often set to a fraction of the technical limits to give the post-planning control system some room to adjust as needed. The bounds for reaction wheel i are $h_i \leq h_{i,\max}$.

7.2.2.c Cost

The last detail that must be specified is the cost function. Effectively, it is a formula that takes the trajectory and scores how good it is. A high cost is bad, a low cost is good. The metric used to score it can vary and be adjusted based on what is needed (including weighting cost components in different ways). The approach used throughout this thesis is described in this section, but different versions could be used. It is entirely possible to vary the cost function, or simply the weights of its components, based on the mission, goal, or even satellite operational mode. This could allow for high pointing accuracy during mission-critical segments, but stable, efficient, and low-power trajectories during idle periods. The weights or cost function can even vary over time in a given trajectory–for example, a tight pointing requirement could be used during an imaging pass but no weight on pointing before and after. Similarly, a low angular velocity cost could be used during transition or idle periods, and a tighter one used during an imaging period to reduce blurring. There are many options to adjust, giving a great deal of flexibility in the cost function.

Our cost function is broken down into four subcosts: state cost (ℓ_x), actuation cost (ℓ_u), angular momentum cost (ℓ_H), and stiction cost (ℓ_s). The state cost is based on the cost function proposed for magnetorquer-only satellites by Gatherer and Manchester [87], with an additional term ($c_{q\omega}$) that provides directional angular velocity damping, increasing cost when the angular velocity is such that the pointing error increases, decreasing cost when the error is decreasing, and zero if the angular velocity and pointing error are perpendicular.

$$\ell_x(x_t, q_{goal}, c_\omega, c_q, c_{q\omega}) = \frac{1}{2}\omega_t^T \omega_t c_\omega + \ell_q c_q - \text{sign}(q_{goal} \cdot q_t) q_{goal} W(q_t) \omega c_{q\omega} \tag{7.29}$$

where ℓ_q is a method of measuring attitude error. In the case that the desired orientation is a fully specified quaternion q_{goal}, the value is

$$\ell_q = (1 - |q_{goal} \cdot q_t|). \tag{7.30}$$

However, as detailed in Chapter 5, many attitude goals do not require a full orientation being specified, with the goal instead to align two vectors. In such a case, if attempting to align body unit vector $\hat{\mathbf{a}}$ with world-frame unit vector $\hat{\mathbf{b}}$, the cost is

$$\ell_q = f\left(\hat{\mathbf{a}}^T A(\bar{q})\hat{\mathbf{b}}\right), \tag{7.31}$$

where $f(x)$ can be a range of functions, including $\arccos(x)$, $(1-x)$, and $\frac{1}{2}(1-x)^2$, with some small adjustments for near-zero cases. The most computationally efficient option is $f(x) = (1-x)$, the differences mainly being in computational cost and how the cost grows as the angular error does. A similar adjustment can be made for the cross-state cost term for c_{qw}.

There are many adjustments that affect the cost function in Equation 7.29. For example, there may be an advantage in setting the control weights for the final timestep to be larger (or, in theory, smaller) than prior time steps. Additionally, it may be necessary to set the angle/pointing control weights to zero if the planner is in a "no goal" time period (constraints and other costs do still apply). The angular velocity used in Equation 7.29 could be an error term between the trajectory angular velocity and some specific goal, but in many of the test cases the "goal" angular velocity is set to zero to avoid having to pre-specify it (and potentially unnecessarily constrain the solution). This is particularly important if the angle cost is reduced as described in Equation 7.31, as setting a body-frame angular velocity might be wrong, sub-optimal, and will unnecessarily constrain the problem. The cost terms could be adjusted to use weight matrices (rather than scalars) to penalize some directions more than others. Lastly, the cost functions could be modified to have higher or lower weights in some ranges–for example, when imaging targets that are smaller than the satellite FOV, this could be adjusted to allow for pointing costs to be zero when pointing within the FOV (or a central zone of it), not only when the pointing is exact. Similarly, a higher angular velocity (or perhaps a quadratic term) could be used if velocity is sufficiently high, to prioritize de-tumbling when necessary. Some versions of adjusted, nonlinear cost functions can be found later in this section, regarding reaction wheels.

As described later in this chapter, it can be valuable to run multiple iterations of the trajectory planner, using the output of the previous one as the input for the next. A non-zero value for $c_{q\omega}$ proves especially useful in early iterations (when the initializing trajectory might be wildly wrong or even random), as it rewards guiding the system towards a reasonable trajectory. It also works for shorter trajectories where the system may not be able to reach the goal yet, but can set the system heading in the correct directions.

Additionally, we have a control cost, which weights the use of the magnetorquers and the reaction wheel torque. As seen in Equation 7.32, the control cost is equivalent to $u^T R u$, the quadratic control cost used in linear quadratic regulator (LQR) control:

$$\ell_u(u_t, c_{MTQ}, c_{RW}, c_u) = \frac{1}{2} u_t^T \operatorname{diag}([c_{MTQ}\ c_{RW}]^T) u_t c_u. \tag{7.32}$$

Finally, there are two additional costs relevant to reaction wheels. We need to avoid cases where the reaction wheels saturate at high speeds and where "stiction" may occur at near zero. When run, the optimization problem has constraints keeping the RW angular momentum beneath its maximum momentum, but we also incur a cost as it gets closer to the angular momentum boundaries–this is to prevent an operational situation where the satellite is trying to fly a trajectory which includes maximized angular momentum in one direction, leaving the satellite unable to correct for disturbances in that direction. Additionally, the more angular momentum that is stored, the worse the integration error in the planner, controller, and estimator gets, so small angular momentum is good. However, we do not want to penalize having any angular momentum, so we use a modified quadratic cost. One form of this could be

$$\ell_H(x_t, c_{H_1}, c_{H_2}) = \frac{1}{2} c_{H_1} \left(\max\left(0, |H| - c_{H_2}\right)\right)^2, \tag{7.33}$$

where c_{H_1} is the weighting of the cost and c_{H_2} is a threshold–when the magnitude of the angular momentum is less than this threshold (when the momentum is either positive or negative), there is no penalty. But once the angular momentum's magnitude passes that threshold (perhaps $\frac{2}{3}$ of the maximum), there is a quadratic cost. The use of the max function can present issues for the optimization algorithm, as the derivative is not continuous and there is a large flat "deadzone." To address these issues, the "softplus" function is used, which approximates the max function continuously and is sometimes used as an activation function in deep neural networks because its derivative is relatively smooth. Thus, adding c_{H_3} as a normalization parameter, the angular momentum cost is:

$$\ell_H(x_t, c_{H_1}, c_{H_2}, c_{H_3}) = \frac{1}{2} c_{H_1} \left(\frac{\log(1 + \exp((|H| - c_{H_2}) c_{H_3})}{c_{H_3}} \right)^2. \tag{7.34}$$

The other reaction wheel cost is a cost to prevent "stiction." Avoiding saturation in reaction wheels is important, limiting how fast they can spin and how much momentum they can store. However, there is also the case where the reaction wheel can spin too slowly. Specifically, if a reaction wheel stops or spins too slowly for too long, there is the risk of it sticking due to friction and internal dynamics. This effect is called "stiction" and can prevent a reaction wheel from starting up again. To deal with this (while preventing the issues with non-continuous functions as previously mentioned), we use a small cost based on the "smoothstep" function and weighted by c_{s_1}, which increases cost if the angular momentum is below a threshold (c_{s_2}). Specifically, we use

$$\ell_s(x_t, c_{s_1}, c_{s_2}) = \frac{1}{2} c_{s_1} \left(\text{smoothstep}\left(\frac{c_{s_2} - |H|}{c_{s_2}} \right) c_{s_2} \right)^2 \tag{7.35}$$

where smoothstep(x) is a smoothed step function:

$$\text{smoothstep}(x) = \begin{cases} 0 & x \leq 0 \\ 3x^2 - 2x^2 & 0 < x < 1 \\ 1 & x \geq 1 \end{cases} \tag{7.36}$$

It should be noted that this cost should be kept low, as it should not be a barrier that prevents the reaction wheel from crossing past zero angular momentum when necessary. The effect of these last two RW-based cost elements can be seen in Figure 7.2.

$$\ell_h\left(\vec{h}\right) = \frac{1}{2}c_{h_1}\left(\text{softplus}\left(0, \left|\vec{h}\right| - c_{h_2}\right)\right)^2 + \frac{1}{2}c_{s_1}\left(\text{smooth step}\left(c_{s_2} - \left|\vec{h}\right|\right)\right)^2$$

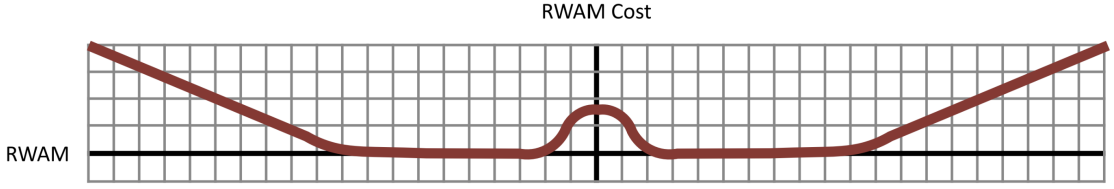


Figure 7.2: Graph illustrating how angular momentum cost ℓ_H and stiction cost ℓ_s vary with stored momentum on a reaction wheel.

The overall step cost functions for the satellite are

$$\ell(x_t, u_t, q_{goal}) = \ell_x(x_t, q_{goal}) + \ell_u(u_t) + \ell_H(x_t) + \ell_s(x_t) \tag{7.37}$$

$$\ell_f(x_t, q_{goal}) = \ell_x(x_t, q_{goal}) + \ell_H(x_t) + \ell_s(x_t). \tag{7.38}$$

The last time step does not have a control cost, as we are only looking at the state in the last time step, not the control for subsequent time steps.

As described previously in this section, the cost function can be readily changed to different functions or different weights, even varying between modes or over time within a given trajectory. The method suggested here (and the notes about values) is a good starting point. Adjusting them can improve performance and is encouraged. In doing so, however, the problem of "fighting" cost terms should be kept in mind–for example, it is possible for the optimizer to be "torn" between keeping angular velocity low and allowing for slews to meet pointing goals. It is wise to keep what is most important (such as pointing) in mind and prioritize that in the cost function. Constraints can be used to enforce other issues instead of the cost function (for example, an angular velocity constraint to prevent high-rate spinning). Adjusting things using the cross terms or weighted matrices (rather than scalars) can also be used to reduce the issue of fighting cost terms.

7.2.2.d Quaternion Inclusion

Many of the mathematical details that underpin ALTRO, AL-iLQR, and other optimization algorithms work best in Cartesian geometry. When working on attitude, there are difficulties that arise from including quaternions in the state. These difficulties are myriad–there is the issue of double cover (q and $-q$ represent the same rotation–this is dealt with in problems throughout this dissertation by forcing the scalar component to be non-negative), their inherent nonlinearity, the issue that they must be normalized to accurately represent rotation, and more. Many of these issues are described in Section 3.4.1. Unexplored in that section, however, is a difficulty that is specific to the use of quaternions in optimization: they are a four-dimensional set of numbers representing a three-dimensional orientation. Explored further (incredibly in-depth) in [24], there is the issue of what this means for a derivative. If you take the derivative with respect to each quaternion element, you end up with a 4D derivative, though the set of acceptable quaternions in our usage (SO(3)) is only 3D. This issue is specifically explored in [89], and their approach is generally used for Jacobians and

Hessians of the cost and constraint functions, as well as the state error term used in the forward pass.

The other issue that must be directly addressed is the requirement of keeping quaternions to unit norm. This is particularly a problem in the numeric integration for the time steps in trajectory generation, where methods like RK4 are used and treat the quaternion as a vector. (There is also a potential issue for the derivative in that case to be over-dimensioned due to the 3D vs 4D issue.) There are other algorithms that deal with the quaternion in a more mathematically correct manner, such as those suggested by Crouch and Grossman. The relative accuracy of these various methods is explored in [132], which shows that the difference in such approaches are small with a small timestep, and would be much smaller than other sources of error in our approach. Thus, we use RK4 and treat the quaternions as a four-vector. We normalize after integration (and in between each RK4 step), and include the normalization when calculating Jacobians and Hessians.

7.2.2.e Other Details

Among other modifications, we ported ALTRO from Julia to C++, to be more specialized for the specific use case and better ready to run onboard a CubeSat. Additionally, we switched away from Julia's autodifferentiation, instead using hard-coded functions for the Jacobians and Hessians of the various cost, dynamics, and constraint functions. This may not be the most efficient, especially when combined with propagating them through RK4 and quaternion normalization, but it results in a consistent function runtime. Additionally, the projection phase (last component, not described in this work) of ALTRO was eliminated in many tests to speed up the solver and improve usability on embedded hardware on CubeSats.

During development, many tweaks to the algorithm were explored in an attempt to improve performance or understand certain behaviors (many of these were revealed to be caused by software bugs, so most of these tweaks were not used later). One example of these tweaks is extending the iLQR approximations to a full differential dynamic programming (DDP) approach, including the second-order tensor components. An exponential angle cost function was explored, as was using a guessed value of desired ω. Various alternatives to the internal settings of the AL-iLQR algorithm were explored, such as a regularization bump function and injecting randomness in new trajectories, in an effort to wed this approach with simulated annealing.

One tweak that continues to be used (though a rigorous study of when it is advantageous should be part of future work) is running several iterations of the planning algorithm. This is implemented for two purposes. The first is to get a general approach using a coarser time step in the initial iteration (say, 10 second timesteps instead of 1 second). This allows for the system to find a general solution quickly and in a less-computationally-intensive way. This approach can also generate a rough trajectory over a longer time horizon, such as the next hour or orbit, incorporating longer-term goals into a detailed short-term trajectory plan with smaller time steps, which is then executed for navigation. While not explored in this dissertation, the system is designed to accommodate the inclusion of future goals and iterative runs, and it could prove beneficial in terms of setting up trajectories for goals that vary frequently. In the current implementation, there are also two passes of the short-time-step trajectory planner. The first is run without most disturbances, settling on a trajectory that

is optimal when there are not disturbances to be countered. This trajectory is then used as the starting pass of the final optimization, which has the disturbances added back in. There is the potential to adjust optimization parameters (such as initial penalty, penalty scaling rate, the different cost parameters, and more) between the various iterations, to speed convergence of later iterations and reduce computation time by, for example, starting with stricter constraints in the final pass.

7.3 Planning as Part of ADCS

When applied to a real mission, the planner will be just one part of the larger ADCS. This necessitates attention to important details that are not directly part of the planner, but must be considered when using the planner. One set of questions regards how to integrate the planner into a system with estimation and live control, as described in Section 7.3.1. When implementing this for a real-world system, there are also issues that must be dealt with such as accounting for computation time and determining what to do if convergence fails. These are further described in Section 7.3.2.

7.3.1 Planner-based ADCS Architecture

Even when restricted to those that include planning, the work for this dissertation includes a variety of ADCS architectures. For example, different control laws can be used to try to fly the planned trajectory, different estimation methods can provide input to the system, and so on. As discussed in Chapter 5, the systems used in this work even allow for different control laws at different times, including some that do not use planning. When such a system is operating as a planner, the architecture (somewhat simplified) is shown in Figure 7.3.

Figure 7.3 shows the three "blocks" of incorporating the planner into larger attitude determination and control system: the planner, the controller, and the estimator.

The estimator is an Unscented Kalman Filter (UKF) or other state estimation algorithm (such as those explored in Chapter 4) that combines information from environmental models (IGRF, celestial mechanics, etc.), the control command, and information from sensors to estimate the system state, using system properties and physical dynamics. In our case, that includes the orientation as a quaternion, q and the body-frame angular velocity $\boldsymbol{\omega}$, with the potential addition of any stored momentum, the disturbance torques, the residual magnetism of the satellite itself, and biases on sensors and actuators.

The state (propagated forward, as described in Section 7.3.2) is input to the planner block, alongside control goals (described in Chapter 5), disturbances, and the same environmental models. The planner block generates a trajectory that optimizes over the control goal's objective function, accounting for specified disturbances when possible and fulfilling dynamics and constraints. The planner block is the primary focus of this chapter. It outputs a trajectory over time, which is series of time points with specified states and control that satisfy the system dynamics. Potentially there is also control-adjustment information for each time step, such as the TVLQR gains in the approach used in [87] and described in Section 7.4.1.

The controller then tries to "fly" this trajectory in the real world, adjusting the actuation to try and follow the plan. It compares the expected state (from the planner) to the estimated

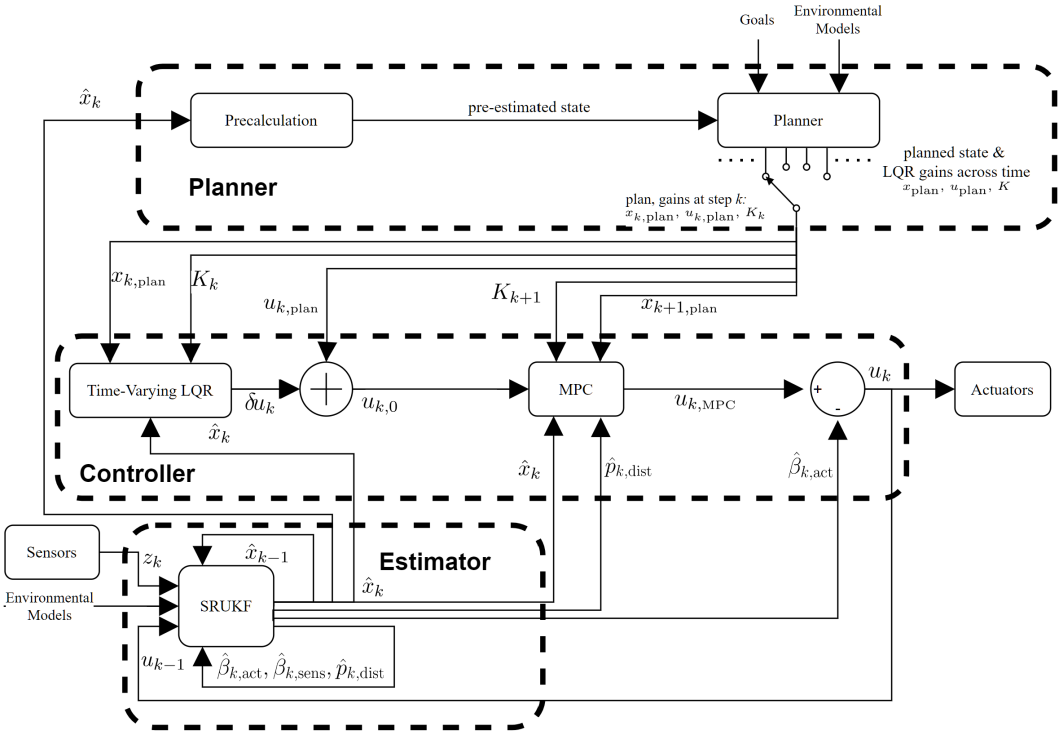


Figure 7.3: Overview of the architecture of the Attitude Determination and Control System (ADCS) as applied in this work. The system is structured around three core components: the estimator, the planner, and the controller. The estimator (such as an Unscented Kalman Filter (UKF)) integrates environmental models, sensor data, and actuator commands to provide real-time state estimation. The planner utilizes the estimated state and mission goals to generate optimized time-varying trajectories, accounting for disturbances and constraints. Finally, the controller works to ensure the spacecraft follows the planned trajectory, adjusting the control commands based on discrepancies between the expected and actual states, as well as any disturbances. This modular architecture allows for flexibility in implementing various estimation and control strategies.

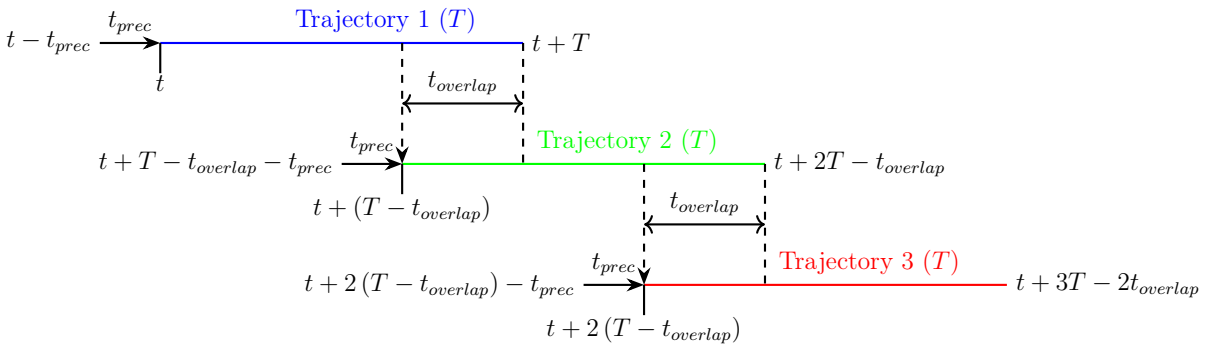


Figure 7.4: Diagram showing timing of trajectories, including trajectory duration, overlap between trajectories, and precalculation time.

state (from the estimator) and adjusts the control output given to the actuators. This can vary from an open-loop control that simply recreates the commands from the planner, to an LQR-based approach where the controls are adjusted based on a linear function of the state error, to a more advanced predictive controller that barely considers the "planned" control. There are further extensions to all of these, based on whether they account for actuator biases and how/whether they include and counter disturbances (that may or may not have been included in the planner and that may certainly differ in value from those in the planner). These options are described further in Section 7.4.

7.3.2 Real-world Implementation

There are specific questions that must be addressed for a real-world, real-time planning system such as this. Computational complexity is a significant concern, but additional challenges arise because the planner requires time to execute. These include generating future orbits and relevant environmental variables (such as the magnetic field), the potential for non-convergence, and the reliance on estimated data.

In an ideal scenario, the planner would continuously update the trajectory at every time step, integrating the latest information. However, this is infeasible due to computational demands and the time required for calculations. Practically, for a trajectory from time t to $t + T$, the planner must begin calculations earlier, t_{prec} ahead of schedule, at $t - t_{\text{prec}}$. As shown in Figure 7.4, the planner is triggered at time $t - t_{\text{prec}}$ to produce the trajectory that runs from t to $t + T$. There is also the possibility that the planner will not converge (or find some odd result), so it is important that t_{prec} is more than double the expected calculation time, allowing for a second attempt, if needed. A number of simple heuristics of cost, control effort, or sampling points along the trajectory can be used to determine when this is needed.

The system must also account for scenarios where the planner produces trajectories that are unusable or delays beyond the allocated margin. In these cases, or when the satellite encounters adverse conditions (e.g., high angular velocity), the ADCS architecture must provide fallback mechanisms in the ADCS architecture, allowing the system to switch to a different control paradigm. There may be several options based on what is going wrong and how severe it is, but the goal is to have a safe mode (or tree of possible safe modes) for the

control algorithm. A hierarchy of safe modes may include switching to simplified planning, PD-based control, or even detumbling modes like B-dot control.

Another mitigation strategy for computational delays is overlapping trajectories. As shown in Figure 7.4, one trajectory may extend beyond its nominal endpoint, covering t to $t+T+t_{\text{overlap}}$, while the subsequent trajectory spans $t+T$ to $t+2*T+t_{\text{overlap}}$. This overlap provides a buffer if the planner requires extra time or experiences errors, as the satellite can rely on the extended portion of the previous trajectory until t_{overlap} has elapsed. A significant overlap, often one-third to one-half of the trajectory duration, mitigates disruptions that typically occur near trajectory endpoints.

Trajectory endpoints may exhibit distinct dynamics compared to the rest of the trajectory, stemming from the planner not "looking forward" as much as it nears the end of the trajectory window. For example, if the desired angular velocity is set to zero, it may maintain an angular velocity throughout most of the trajectory as needed to continue to meet pointing goals, but slow the velocity near the end as there are no more future pointing goals to be met in its timespan. If using a TVLQR post-plan control approach, the gains may also get smaller near the end, as the future cost decreases with the remaining time horizon. Transitioning to a new trajectory midway is a straightforward solution to address these endpoint effects while creating a buffer for computational delays.

Larger overlaps (e.g., 75%, 90%, or more) can enhance responsiveness, but they also increase computational costs and introduce risks like trajectory chatter. Chatter occurs when multiple near-equivalent solutions exist–such as a high-accuracy trajectory with a high control cost versus a less accurate but more efficient alternative, or a constraint-based trade such as going above versus below a keep-out zone–and the planner oscillates between these options based on small initialization differences or minor changes in costs. The result can be a composite trajectory that appears indecisive, switching between solutions. With shorter overlaps, each trajectory segment is only recalculated a few times (typically 1-2), minimizing the risk of oscillations. However, chatter may still occur within a single planner run, such as when the algorithm switches between coarse and fine timesteps or during ALTRO convergence. Although these effects have generally not compromised trajectory feasibility in tests, they do occasionally increase computation time and should be monitored closely, particularly if larger overlaps are considered.

The necessity of precalculation also creates the issue of what initial state to give the planner. The planner needs an initial state from which to base its trajectory. The simplest approach is to just use the current state–which may work if the precalculation time is short or the system has high inertia (or minimal disturbances) and low angular velocity. If the precalculation time is long (as it may need to be, depending on the length of the trajectory, the complexity of the goals and constraints, and the computational speed), the estimated state at the time of initiation cannot be used, as the system will differ greatly by that point. Another approach is to propagate a control law forward, making assumptions about disturbances and errors, and use that as the starting point. It is unlikely to be exact, but it may be close enough for the controller to later correct for it. The use of trajectory overlap presents an additional opportunity–if the system is assumed to follow the trajectory closely, the appropriate point on the current trajectory can be used as the starting point on the next trajectory which overlaps with the current. This works quite well and is the approach generally used in this work. It does place some constraints on values of T, t_{prec}, and t_{overlap},

as the time of calling the next trajectory at $t + T - t_{\text{prec}}$ cannot fall before the start of the current trajectory window at time t (this prevents us from using a very large value of t_{prec}). Once a trajectory is planned, the last real-world detail is how to act on it in the larger ADCS. The question of following a planned trajectory is addressed in Section 7.4.

Connected to these real-world timing issues is the question of how frequently to call the planner–how long should the trajectories be? In Section 7.5, a number of examples are shown, with trajectory durations ranging from 450s (including 150s of overlap) to 3600s. ALTRO and the rest of the planner architecture can handle longer and shorter trajectories. The length of the trajectory time step is a key question here, as a two-day (48h) trajectory with a minute-long timestep is actually less computationally intensive for ALTRO than an hour-long trajectory with a second-long timestep. However, propagating the orbit ahead and calculating relevant goals, disturbances, and environmental information would be more complicated for longer durations. There is also the risk of orbit propagation diverging slightly (but increasingly over time) from the actual orbit. When orbital details (like the magnetic field for magnetorquers or a nadir-directed pointing goal) are part of the mission, this can cause issues–an example can be seen in Section 7.5.4. Generally, this work was based on long-term goal-meeting trajectories that last at most a couple hours and real-time path-planning trajectories on the scale of 2-20 minutes. Further study into the most efficient choices of duration and timestep is warranted, perhaps including some mix of coarse and fine timesteps (already used here) or coarse long-duration mission planning combined with finer, repeated short-term trajectories.

7.3.3 Convergence Questions

Convergence is a critical concept in trajectory planning, representing the ability of the algorithm to arrive at a solution that meets constraints, minimizes cost, and aligns with trajectory objectives. Evaluating convergence once a planner has completed its output is critical for ensuring the solution is viable but as-yet unimplemented. This evaluation can involve analyzing the final trajectory's smoothness, adherence to constraints, and the magnitude of penalty terms. Comparing these metrics across test cases or scenarios provides valuable feedback for refining the planner and understanding its performance and can be used by the ADCS to determine if a trajectory should be attempted or whether it should be re-run.

There are also opportunities to assess convergence mid-optimization, potentially allowing for more rapid analysis and re-attempting when needed. Intuition from live planner runs suggests opportunities for developing guidelines to indicate when convergence is unlikely (or slow and inefficient). Steady decreases in the cost function and smooth trajectory transitions often suggest progress. Conversely, persistent oscillations, unresolved high penalty costs, sudden trajectory jumps, or "bumps" in an otherwise smooth trajectory can signal poor convergence. Translating these observations into real-time flags or scoring systems could improve responsiveness. For instance, flags might track penalty dominance or sudden shifts in angular velocity to diagnose issues dynamically. While advanced methods like machine learning and other statistical methods could be used here to identify complex patterns, heuristic-based approaches might offer significant utility with minimal overhead.

Criteria for stopping the planner could include thresholds for cost improvement, iteration

limits, or acceptable residual constraint violations. For example, minimal cost function improvement over successive iterations might indicate convergence. ALTRO already incorporates many such criteria, including iteration limits and high penalty detection. However, additional tests, such as monitoring trajectory behavior or constraint dynamics, could further refine decision-making.

A key distinction exists between cases where the planner struggles to converge and those where no feasible solution exists. Poor initial trajectory guesses or restrictive constraints might prevent convergence, suggesting potential adjustments like relaxing constraints or refining parameters. In contrast, infeasible problems—marked by persistent high penalties or dynamics which cannot be controlled—require different strategies.

Future work could include creating a dataset of planner runs labeled by success or failure to refine diagnostics. These tools could help identify patterns in cost trends or trajectory behaviors, improving both reliability and adaptability. Incorporating such insights into the planner framework would enable dynamic adjustments and fallback strategies, enhancing its overall effectiveness.

7.4 Following a Planned Trajectory

A planned trajectory may take into account many details of the satellite's environment, but there are questions of how to then follow it. In a real-world system, with uncertainty, error, and noise, trying to follow the trajectory blindly will cause most real-world systems that are not self-stabilizing to diverge exponentially. There are many different solutions, including using any basic control law (like PID or sliding surface) with the trajectory state as the goal, but two general, trajectory-based ideas are highlighted in this section: a linear LQR law based on the trajectory, and a short-time-horizon model predictive control (MPC) approach. Both of these approaches can be further augmented by modifications that account for actuator biases or additional disturbances.

7.4.1 Time-Varying Linear Quadratic Regulator

One robust approach is to use the trajectory information itself to make a simple control law. The linear quadratic regulator (LQR) problem has a long history in control theory and provides an optimal solution for simple systems in minimizing a cost function over time for an evolving, controlled system. By linearizing more complex systems, it can be used to derive a control law (no longer provably optimal) for them. This is actually very similar to the mid-level loop in ALTRO, as part of the iLQR loop, as described in Figure 7.1. It is simple to have the planner output gains that can be used to guide the system over time, called a time-varying linear quadratic regulator (TVLQR). Specifically, based on the system cost and forward trajectory, the planner can output for each timestep a matrix K_k such that

$$\mathbf{u}_{k,\text{control}} = \mathbf{u}_{k,\text{plan}} + K_k \left(\mathbf{x}_{k,\text{estimate}} - \mathbf{x}_{k,\text{plan}} \right) \tag{7.39}$$

generates a control that attempts[4] to minimize the cost of the remaining trajectory. Note that the subtraction of states in 7.39 (and in other equations throughout this section) represents not just arithmetic subtraction, but the correct method of taking differences. For example, in the case of the quaternion, subtraction will not represent the actual difference between one quaternion and another. Instead, the quaternion difference should be used and reduced to a three-dimensional representation. Other data types may need to have similar adjustments.

A more advanced version of the TVLQR involves including the disturbance environment in the modification, Specifically, generating a K_k matrix that includes the effects of the disturbances to result in the LQR control law

$$\mathbf{u}_{k,\text{control}} = \mathbf{u}_{k,\text{plan}} + K_k \left(\begin{bmatrix} \mathbf{x}_{k,\text{plan}} \\ \boldsymbol{\tau}_{k,\text{expected}} \\ \mathbf{m}_{k,\text{expected}} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_{k,\text{estimate}} \\ \boldsymbol{\tau}_{k,\text{estimate}} \\ \mathbf{m}_{k,\text{estimate}} \end{bmatrix} \right) \tag{7.40}$$

that accounts for differences in the disturbance torques $\boldsymbol{\tau}$ and the spacecraft residual dipole $\mathbf{m}$. This requires modifying the trajectory planner (or the TVLQR method within it, at least) to account for disturbances when performing the LQR calculation. This modification can be valuable when there is an added disturbance that is hard to predict. It also requires outputting and storing the expected disturbance for each time point and then estimating the actual disturbance with sufficient accuracy.

7.4.2 Model Predictive Control

A more advanced approach to following a planned trajectory leverages the system's dynamics model (contained, in this instance, in the unified Satellite class framework, described in Section 4.3) to calculate control inputs that guide the system toward the planned trajectory while respecting constraints. This method, known as Model Predictive Control (MPC), typically solves an optimization problem covering a trajectory for a finite time horizon ahead, recalculating the current-time controls based on this. It is discussed in Section 2.2.2, but is not frequently used as it has a high computational cost because MPC performs a trajectory optimization at each time step. However, the approach implemented here simplifies the computational burden significantly by solving only a single-step optimization problem at each iteration. This makes it computationally tractable for real-time applications while retaining many of the advantages of traditional MPC.

By optimizing control inputs one step ahead, the method still utilizes the full nonlinear dynamics model, enabling better incorporation of actuator biases, disturbances, and system nonlinearities. Because it aims to follow a precomputed trajectory generated by a planner, many of the benefits of traditional multi-step MPC—such as trajectory compliance and dynamic adaptability—are preserved. The precomputed trajectory ensures that the control inputs are directed toward meeting long-term goals, while the single-step approach recalculates controls dynamically to handle deviations caused by unmodeled effects or disturbances. This can include disturbances which were not modeled in the original planned trajectory and those that were modeled with a different value (such as a disturbance evolving over time).

[4]For a linear system and quadratic cost, this will fully minimize, but the system in this work does not meet these properties and thus it is just an approximation.

A slimmed-down version of MPC is used in much of this work, to have a short computation time while dealing well with the dynamics nonlinearities, control bounds, and disturbances. A basic solution would be to solve (either using an iterated line search and log-barrier method[5] or readily available optimization libraries) the optimization problem

$$\begin{aligned} \underset{\mathbf{u}_k \in \mathscr{U}}{\text{minimize}} \quad & \frac{1}{2} \left\| \mathbf{x}_{k+1,\text{estimate}} - \mathbf{x}_{k+1,\text{plan}} \right\|_{Q_s}^2 + \frac{1}{2} \left\| \mathbf{u}_k \right\|_{Q_u}^2 \\ \text{subject to} \quad & \mathbf{x}_{k+1,\text{estimate}} = \text{RK4}\left(\mathbf{x}_{k,\text{estimate}}, \mathbf{u}_k, t_k \right), \end{aligned} \tag{7.41}$$

where $\mathbf{x}_{k+1,\text{estimate}}$ is the predicted state using the RK4 integrator, $\text{RK4}\left(\mathbf{x}_{k,\text{estimate}}, \mathbf{u}_k, t_k \right)$ is the propagation of system dynamics forward one time step (via RK4 numerical integration) from time t_k (including disturbances[6] and biases), $\mathbf{x}_{k+1,\text{plan}}$ is the planned trajectory state, $\mathscr{U}$ is the feasible set of control inputs constrained by actuator limits, and Q_s and Q_u are weighting matrices for trajectory error and control effort, respectively.

The approach adopted in this work differs slightly from the formulation in Equation 7.41. Specifically, the MPC method used takes advantage of the trajectory's calculation (and potentially can benefit from the planner's understanding of the cost of the rest of the trajectory) by weighting the future state difference by the TVLQR gains from the *next* time step. This also removes the question of how to choose Q_s and Q_u and incorporates the trajectory-following effects of the LQR gains into the trajectory-following efforts. Specifically, the optimization problem that is solved is

$$\underset{\mathbf{u}_k \in \mathscr{U}}{\text{minimize}} \quad \frac{1}{2} \left| K_k \left(\text{RK4}\left(\mathbf{x}_{k,\text{estimate}}, \mathbf{u}_k, t_k \right) - \mathbf{x}_{k+1,\text{plan}} \right) \right|^2 \tag{7.42}$$

A more elegant approach may be to use the cost-to-go difference matrix S_k as the weight instead of $K_k^T K_k$, which is calculated in the TVLQR as part of generating the set of K_k.

Although the optimization problem in Equation 7.42 is not provably convex (even if $\mathscr{U}$ is) due to the nonlinear rotation dynamics, small time steps and low-speed trajectories may allow the use of a convex-optimization approach or other simple optimizers. This combination of flexibility, adaptability, and computational efficiency makes MPC particularly suitable for systems with nonlinear dynamics, actuator limits, and varying disturbance profiles.

This formulation makes use of the system's full dynamics model, providing several advantages over simpler feedback-based controllers. It enables direct handling of nonlinear dynamics, actuator bounds, and disturbances. For example, time-varying disturbances or actuator biases can be included in the optimization problem, allowing the controller to counteract these effects dynamically without requiring a reformulation of the system model or re-planning of the trajectory. Unlike the TVLQR, this MPC approach adjusts dynamically to disturbances encountered during execution. While trajectory planning anticipates disturbances based on a priori knowledge, real-world deviations often arise. By recalculating the controls at each step, instead of just a linear adjustment, MPC naturally compensates for disturbance deviations, reducing reliance on conservative design margins or slower integral correction mechanisms.

[5] This seems feasible to the author. For short time steps and small stored angular momentum, the problem seems close to convex.

[6] Which disturbances to include is a design decision–an "all-in-one" disturbance as described in Section 6.3 could be applied–though care should be taken if the trajectory involves much rotation, as many external torques vary with orientation.

7.5 Tests and Results

The section focuses on validating the proposed trajectory planning framework through a series of simulation tests designed to evaluate its performance across various scenarios. These tests aim to demonstrate the robustness, adaptability, and accuracy of the trajectory planner in handling different mission constraints, including long-term planning, disturbance rejection, and multi-target transitions. By examining the behavior of both underactuated and fully-actuated satellites, this section highlights the practical implications of using an autonomous planner to improve mission success rates and operational efficiency in real-world satellite missions.

7.5.1 Long-Term Trajectory Planning for Underactuated Satellite with Multiple Goals

The ALTRO-based Planner is applied to the problem of a 3U CubeSat with only magnetic control, trying to point at a target on the ground, followed by pointing a thruster anti-ram. The underactuated nature of magnetic control, including a model of the orbital magnetic field, is taken into account, but disturbances are not included. Test parameters can be found in Table 7.2.

The details of this trajectory can be seen in Figure 7.5. The satellite, with very little actuation effort, steers itself to track a target on the ground (though limited by its maximum angular velocity constraint–which is why there are two small humps of pointing error at the highest speed), then turns to point for a different goal (a propulsion system) in a different direction. All of this is controlled by the nonlinear (and underactuated) magnetorquers. Note how the satellite turns about the pointing axes, as rotation there is not penalized.[7] The details of the trajectory can be found in 7.6.

The results presented in Figures 7.5 and 7.6 demonstrate the trajectory planner's ability to address the challenges of a MTQ-only satellite tasked with achieving multiple, rapidly changing goals over the span of a single hour. Despite the inherent underactuation of MTQs, which can only generate torque perpendicular to the Earth's local magnetic field, the planner generates smooth and feasible trajectories that respect strict operational constraints.

Figure 7.5 highlights the evolution of the satellite's attitude over time. Starting from an arbitrary orientation, the satellite transitions smoothly between distinct pointing goals, first tracking an Earth-fixed target and then orienting its propulsion system anti-ram. The trajectory avoids unnecessary rotations and remains efficient, even under the limitations imposed by the MTQs. It incorporates magnetic field knowledge from the IGRF to plan its underactuated trajectory. The ability to preemptively account for underactuation and constraints while still achieving accurate pointing performance represents a significant advancement over traditional methods such as simple feedback controllers.

In Figure 7.6, the angular velocity, control effort, and angular error are plotted, providing further insight into the planner's performance. The angular velocity plot clearly shows the planner respecting the 1 deg/s angular velocity constraint it was given. However, as a result

[7] An animated version of this can be found at https://github.com/patrickmckeen/PhD_Dissertation_Code/blob/main/beavercube-adcs-master_thesis/traj_anim5.gif

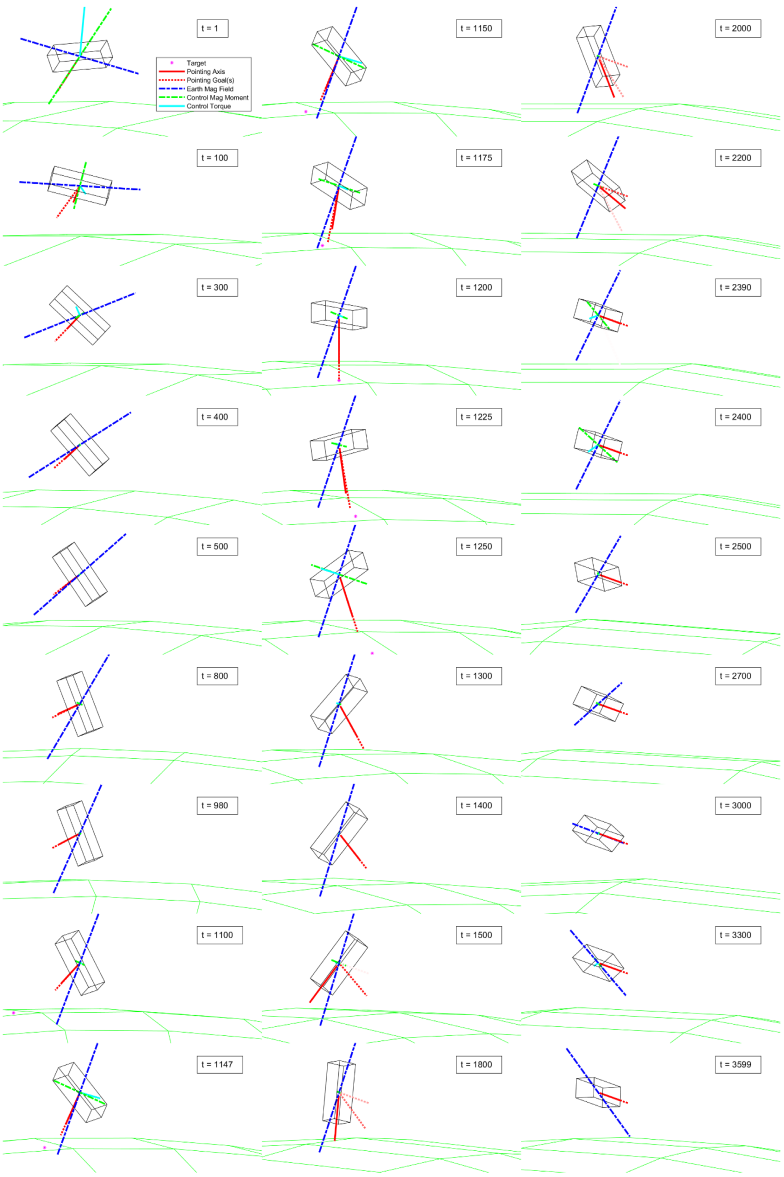


Figure 7.5: Graphic showing various time points of a trajectory generated for a CubeSat to point at an Earth-fixed target then point its propulsion system anti-ram, with details in Table 7.2. At $t = 1$, the CubeSat is floating freely in an arbitrary orientation. It plans a trajectory for the next hour (3600 seconds) to meet two goals: tracking a target on the ground (small star) within an 8-minute window centered on 1200 seconds (960s-1440s), and then pointing its propulsion unit anti-ram from 2400 seconds onwards.

Table 7.2: Satellite Properties and Test Parameters for Underactuated CubeSat Two-Goal Test

Property	**Value/Description**
Inertia Matrix[a]	$J = \begin{bmatrix} 0.0314 & -0.0067 & 5.9 \times 10^{-5} \\ -0.0067 & 0.01005 & -0.00012 \\ 5.9 \times 10^{-5} & -0.00012 & 0.0341 \end{bmatrix}$ kg·m^2;
MTQ[a]	Three-axis, identical on each Maximum Magnetic Moment: 0.15 Am^2 No bias.
Disturbances	None
Orbit Parameters	ISS orbit
Pointing Goals	960s-1440s[b] point camera ($-\hat{\mathbf{z}}$) at specific ground target.[c] After 2400s: point propulsion ($\hat{\mathbf{x}}$) anti-ram
Initial State	Angular Velocity: $[0, 0, 0]$ deg/s Quaternion: $[-0.232, -0.664, -0.234, -0.671]$ Stored Momentum: 0 Nms
Planner Time Step	1s
Trajectory Duration	3600s
Planning Constraints	Respect MTQ control bounds. $\lvert\vec{\boldsymbol{\omega}}\rvert <= 0.8°/\text{s}$ $\lvert\omega_i\rvert <= 0.8°/\text{s}$ for $i = x, y, z$.

[a] Inspired by BeaverCube, the original motivation for much of this work (Section 2.4).

[b] This is an 8-minute-duration window centered on 1200s.

[c] Ground target moves with planet beneath satellite orbit, but is selected to be directly nadir at $t = 1200$s, with an ECEF radius of 6370 km.

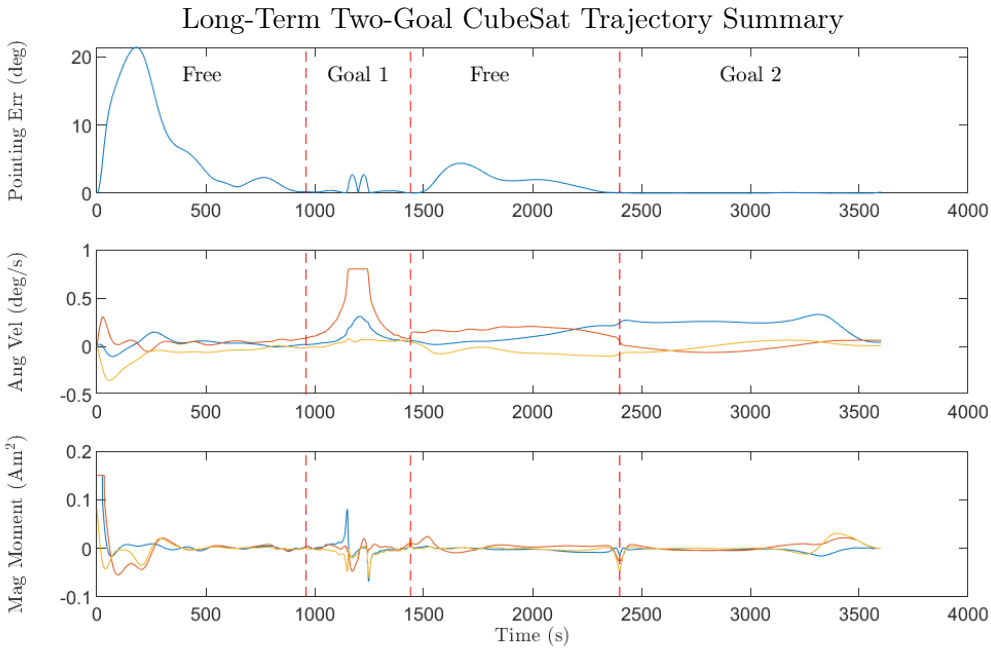


Figure 7.6: Plots showing the generated trajectory (angular velocity, control effort, and angular error from goal) of a trajectory generated for a CubeSat to point at an Earth-fixed target, then point its propulsion system anti-ram, with details in Table 7.2.

of this limitation, small angular error humps arise during transitions between goals. These humps are not due to insufficient control authority but are instead a deliberate consequence of strictly adhering to the given trajectory constraints. The presence of these humps highlights an important trade-off: respecting angular velocity limits means the satellite cannot slew as quickly as it might otherwise, leading to small temporary deviations. However, the planner balances these constraints with the need to meet pointing goals. The control effort plot confirms the efficient use of the magnetorquers, with actuation levels well within their operational limits.

These results underscore the importance of planning ahead in underactuated systems. By leveraging trajectory optimization, the planner mitigates the effects of underactuation and disturbances, enabling the satellite to achieve performance levels that would traditionally be considered infeasible for an MTQ-only system. The combination of constraint-respecting behavior and accuracy validates the planner's utility for real-world applications, such as autonomous CubeSat missions requiring rapid retargeting, efficient goal sequencing, and adherence to strict safety constraints.

7.5.2 Planning around a Disturbance

The ALTRO-based Planner is applied to the problem of a 3U CubeSat with magnetic control and one reaction wheel (around the y-axis), trying to point a z-axis thruster (which has a disturbance torque of 0.3 mN around the x-axis) anti-ram. The maximum moment of the magnetorquers is 0.19 Am^2 around the x-axis and 0.57 Am^2 around the y- and z-axes. The underactuated nature of magnetic control, including a model of the orbital magnetic field, is taken into account. The planner is given propulsion disturbance and accounts for it in the trajectory. The disturbance is not easily corrected for, however–the reaction wheel is perpendicular to the disturbance and the disturbance magnitude is too great for the magnetorquers to counter. Further details can be found in Table 7.3.

Table 7.3: Satellite Properties and Test Parameters for Overwhelming Disturbance Test

Property	**Value/Description**
Inertia Matrix[a]	$J = \begin{bmatrix} 0.1 & 0 & 0.00013 \\ 0 & 0.05 & -0.00021 \\ 0.00013 & -0.00021 & 0.005 \end{bmatrix}$ kg·m^2
MTQ	Three-axis Maximum Magnetic Moment: 0.19 Am^2 on x-axis, 0.57 Am^2 on y- and z-axes No bias No noise
RW[b]	Only one RW, aligned with y-axis Maximum Torque: 0.2 mNm Maximum Momentum: 2 mNms Wheel Inertia (about axis): 2×10^{-6} kg·m^2 Assumed perfect balance about axis
Propulsion Disturbance	Constant: Value: $[0.3, 0, 0]$ mNm Applied throughout.
Other Disturbances	None
Planner Disturbances	Only propulsion.
Orbit Parameters	Near-circular orbit Orbital Radius: 6800 km Inclination: 51.5° J2 effects included
Pointing Goals	Point propulsion ($\hat{\mathbf{z}}$) anti-ram
Initial State	Angular Velocity: $[0, 0, 0]$ deg/s Quaternion: $[-0.232, -0.664, -0.234, -0.671]$ Stored Momentum: 0 Nms
Planner Time Step	1s, 10s coarse time step first.
Trajectory Duration	500s

[a] Inspired by BeaverCube II, a successor to BeaverCube, the original motivation for much of this work (Section 2.4).

[b] Inspired by CubeSpace CubeWheel (small). [41]

The results of this test highlight the planner's ability to exploit the spacecraft's dynamics and available actuation in an underactuated and disturbance-affected system. Faced with a disturbance torque of 0.3 mNm around the x-axis, which the reaction wheel cannot directly oppose due to its alignment with the y-axis, the planner integrates the system's constraints into its optimization process. The magnetorquers, while capable of producing torque, are limited to 0.19 Am^2 around the x-axis, far below what is required to fully counter the disturbance. Instead of relying on instantaneous disturbance rejection, the planner engineers a dynamic equilibrium by introducing a deliberate spin about the z-axis (aligned with the nominal propulsion direction).

This spinning strategy effectively distributes the disturbance torque over time, "smearing" the disturbance around a cone. As a result, the net effect of the disturbance is averaged out, enabling the spacecraft to maintain stability and achieve precise pointing despite its physical limitations. The angular velocity plot in Figure 7.8 illustrates this behavior, showing the steady spin induced along the propulsion axis, while the attitude error plot (Figure 7.7) demonstrates the system's ability to maintain a tight pointing angle. Notably, the planner's solution adheres to actuator limits, as shown in the commanded torque plot (Figure 7.9), with the reaction wheel and magnetorquers operating well within their respective bounds.

This result underscores two critical aspects of the planner's approach. First, by explicitly including the disturbance in its optimization, the planner can devise solutions that would be unattainable with conventional, purely reactive control methods. Second, by leveraging the reduced control goals–specifically the reduced attitude goal of aligning two vectors (see Section 5.3) rather than enforcing a fully-specified orientation–the planner gains additional flexibility to adapt the spacecraft's dynamics to the disturbance environment. This combination of onboard disturbance estimation and incorporation, trajectory optimization, and reduced goal constraints allows the system to achieve stable pointing with limited actuation and under significant external torque.

These findings are particularly relevant for small satellites and systems with limited actuator authority, demonstrating that even in challenging, underactuated scenarios, high-precision pointing can be achieved. The planner's ability to generate feasible trajectories under such conditions highlights its robustness and adaptability, paving the way for missions requiring precise attitude control in the presence of complex disturbances.

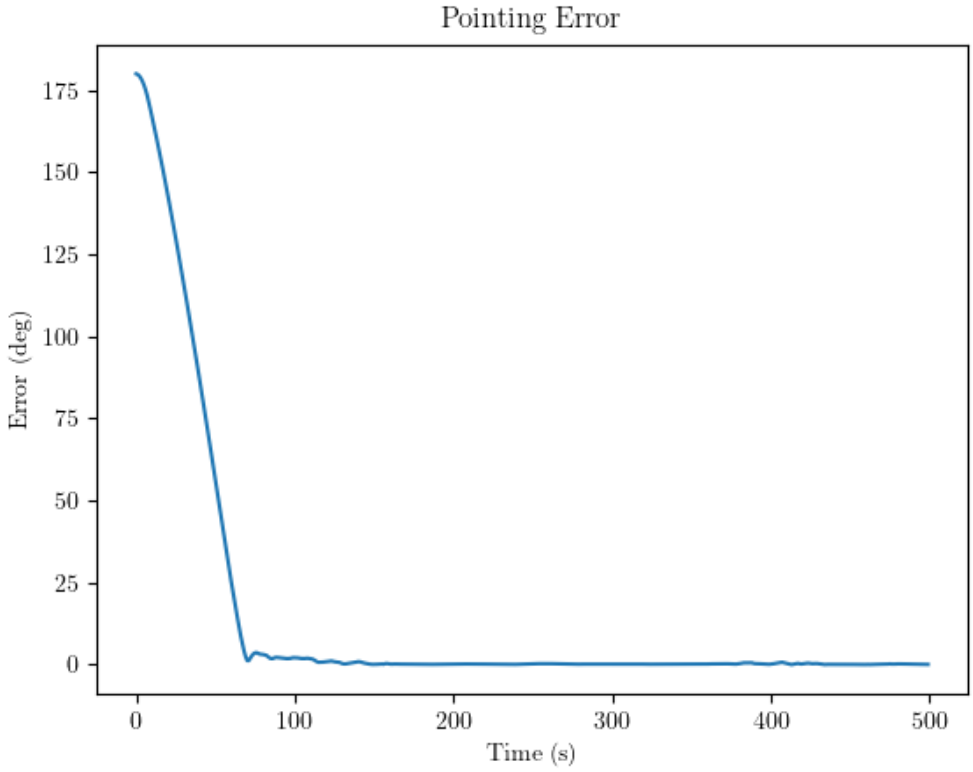


Figure 7.7: Plot of the angular pointing error experienced by a CubeSat countering a continuous, body-frame disturbance torque by inducing a controlled spin. Case details can be found in Table 7.3. The spinning strategy effectively distributes the disturbance over time, reducing its cumulative impact. The planner utilizes reduced attitude control goals to achieve high-precision pointing while staying within actuator limits. The data demonstrate the CubeSat's ability to maintain single-digit and sub-degree pointing accuracy despite significant external torque and limited actuation authority. The constant body-frame torque is too strong for actuators to counter directly and, without the spin, the system would build up too much momentum.

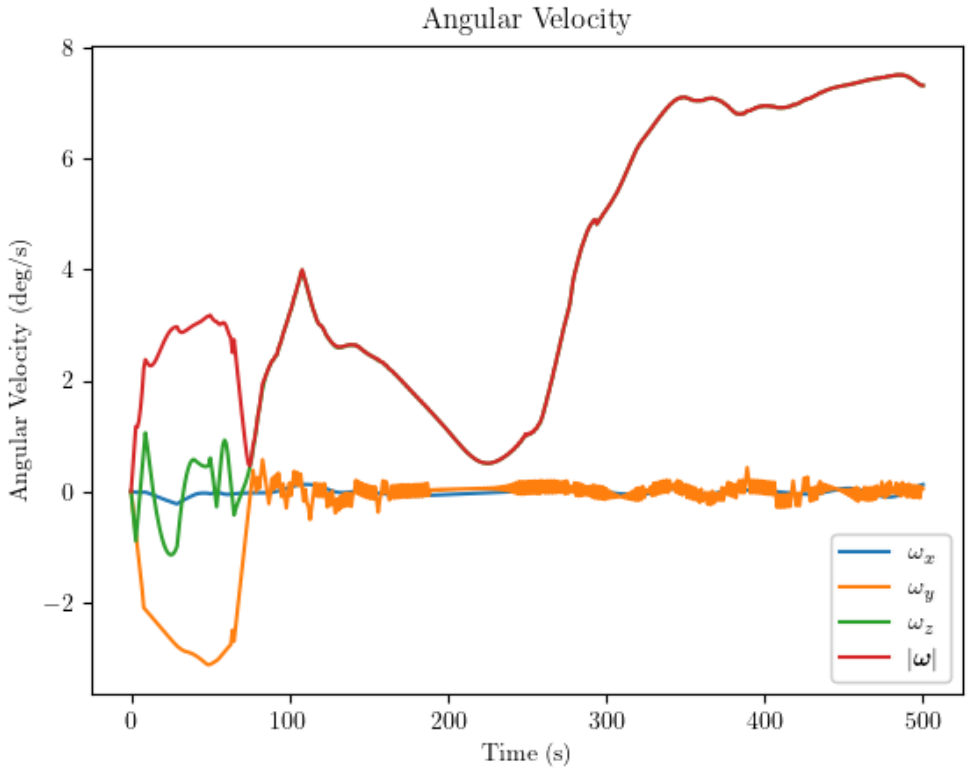


Figure 7.8: Plot illustrating the CubeSat's angular velocity profile as it counters a constant disturbance torque by inducing a controlled spin. Case details can be found in Table 7.3. The spinning strategy redistributes the disturbance effect over time, preventing momentum buildup. The trajectory planner effectively adjusts the angular velocity to ensure consistent pointing performance throughout the disturbance period. The constant body-frame torque is too strong for actuators to counter directly and, without the spin, the system would build up too much momentum.

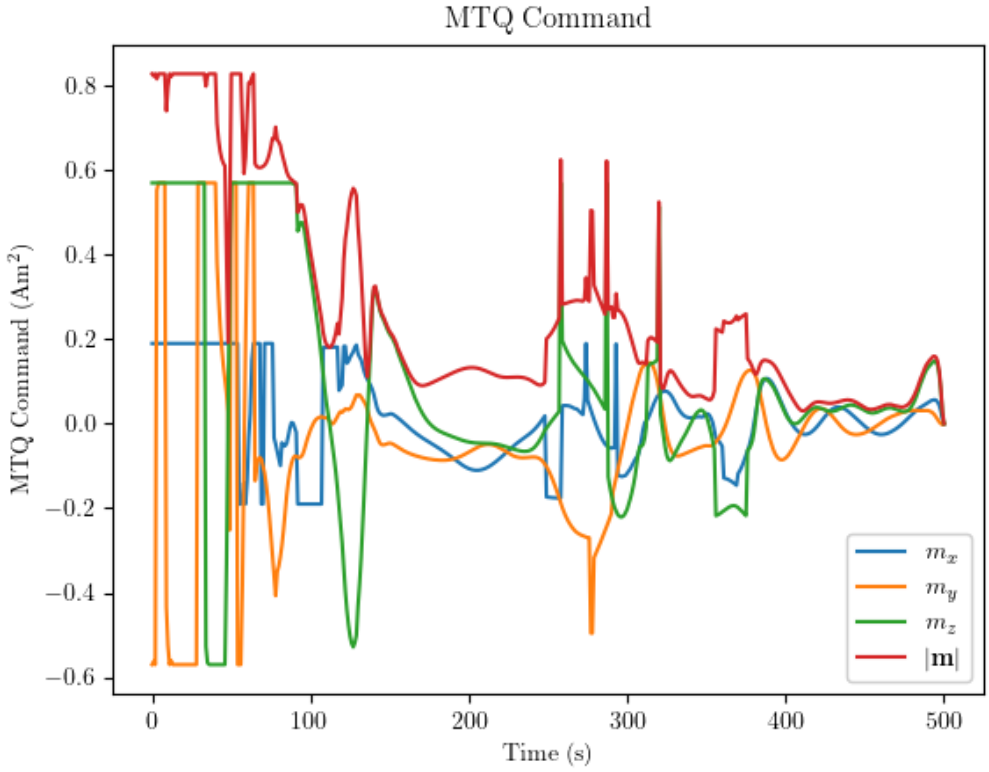


Figure 7.9: Graphic showing the control commands executed by the CubeSat during a trajectory in which it spins to maintain pointing accuracy under constant disturbance torque. Case details can be found in Table 7.3. Although the control inputs appear irregular, they are optimized to counter the changing effect of the magnetic field and maintain precise orientation. The trajectory planner adapts the control effort, while rotating through a changing external B-field, to balance torque counteraction and pointing goals. The constant body-frame torque is too strong for actuators to counter directly and, without the spin, the system would build up too much momentum.

Table 7.4: Satellite properties (reaction wheel properties below the dashed line).

Property	Value
Satellite J_{xx} (kg m^2)	0.005256
Satellite $J_{yy} = J_{zz}$ (kg m^2)	0.04939
Maximum magnetic moment (x) (Am^2)	0.19
Maximum reaction wheel torque (Nm)	0.0002
Momentum storage (Nms)	0.002
Reaction wheel moment of inertia (kg m^2)	2e-6
Reaction wheel rotation axis	$[0\ 1\ 0]^T$

7.5.3 Monte Carlo for Various Goals

A test is conducted to evaluate the usefulness of the ALTRO-based planner when applied to discrete goals across a range of starting conditions for satellites equipped with magnetorquers or a hybrid system of magnetorquers and a single reaction wheel. The properties of the simulated satellites, based on design parameters from the successor to the mission described in Section 2.4, are provided in Table 7.4.

As in Manchester & Gatherer [87], the trajectory planner's performance is evaluated by planning 180° slews from $q = [0\ 0\ 1\ 0]^T$ to $q = [0\ 1\ 0\ 0]^T$. Additionally, the planner's capabilities are tested in two other cases: a reduced-attitude goal slew that aims to align the satellite body x-axis with a direction in the Earth-centered inertial (ECI) coordinate frame, and a multiple-target trajectory that aims to sequentially align the satellite body x-axis with three different targets in the ECI frame.

For all three cases, the satellite's orbital position is initialized randomly along a circular ISS orbit (51.5° inclination) at an altitude of 429 km. The angular velocity and reaction wheel stored momentum are set to zero at initialization. In the cases of Reduced Attitude Goals and Multiple Targets, a 20-degree (10-degree half-angle cone) keep-out zone around the sun is applied, ensuring realistic operational constraints.

One of the architectures tested in these scenarios is the hybrid 3MTQ+1RW system, an uncommon and perhaps novel actuation framework. This configuration combines three magnetorquers, which generate torques aligned with the Earth's magnetic field, with a single reaction wheel capable of providing additional torque, including around the magnetic field line when properly aligned. Traditional control laws might struggle with this architecture due to the interplay of underactuation, non-aligned torque directions, different types of actuation, and an uneven distribution of control authority in the body frame.

The ALTRO-based trajectory planner, however, enables this architecture to perform reliably by leveraging its optimization framework to plan efficient and feasible trajectories. For instance, the planner can execute maneuvers that subtly align the reaction wheel with the magnetic field line, perform necessary adjustments using the reaction wheel, and then reorient the satellite while countering non-magnetic torques with the magnetorquers. This capability enables the 3MTQ+1RW system to achieve remarkable performance, far exceeding that of MTQ-only configurations, while requiring the addition of just a single reaction wheel. This improvement strikes an excellent balance between enhanced pointing precision and minimal increases in size, weight, and power, making it particularly advantageous for CubeSats and

Metric	MTQ-Only	3MTQ+1RW
% Within 1 Degree	11%	96%
% Within 10 Degrees	73%	100%
Mean Error (deg)	10.2	0.11
Median Error (deg)	4.2	1.2×10^{-4}
Std Dev Error (deg)	20.2	0.31

Table 7.5: Pointing Accuracy in Monte Carlo Simulations for the 180-Degree Slew Case. Test parameters can be found in Table 7.4.

small satellites.

The Monte Carlo cases presented in the following sections are designed to highlight the planner's adaptability and robustness in addressing diverse challenges. These include simple slews, reduced-attitude goals, and complex multi-target maneuvers, showcasing how trajectory planning can enable precise pointing and rapid retargeting for satellites equipped with unconventional actuation architectures.

7.5.3.a Simple 180° Slew

We perform a 100-trial Monte Carlo simulation with randomized initial orbital conditions to evaluate the final attitude error after planning and executing 180° slews for two satellite configurations: magnetorquer-only (MTQ-only) and a hybrid configuration with three magnetorquers and one reaction wheel (3MTQ+1RW). The final attitude error is defined as the angle between the achieved attitude quaternion and the target quaternion $[0\ 1\ 0\ 0]^T$. Key statistics of the results are summarized in Table 7.5.

The results for the MTQ-only configuration are shown in Figure 7.10. While the MTQ-only satellite exhibits larger variability in final attitude error, it achieves acceptable performance in many trials and performs admirably for MTQ-only control on a CubeSat. The mean final attitude error for MTQ-only slews is 10.2°, with a median error of 4.2°, and a standard deviation of 20.2°. Notably, 73% of the trials result in less than 10° error and 50% are within 4°. However, some trials converge to suboptimal trajectories, characterized by spinning and final errors exceeding 10°. These cases highlight the challenges of achieving precise pointing with magnetic control (due to its underactuation), particularly for maneuvers requiring significant reorientation. Importantly, the mean slew time for these trials is around 500 seconds, which is notably fast for an MTQ-only satellite.

For the hybrid 3MTQ+1RW configuration, the trajectory planner demonstrates significantly improved performance, as shown in Figure 7.11. The mean final attitude error is 0.11°, with a median error of 1.2×10^{-4} ° and a standard deviation of 0.31°. Importantly, 96% of trials achieve sub-degree accuracy, and all trials result in less than 10° error. This improved accuracy is expected due to the added torque capability provided by the reaction wheel, which allows for control around the magnetic field axis. The trajectory planner leverages this capability effectively, maintaining low variability and achieving precise pointing (and rapidly, too–most maneuvers converge within roughly 100 seconds). The planner is not overly dependent on the reaction wheel, however–it generally stores very little momentum in the

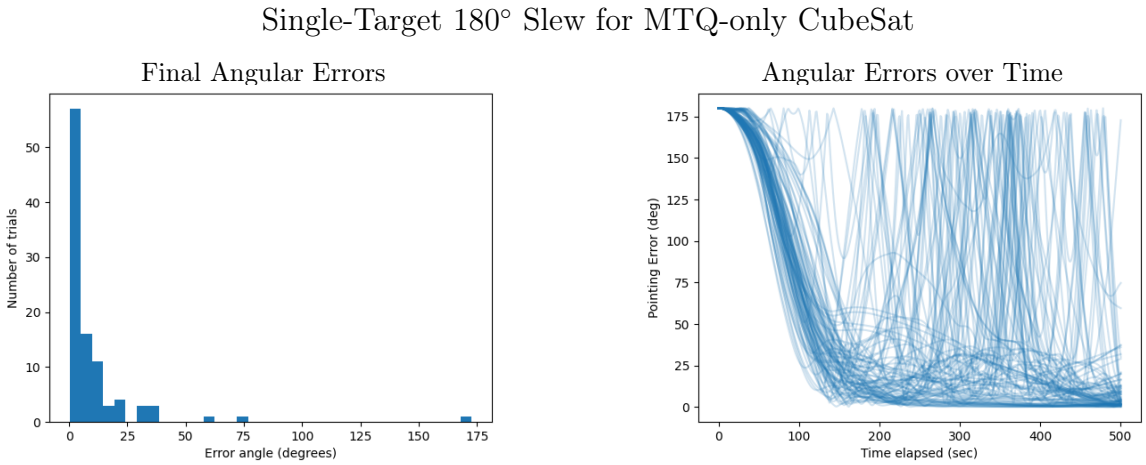


Figure 7.10: Results of 100-trial Monte Carlo simulation for planning 180° slews for a magnetorquer-only satellite. Left: Final attitude error for all trials. Right: Visualization of attitude error over time for all trials. Test parameters can be found in Table 7.4.

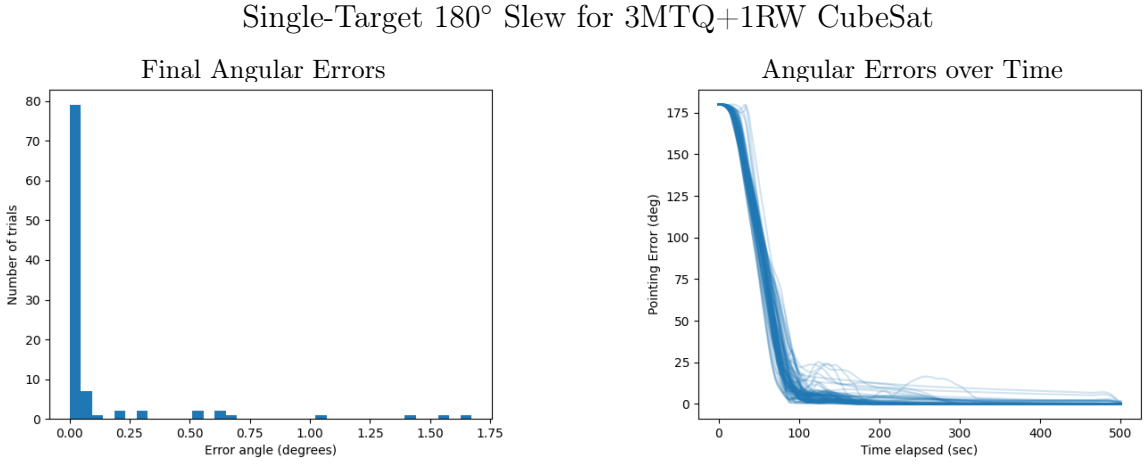


Figure 7.11: Results of 100-trial Monte Carlo simulation for planning 180° slews for a 1RW satellite. Left: Final attitude error for all trials. Right: Visualization of attitude error over time for all trials. Test parameters can be found in Table 7.4.

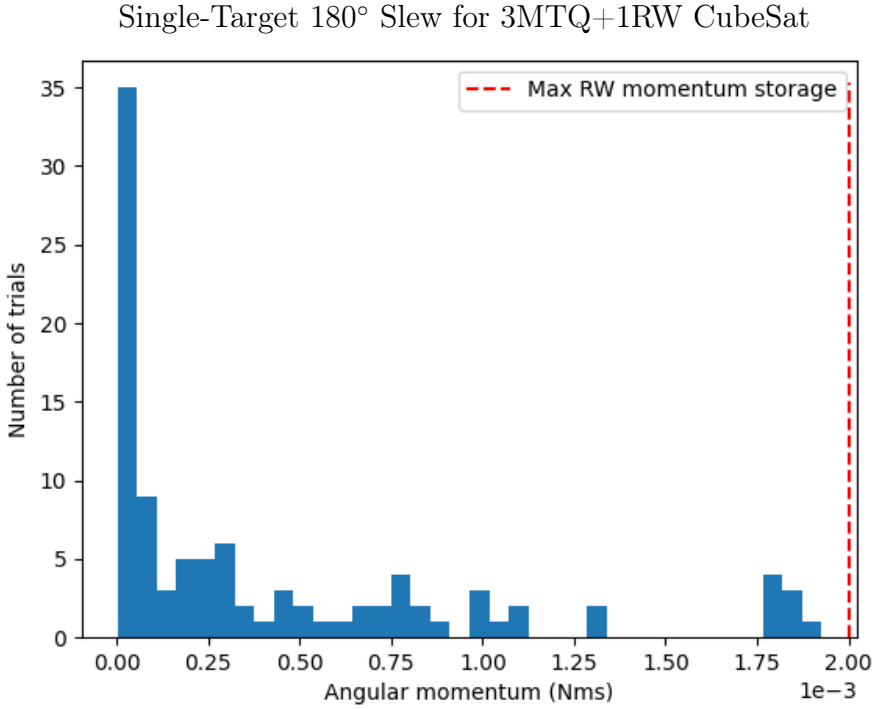


Figure 7.12: Results of 100-trial Monte Carlo simulation for planning 180° slews for a 1RW satellite: final RW stored angular momentum for all trials. Test parameters can be found in Table 7.4.

reaction wheel and prevents reaction wheel saturation across all trials (Figure 7.12).

Figures 7.13 and 7.14 provide insight into the planned quaternion trajectories for both the MTQ-only and 3MTQ+1RW configurations. In the MTQ-only case, poor convergence is characterized by persistent oscillations in the planned trajectory, as shown in the right panel of Figure 7.13. These oscillations indicate the inherent limitations of magnetic control in achieving precise pointing when the trajectory planner converges to suboptimal solutions and may be a direct impact of the limits of uneractuation. For the hybrid configuration (Figure 7.14), even in poor convergence cases, the trajectories oscillate closer to the target orientation and achieve pointing errors below 2°, which is well within the acceptable range for many CubeSat missions. This underscores the planner's ability to effectively balance actuation between the reaction wheel and magnetorquers, enabling robust control even under challenging conditions.

While the MTQ-only results fall short of the hybrid configuration, they are still impressive given the limitations of purely magnetic control and would likely be acceptable (and perhaps even an improvement) for many CubeSat missions. Achieving sub-degree pointing in 11% of trials and errors under 10° in 73% of trials demonstrates the trajectory planner's ability to handle underactuation effectively and to incorporate information of the magnetic field into the plan. These results make MTQ-only control feasible for missions with relaxed pointing requirements or sufficient time to stabilize after slews. The pointing accuracy surpasses typical expectations for CubeSat magnetic control, where errors of 10°–20° are common.

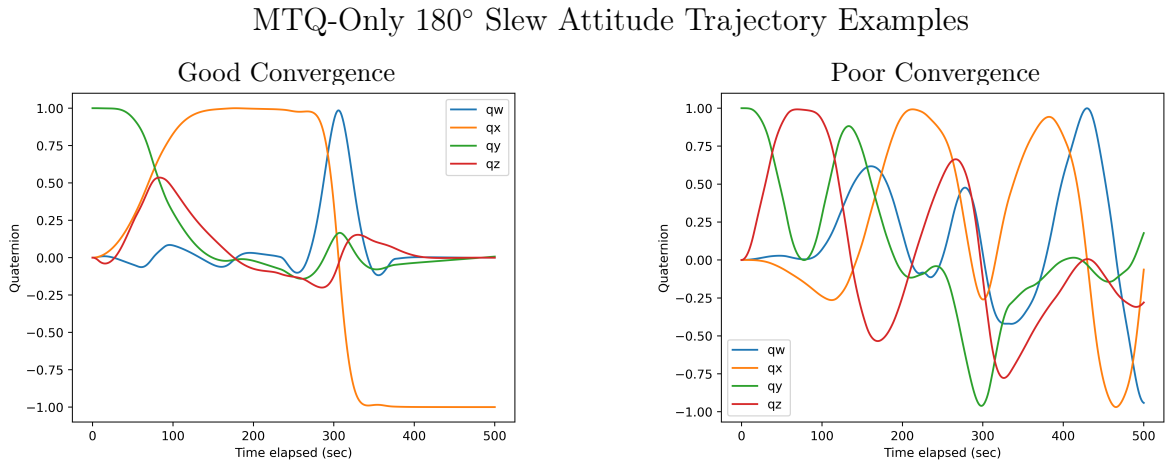


Figure 7.13: Two planned slews for a magnetorquer-only satellite. Left: "Good convergence" case, demonstrating a slew to target. Right: "Poor convergence" case, showing noisy oscillation in attitude and no convergence to the target. Test parameters can be found in Table 7.4.

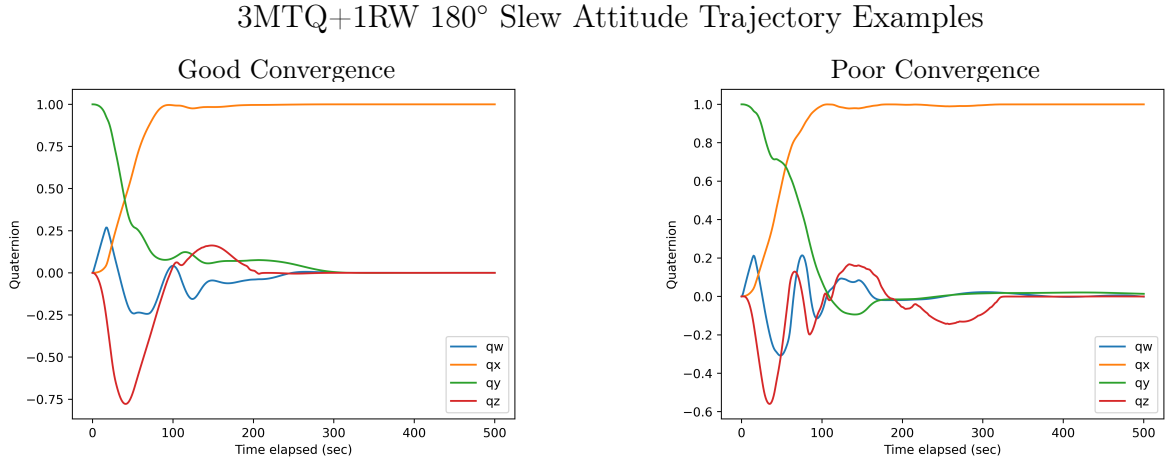


Figure 7.14: Two planned slews for a 1RW satellite. Left: "Good Convergence" case, demonstrating a direct slew to target. Right: "Poor Convergence" case, demonstrating a slow oscillation towards the target. Test parameters can be found in Table 7.4.

Metric	MTQ-Only	3MTQ+1RW
% Within 1 Degree	67%	96%
% Within 10 Degrees	75%	100%
Mean Error (deg)	16.2	0.16
Median Error (deg)	0.04	2.6×10^{-3}
Std Dev Error (deg)	35.0	0.55

Table 7.6: Pointing Accuracy in Monte Carlo Simulations for the Reduced Attitude Goal Case. Test parameters can be found in Table 7.4.

For the hybrid 3MTQ+1RW architecture, the results emphasize the planner's role in making this unconventional configuration highly effective. By coordinating the reaction wheel and magnetorquers, the planner enables precise control even under challenging initial conditions. For those missions with sub-degree pointing requirements, our extension of ALTRO is fast enough that if a slew is not time-critical, a 1RW satellite could replan and attempt to slew again if the first plan results in more than one degree of final attitude error. This makes the hybrid configuration a strong candidate for missions requiring high pointing accuracy while maintaining a balance between cost, complexity, and performance.

7.5.3.b Reduced Attitude Goal Slew

To evaluate the reduced-attitude goal slews, we performed a 100-trial Monte Carlo simulation with randomized initial orbital conditions. The goal is to align the satellite body x-axis with the vector $[\cos(10°), 0, \sin(10°)]^T$ in the Earth-centered inertial (ECI) frame. Unlike the 180° slews in Section 7.5.3.a, this reduced-attitude goal requires only alignment of one satellite body axis, leaving rotations around that axis unconstrained. (The equations describing the set of quaternions which meet this goal can be succinctly described as in [130] or as defined in Section 5.3.) This relaxation increases the set of feasible solutions and is particularly advantageous for underactuated systems like those relying on magnetorquers. The results are summarized in Table 7.6.

As for the simple 180° slew test cases, we performed a 100-trial Monte Carlo simulation over the randomized initial orbital conditions in order to analyze the final attitude error – the angle between the satellite body x-axis and $[\cos(10°), 0, \sin(10°)]^T$ in the ECI frame at the end of the trajectory. The results for the MTQ-only configuration are shown in Figure 7.15. The mean final attitude error is 16.16°, with a median error of 0.04°, and a standard deviation of 34.99°. Remarkably, 67% of trials achieve sub-degree precision, while 75% result in errors below 10°. These results represent a clear improvement compared to the 180° slew results for the same configuration, where only 11% of trials achieves sub-degree precision and 73% are within 10°. The increased flexibility of the reduced-attitude goal allows the trajectory planner to identify feasible solutions more effectively, leading to higher success rates and improved precision. With 67% of cases converging to within 1 degree in 500 seconds, this method shows incredible performance for a CubeSat, especially one that is magnetically controlled.

As good as the MTQ-only results are, the hybrid 3MTQ+1RW configuration's performance is significantly enhanced, as shown in Figure 7.16. The errors are more than an order of

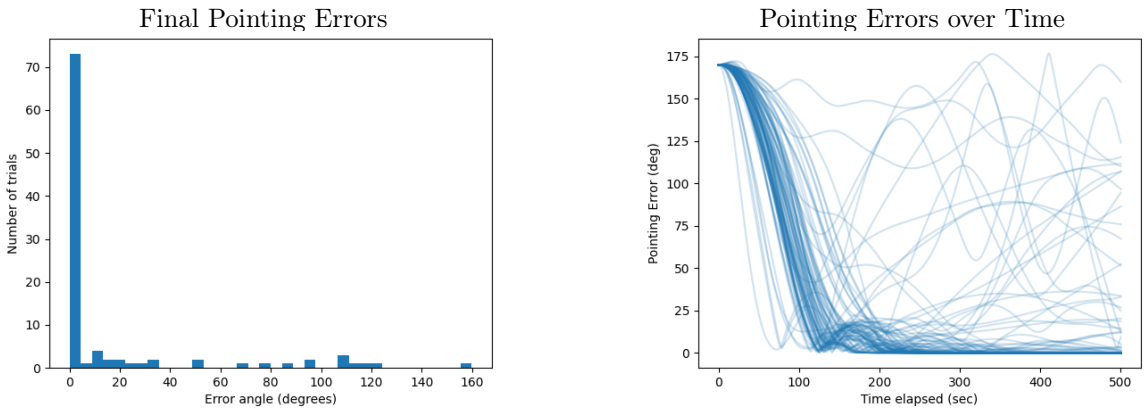


Figure 7.15: Results of 100-trial Monte Carlo simulation for planning reduced attitude goal slews for a magnetorquer-only satellite. Left: Final pointing error for all trials. Right: Visualization of pointing error over time for all trials. Test parameters can be found in Table 7.4.

magnitude less than the MTQ-only case. The mean final attitude error is 0.16°, with a median error of 2.6×10^{-3} °, and a standard deviation of 0.55°. Importantly, 96% of trials achieve sub-degree precision, and all trials have errors under 10°, consistent with the 180° slew results for this configuration. These results highlight the effectiveness of the reaction wheel in providing additional control torque, allowing for more precise pointing.

Figures 7.17 and 7.18 illustrate planned quaternion trajectories for both configurations. For MTQ-only satellites, an example of poor convergence is oscillatory behavior that slows but fails to reach the target within the 500-second timespan, as shown in the right panel of Figure 7.17. These cases are substantially less frequent than in the 180° slew trials, as can be seen by comparing the right panels of Figures 7.13 and 7.17, further demonstrating the benefits of reduced-attitude goals for underactuated systems. In contrast, the hybrid configuration achieves tighter control even in "poor" convergence cases, with trajectories oscillating around the target orientation and achieving errors below 5°, acceptable for many CubeSat missions, as seen in the right panel of Figure 7.18. These examples underline the advantages of reduced-attitude goals, particularly for underactuated systems, where the flexibility of the goal compensates for limited control authority.

The ability of the planner to balance pointing with constraints is also evident in the 3MTQ+1RW test cases, where the reaction wheels avoid saturation while meeting their pointing goals, as shown in Figure 7.19. Across all trials, the reaction wheel operates within 80% of its storage capacity, avoiding saturation. The mean final angular momentum is 5.9×10^{-4} Nms, with a median of 5.0×10^{-4} Nms and a standard deviation of 4.1×10^{-4} Nms. Compared to the 180° slew, the reduced-attitude goal allows for similar precision with slightly reduced reaction wheel usage, further validating its practical benefits.

By comparing these results to the 180° slew trials, the advantages of reduced-attitude goals become clear. For the MTQ-only satellite, the ability to achieve sub-degree pointing

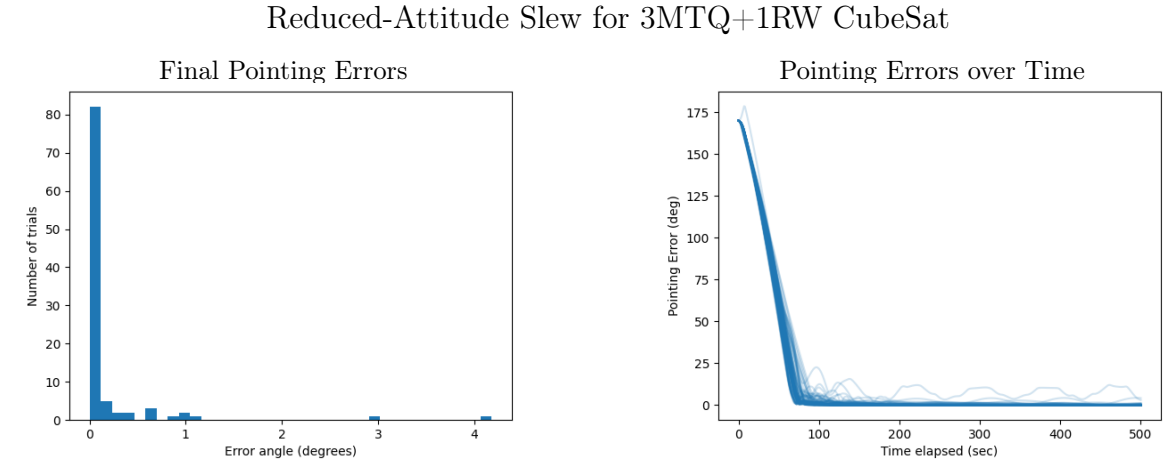


Figure 7.16: Results of 100-trial Monte Carlo simulation for planning reduced attitude goal slews for a 1RW satellite. Left: Final pointing error for all trials. Right: Visualization of pointing error over time for all trials. Test parameters can be found in Table 7.4.

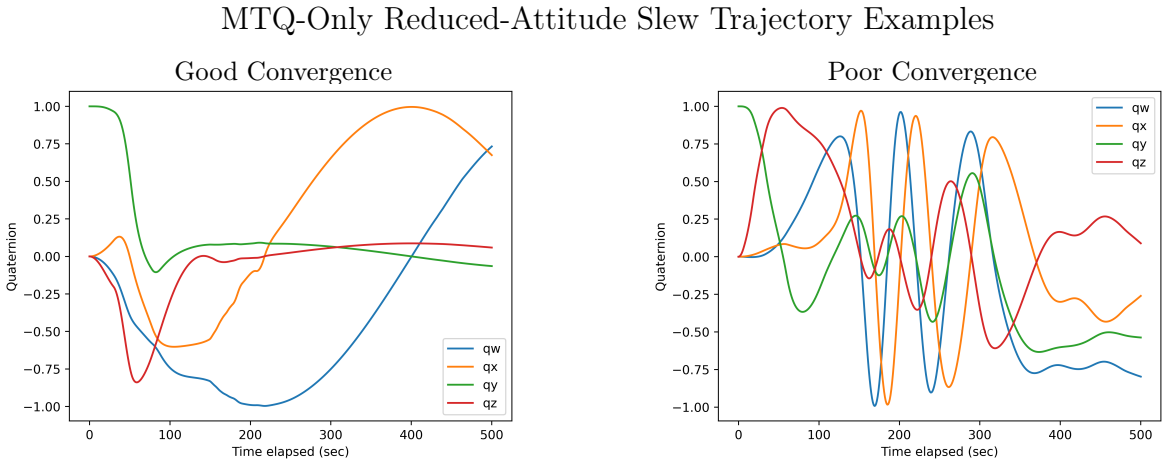


Figure 7.17: Two planned slews for a magnetorquer-only satellite. Left: "Good convergence" case, demonstrating a slew to target. Right: "Poor convergence" case, showing oscillation in attitude and slow convergence to the target. Test parameters can be found in Table 7.4.

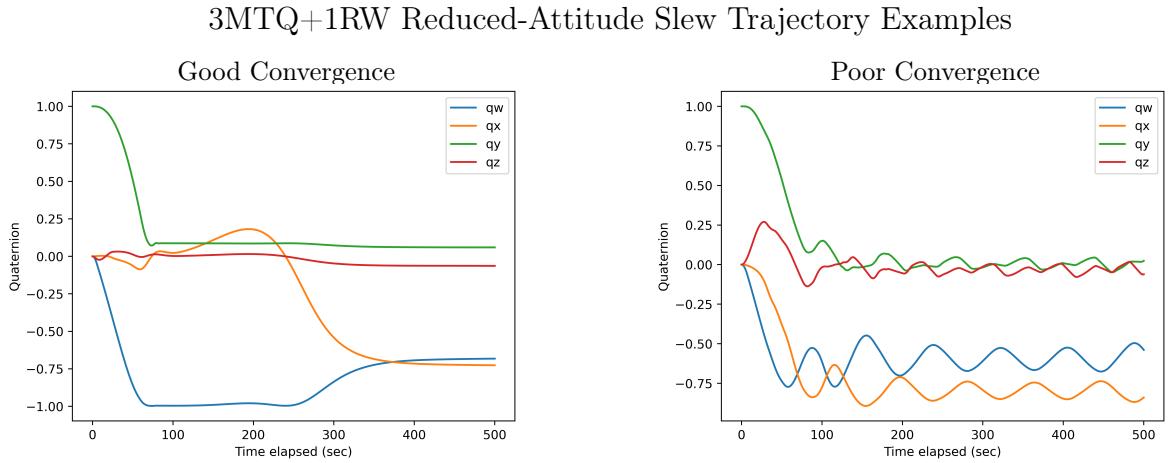


Figure 7.18: Two planned slews for a 1RW satellite. Left: "Good convergence" case, demonstrating a direct slew to target. Right: "Poor convergence" case, demonstrating a small oscillation around the target. Test parameters can be found in Table 7.4.

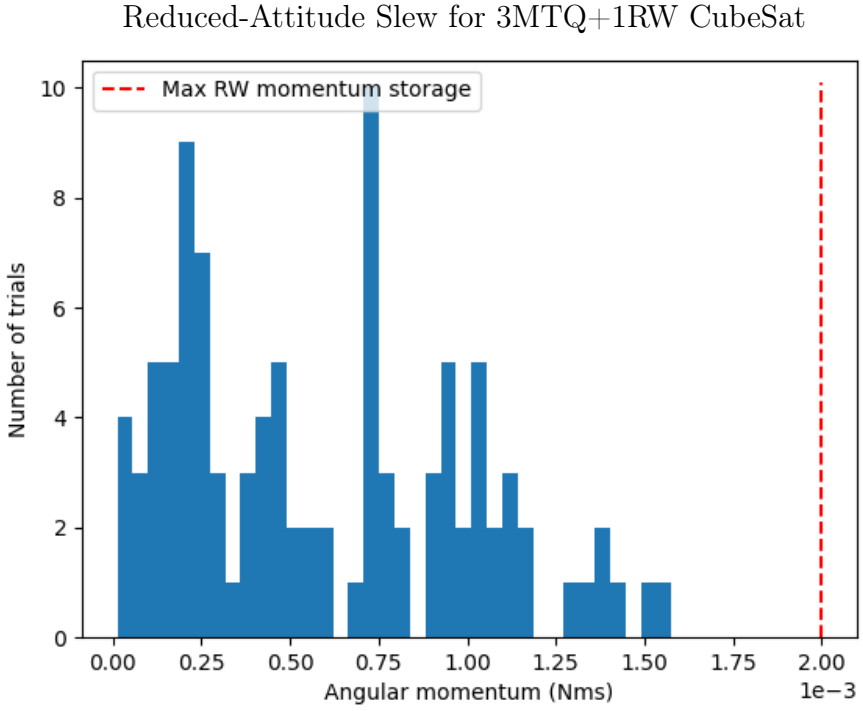


Figure 7.19: Results of 100-trial Monte Carlo simulation for planning reduced attitude goal slews for a 1RW satellite: final RW stored angular momentum for all trials. Test parameters can be found in Table 7.4.

in 67% of trials represents a substantial improvement over the 11% achieves in the simple slew case. This improvement underscores the importance of tailoring goals to match the capabilities of the control system. For the hybrid configuration, while performance is already excellent for both goal types, the reduced-attitude goal further enhances robustness and ensures greater consistency in pointing performance. These findings reinforce the utility of reduced-attitude goals for small satellites and CubeSats, particularly for missions with relaxed requirements on full three-dimensional attitude control but stringent requirements on single-axis pointing accuracy.

7.5.3.c Multiple Targets Slew

For the multi-target trajectories, we planned slews to align the satellite body x-axis with three distinct target vectors in the Earth-centered inertial (ECI) frame over a 500-second trajectory. The first target vector is $[\cos(10°),\ 0,\ \sin(10°)]^T$ (first 170 seconds), the second is $[\cos(-10°), 0, \sin(-10°)]^T$ (200 to 420 seconds), and the third target vector returned to $[\cos(10°),\ 0,\ \sin(10°)]^T$ again (450 to 500 seconds). Transition periods between these targets, spanning 30 seconds each, are not penalized for deviation from the target quaternion. This design allowed the planner to use these periods for optimal transitions rather than rushing to the next target to attempt precise pointing. As with the reduced-attitude goal slews, these trajectories constrained only a single satellite body axis, leaving other rotations unconstrained. As for the simple 180° slews, we performed a 100-trial Monte Carlo simulation over the randomized initial orbital conditions in order to analyze the attitude error – the angle between the satellite body x-axis and the ECI-goal frame vectors at the end of each targeting window. The results are summarized in Table 7.7.

The multiple target trajectories provide a challenging test case, particularly for underactuated systems. Rapidly adjusting to meet three distinct goals within a single 500-second window, while transitioning efficiently between them, represents a capability that is rare in CubeSat missions. This type of maneuver could enable imaging larger areas at higher resolutions or meeting multiple objectives during a single orbital pass, addressing the rapid retargeting technical gap identified in Section 2.3.2.d.

The results for the MTQ-only configuration are shown in Figure 7.20. The mean final attitude error across all trials is 5.7°, with a median error of 0.2° and a standard deviation of 21.3°. However, significant variability is observed between targets. The mean final error for the first target is 19.4°, decreasing to 12.1° for the second target and to 5.7° for the third target. This trend reflects the planner's difficulty in achieving precise alignment for early targets while still preparing for subsequent transitions. The final attitude error for the MTQ-only satellite often increased toward the end of each pointing period, as shown in Figure 7.20b, suggesting the satellite is positioning for the next target, which may affect these measurements (this would not be an issue for the final target, potentially explaining its lower value).

For the hybrid 3MTQ+1RW configuration, the results are significantly better, as shown in Figure 7.21. The mean final attitude error across all trials is 0.45°, with a median error of 0.03° and a standard deviation of 1.77°. The final attitude error for the first, second, and third targets is 2.22°, 1.53°, and 0.45°, respectively, demonstrating precise and consistent pointing throughout the trajectory. Similar to the MTQ-only case, the performance is better

Metric	Target	MTQ-Only	3MTQ+1RW
% Within 1 Degree	Target 1	1%	37%
	Target 2	0%	56%
	Target 3	77%	91%
% Within 10 Degrees	Target 1	13%	98%
	Target 2	83%	99%
	Target 3	91%	99%
Mean Error (deg)	Target 1	19.4	2.22
	Target 2	12.1	1.53
	Target 3	5.7	0.45
Median Error (deg)	Target 1	18.6	1.60
	Target 2	4.8	0.78
	Target 3	0.22	0.03
Std Dev Error (deg)	Target 1	11.4	2.37
	Target 2	14.5	2.11
	Target 3	21.3	1.77

Table 7.7: Pointing Accuracy in Monte Carlo Simulations for the Multi-Target Slew Case. Test parameters can be found in Table 7.4.

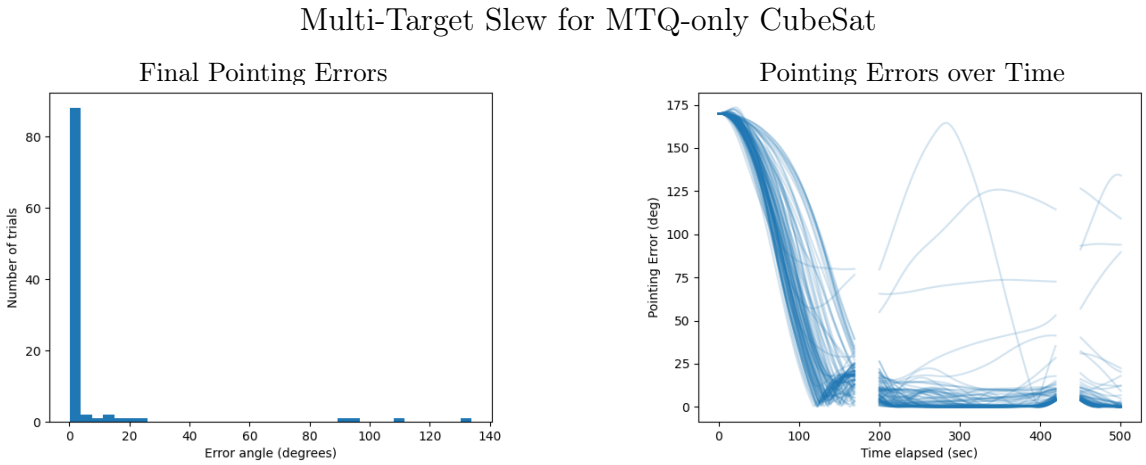


Figure 7.20: Results of 100-trial Monte Carlo simulation for planning multiple target trajectories for a magnetorquer-only satellite. Left: Final pointing error for all trials. Right: Visualization of pointing error over time for all trials. Gaps show the "free time" or transition periods during which there is no goal and no cost to deviation. Test parameters can be found in Table 7.4.

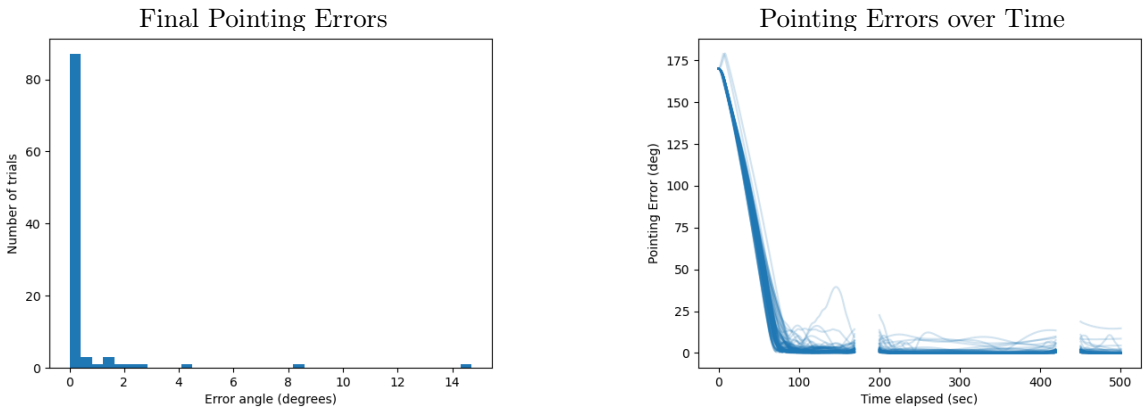


Figure 7.21: Results of 100-trial Monte Carlo simulation for planning multiple target trajectories for a 3MTQ+1RW satellite. Left: Final pointing error for all trials. Right: Visualization of pointing error over time for all trials. Test parameters can be found in Table 7.4.

for later targets, which might stem from the planner including efforts to prepare for later pointing during the earlier periods. The hybrid configuration achieves less than 10° error in 98% of trials for the first target and 99% for the second and third targets. Additionally, sub-degree precision is achieved in 91% of trials for the third target and 56% for the second, but only 37% for the third. Considering the right image in Figure 7.21, we can see that the 3MTQ+1RW hybrid architecture results do not show the same level of "lifting up" near the end of pointing goals, though it might be occurring at the end of the first pointing period. The lack of this phenomenon shows the benefits of the increased torque capability from the added reaction wheel.

Figures 7.22 and 7.23 illustrate planned quaternion trajectories for both configurations. The planned attitude in the MTQ-only "poor" convergence case shown in Figure 7.22 shows oscillation that slows but does not approaches the target in the short timespan. This may arise from the planner trying to over-optimize across all three goals simultaneously. This oscillation is characteristic of the planned magnetorquer-only trajectories that result in more than 10° of final attitude error. For the MTQ-only satellite, even in "good convergence" cases, drift occurs during the imaging periods, and the satellite does not remain stationary at any of the three targets, though this appears to largely be rotation around the imaging axis. In contrast, the hybrid configuration achieves precise pointing at each target, with barely any oscillations around the target orientations. This difference highlights the limitations of magnetic control in handling rapid transitions and the hybrid system's ability to manage complex trajectories with high accuracy.

The rapid slewing and adjustments demonstrated by both configurations are impressive, especially for CubeSats. However, the magnetorquer-only system struggles with the inherent limitations of underactuation, particularly for the early targets. By leveraging the additional control authority provided by the reaction wheel, the hybrid configuration excels in this scenario. The trajectory planning approach is critical here, enabling the uncommon hybrid

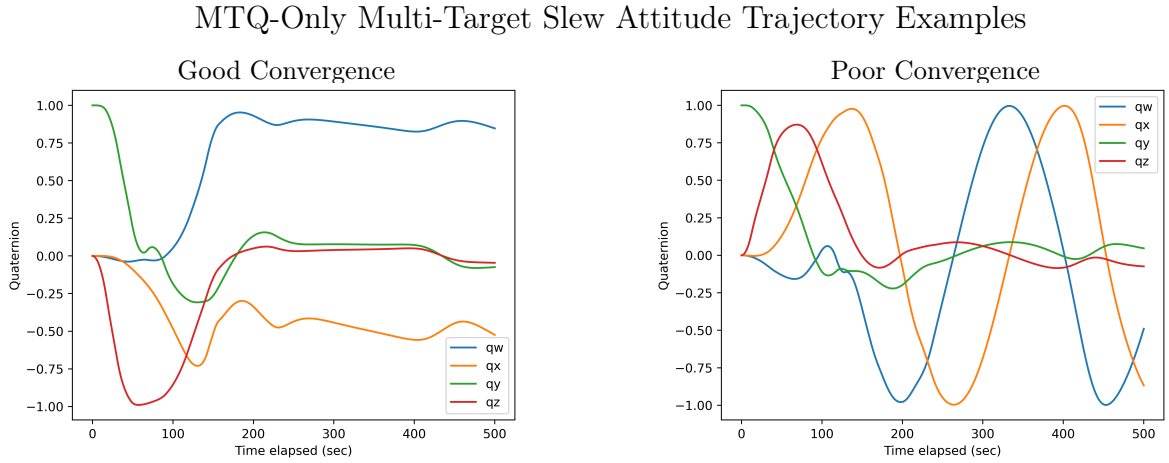


Figure 7.22: Two planned slews for a magnetorquer-only satellite. Left: "Good convergence" case, demonstrating slewing to multiple targets. Right: "Poor convergence" case, showing oscillation in attitude and limited convergence to the target. Test parameters can be found in Table 7.4.

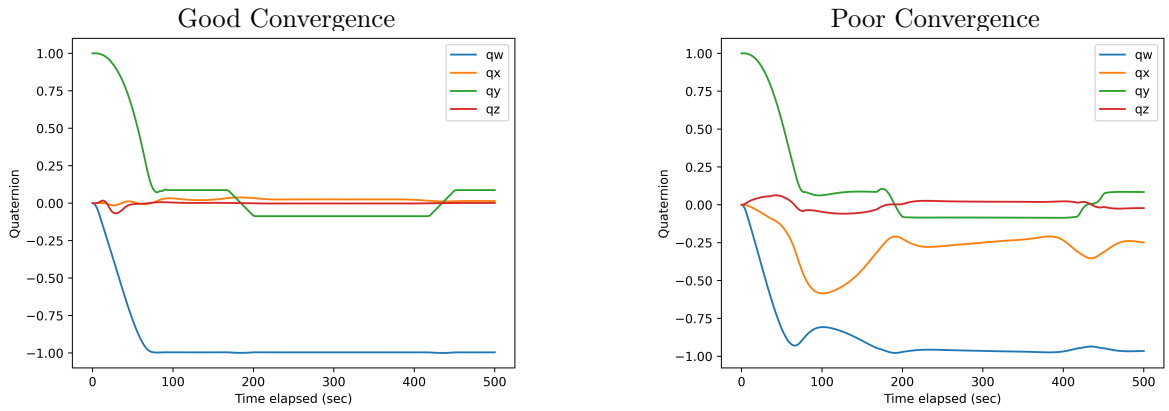


Figure 7.23: Two planned slews for a 3MTQ+1RW satellite. Left: "Good convergence" case, demonstrating slewing to multiple targets. Right: "Poor convergence" case, demonstrating only approximate convergence to the target. Test parameters can be found in Table 7.4.

system to plan and execute complex, time-constrained trajectories that traditional control laws would struggle to manage. This capability paves the way for advanced CubeSat missions that require rapid retargeting or sequential imaging within a single orbital pass.

7.5.4 Sequential Trajectories

Many of the other examples in this chapter show the planner applied to a single long trajectory. In operation, however, the planner would be run repeatedly with a finite time horizon as described in Section 7.3.2. This section shows how sequential trajectories can perform in simulation. The approach from Section 7.3.2 is applied to a 6U CubeSat with three reaction wheels, complete with an estimator and modeled disturbances. Based on simulated orbits, external dynamics, and sensor information, the system repeatedly generates a 450s trajectory (with 150s of overlap and 100s of precalculation) that attempts to meet a variety of time-varying goals. Details can be found in Table 7.8. The trajectory generated is shown in Figure 7.24, and its angular velocity in Figure 7.25. The pointing accuracy for this can be seen in Figure 7.26, and its small-angle performance is better shown in a log-scale plot in Figure 7.28. The trajectory command can be seen in Figure 7.27.

Table 7.8: Satellite Properties and Test Parameters for Sequential Planning Test

Property	**Value/Description**
Mass[a]	$m = 10.165$ kg
Inertia Matrix[a]	$J = \text{diag}\left(\begin{bmatrix}0.0969 & 0.1235 & 0.1918\end{bmatrix}\right)$ kg·m^2
MTQ	Three-axis, identical on each Maximum Magnetic Moment: 5 Am^2 Initial Bias:[b] $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ Noise Std Dev: 0.0001 Am^2 s$^{0.5}$ Bias Drift Rate:[b] 0 Am^2/s$^{0.5}$
Gyroscope	Initial Bias:[c] $\frac{0.1}{\sqrt{11}} * \begin{bmatrix}1 & -1 & 3\end{bmatrix}$ deg/s Noise Std Dev:[c] 0.0004 deg/s$^{0.5}$ Bias Drift Rate:[c] 0.03 deg/s$^{1.5}$
Magnetometer	Initial Bias: $[-9.948, -0.199, -0.995]$ nT Noise Std Dev: 300 nT·s$^{0.5}$ Bias Drift Rate: 1 nT/s$^{0.5}$
Coarse Sun Sensors	Efficiency: 0.3 Initial Bias: $[0.0015, 0.0027, -0.009]$ (unitless) Noise Std Dev: 0.0003 s$^{0.5}$ Bias Drift Rate: 0.000003 s$^{-0.5}$
Propulsion Disturbance	Initial Value: $[0.012, -0.992, 0.124]$ μNm Maximum Magnitude: 100 μNm Drift Rate: 0.1 μNm/s$^{-0.5}$ Applied during 30s-50s, 400s-1100s, and 2800s-3600s.
Other Disturbances	Gravity gradient, drag, and solar radiation pressure are also included, in both the estimator and the "real" satellite.[g]
Planner Disturbances	Gravity gradient and propulsion.[g]
Orbit Parameters	Geocentric circular orbit Altitude: 450 km

Continued on next page

Table 7.8 – *Continued from previous page*

Property	**Value/Description**
	Inclination: 87° J2 effects included
Update Period	1 second
Simulation Duration	3600 seconds
Control Algorithms	No Control for first 5 seconds. B-dot applied from 5s until 150s ALTRO-based Plan and Track Algorithm applied after 150s mark
Pointing Goals	0s to 150s: None (no control or B-dot) 150s to 1100s: point body $-x$-axis anti-ram; 1100s to 1200s: no goal; 1200s to 1500s: point body z-axis nadir; 1500s to 1600s: no goal; 1600s to 1900s: point body z-axis zenith; 1900s to 2000s: no goal; 2000s to 2400s: point body z-axis towards orbit normal; 2400s to 2500s: no goal; 2500s to 3600s: point body $-x$-axis anti-ram;
Initial State	Angular Velocity: $[0, 0, 0]$ deg/s Quaternion: $[0.153, 0.685, 0.695, 0.153]$
Planner Time Step	1s, 10s coarse time step first.
Trajectory Duration	450s, including 150s of overlap
Precalculation	100s

[a] Based on the physical properties of ASTERIA [1].
[c] Similar to MEMS gyros [27][25].
[g] These disturbances are modeled with exact information as described in Section 4.3.4. However, there may be discrepancies between the disturbances on the "real" and virtual satellites from orientation estimation error. For example, if the estimated orientation is such that the major principal axis of the satellite is pointing nadir, it will estimate zero gravity gradient torque, while the "real" satellite may experience a torque corresponding to its actual orientation.

The results presented here demonstrate the efficacy of sequential trajectory planning in creating feasible trajectories, even when there are multiple goals, and achieving rapid, autonomous retargeting for satellite missions. By employing the ALTRO-based planner from Section 7.2 and the trajectory optimization framework from Section 7.3, the satellite successfully transitions between multiple pointing goals while adhering to actuator constraints and disturbance limits. As illustrated in the pointing error plots (Figures 7.26 and 7.28), the planned trajectories maintain sub-degree pointing accuracy throughout the simulation. As shown in Figure 7.24, the trajectory is rather complicated, including a high rate of spin between 2000 and 2500 seconds (this is supported by Figure 7.25). Notably, the angular velocity plot (Figure 7.25) reveals precise control of rotational rates, avoiding overshoots and

ensuring a stable approach to each target. This complex trajectory does not require large amounts of control effort, as shown in Figure 7.27, indicating the trajectory is efficient despite its complexity. This trajectory would not be found by a simple control law, but can only be found through a nonlinear trajectory solver, such as ALTRO. These results highlight the robustness of the method, even under challenging conditions such as underactuated magnetic control, where traditional methods often struggle to converge or meet performance criteria. Sequential trajectory planning allows the satellite to dynamically adapt to time-varying mission objectives, such as alternating between imaging and communications tasks. It also allows it to adjust to more recent information, as the plan is always more recent than planning one large trajectory a long time ahead.

An interesting detail can be seen in Figure 7.28. The spikes that are visible in this and Figures 7.27 and 7.25 occur at the start of trajectories–which should be aligned with the prior trajectory which is used as a starting point. Why is there a spike as the satellite adjusts, when the prior trajectory was already very close? In fact, the satellite at the start of each trajectory is in an appropriate orientation compared to the previous trajectory. However, this orientation is not appropriate for the goals! The error (and correcting control torque) comes from the fact that the trajectory planner uses internal orbit propagation to give the pointing goals to the planner. There is some error in the orbit propagation that is being used (an area for improvement, as developing a new orbit propagator was not part of this work), and it is based off noisy GPS values, so that propagation diverges from the actual position of the satellite–meaning that directions such as nadir and zenith are slightly off in the planner and the flight. Running a similar trajectory with the directions replaced with ECI-frame axes removes this problem–indicating a better orbit propagator would fix it as well.

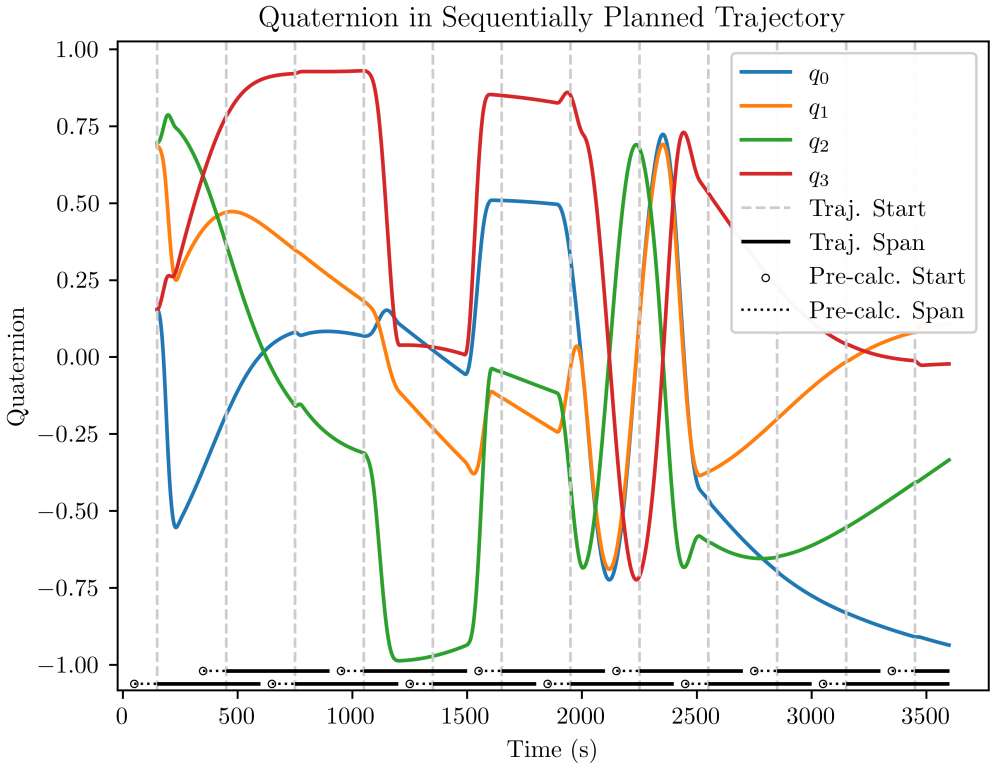


Figure 7.24: The quaternion over time for a CubeSat trajectory that is planned in increments. Test details can be found in Table 7.8. Note the trajectory start times, overlapping trajectory spans, and precalculation time, as described in Section 7.3.

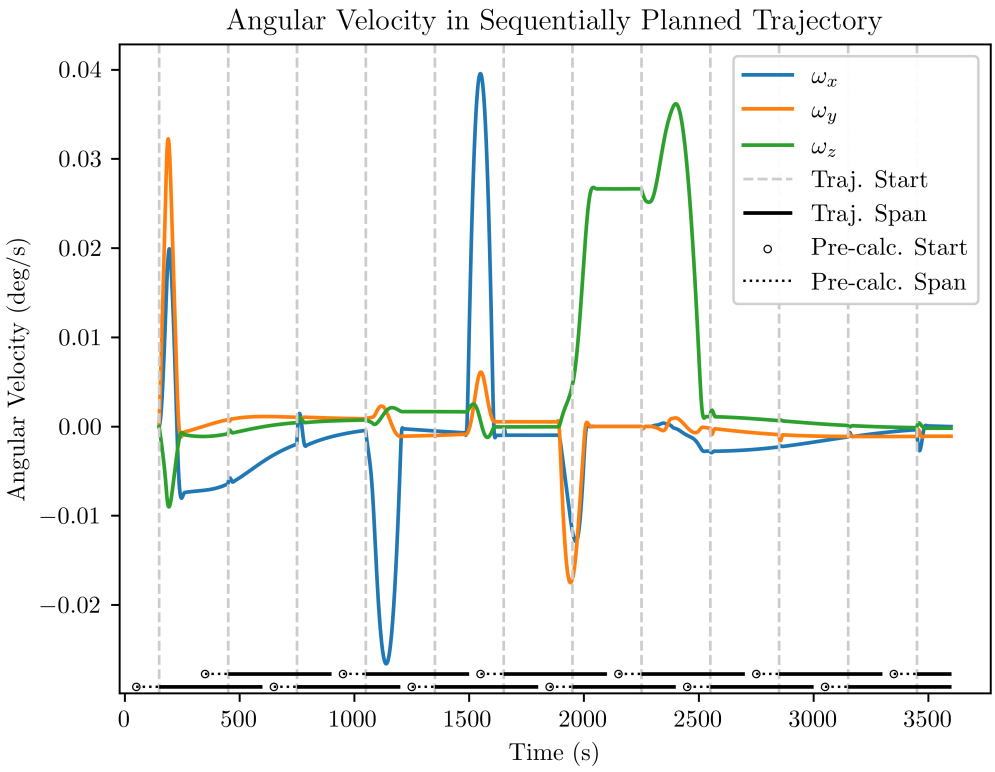


Figure 7.25: The angular velocity over time for a CubeSat trajectory that is planned in increments. Test details can be found in Table 7.8.

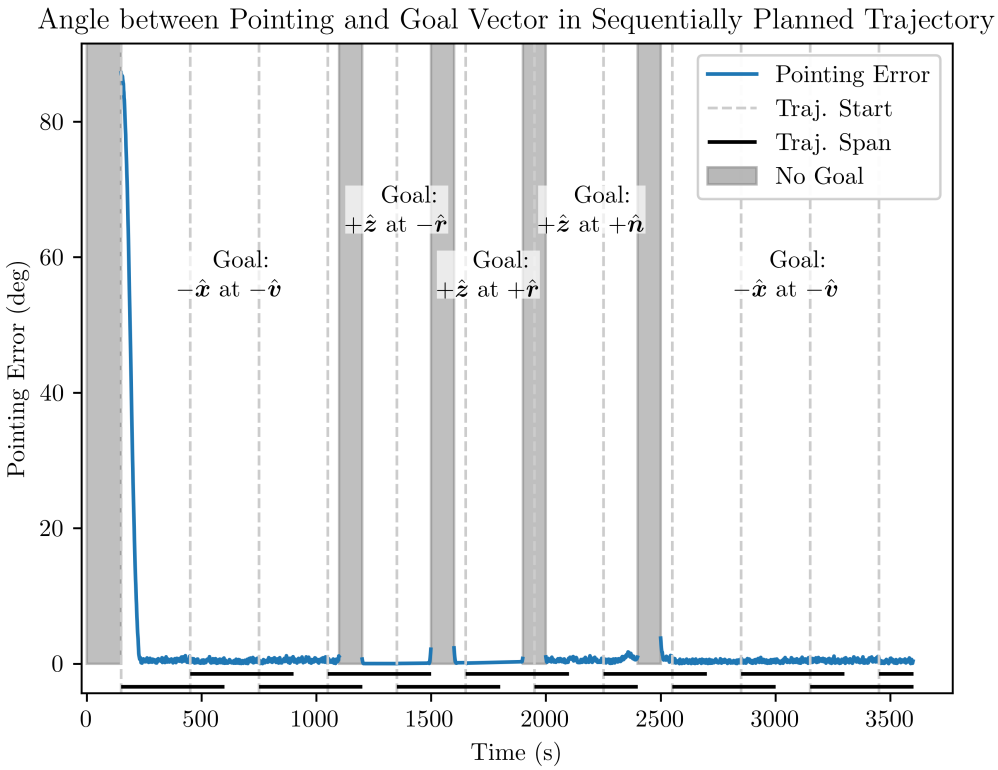


Figure 7.26: The pointing error from goals over time for a CubeSat trajectory that is planned in increments. Test details can be found in Table 7.8. The vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are the body-frame axes, while $\hat{\mathbf{r}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{n}}$ represent world-frame directions corresponding to zenith, ram, and orbit normal, respectively.

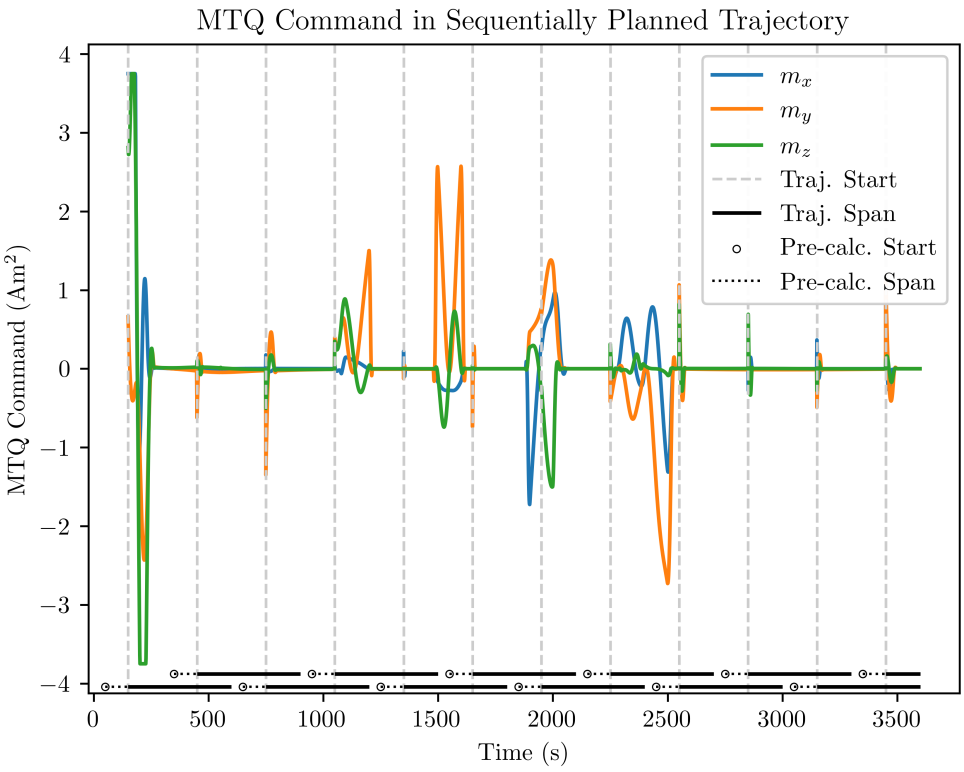


Figure 7.27: The command over time for a CubeSat trajectory that is planned in increments. Test details can be found in Table 7.8.

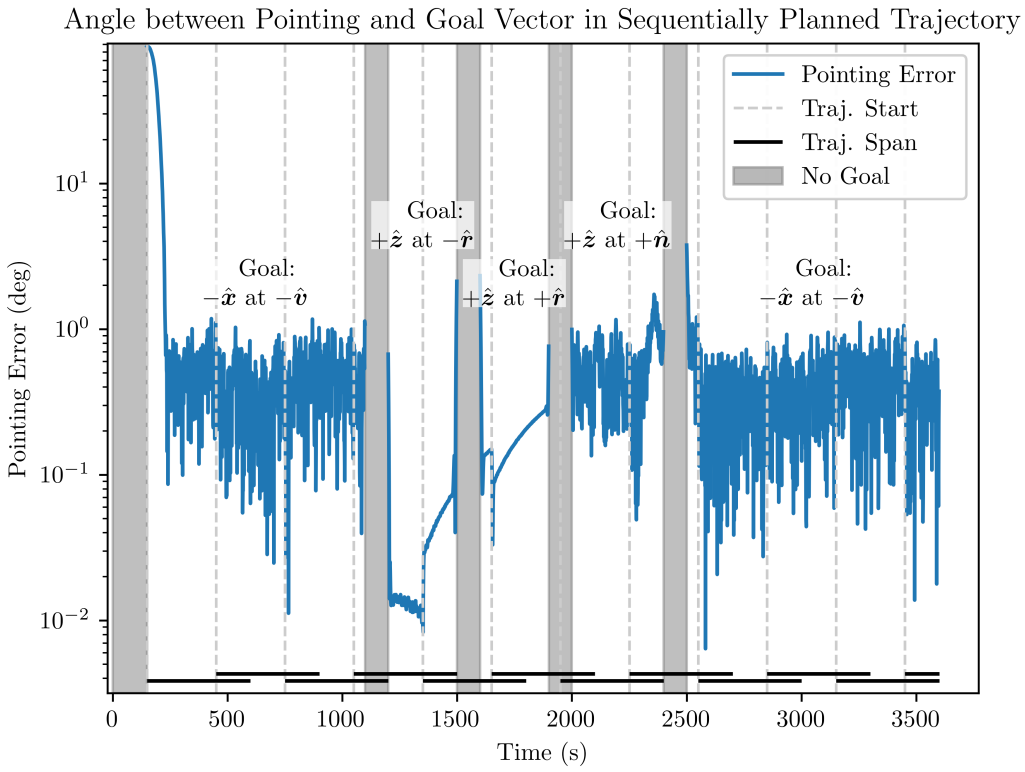


Figure 7.28: The log-scale pointing error from goals over time for a CubeSat trajectory that is planned in increments. Test details can be found in Table 7.8. The vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are the body-frame axes, while $\hat{\mathbf{r}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{n}}$ represent world-frame directions corresponding to zenith, ram, and orbit normal, respectively.

7.6 Conclusion

This chapter presents a comprehensive framework for autonomous trajectory planning within Attitude Determination and Control Systems (ADCS), addressing key challenges of adaptability, precision, and autonomy in modern satellite operations. By enabling satellites to compute and execute time-varying trajectories in real-time, this work provides a foundation for enhanced mission flexibility and operational efficiency. The methods developed here build upon the dynamic-aware estimation and disturbance-aware control strategies explored in earlier chapters, integrating them into a cohesive planning framework. By incorporating dynamic system models and building on the unified modular architecture described in Chapter 4, this approach ensures compatibility with a wide range of satellite architectures, from CubeSats relying solely on magnetic torque to larger, more capable spacecraft requiring precise multi-phase trajectory planning.

The work in this chapter is based on the ALTRO trajectory planning algorithm, modified for use as part of a real-time satellite ADCS, to solve trajectory planning problems efficiently. This method is tailored to address the challenges of underactuated systems, actuator constraints, and dynamic disturbances. The algorithm and modifications are described in Section 7.2. However, the algorithm is not enough on its own to allow planning-enabled ADCS. There are other considerations required for its implementation including accounting for calculation time and incorporating estimations of system state. These topics are discussed in Section 7.3. Additionally, once a trajectory is generated, the satellite must be controlled to follow it under disturbances, noise, and uncertainty. This question is discussed in Section 7.4 and two methods are described: an LQR-based method which is linearized around the trajectory, and a more advanced, adaptive approach based on model predictive control. Section 7.5 validates the framework across diverse scenarios, demonstrating its effectiveness in achieving precise pointing goals, dynamic retargeting, and smooth trajectory transitions. For instance, a long-term trajectory is planned for an underactuated CubeSat that contains seamless and quick transitions between tracking a target on the ground and pointing a thruster anti-ram. A CubeSat with a significant propulsion disturbance (one that exceeds the strength of the spacecraft's actuators) is still able to plan a trajectory with sub-degree pointing accuracy by spinning around the propulsion axis, averaging the torque out over the rotations. The algorithm is also tested for satellites with varying initial conditions in Monte Carlo simulations for a simple slew, slews to a goal set (as described in Chapter 5), and slews between multiple targets, all in 500 seconds. For the single, 180 degree slew, the results for a satellite with only magnetorquers are typically $< 10^\circ$ (73% of trials) at the end of the slew, and are almost always within 20°, while adding a single reaction wheel results in errors usually within 1° (96% of cases) and always within 2°. For the goal-set slews, the magnetorquer-only satellite is within 1° in 67% of cases, while the CubeSat with one RW added has sub-degree precision in 96% of cases. When switching between three targets (all in 500 seconds), the satellite with one RW is within 10 degrees in 98% or more of all cases for each target, and has sub-degree precision for 37% of cases for the first target, 56% for the second, and 91% for the third. The magnetorquer-only satellite typically is not within one degree (except for the third target, when it is correct for 77% of cases), but is within 10 degrees in 13% of cases for the first target, 83% for the second, and 91% for the third. These performances, especially on a 500 second

time frame, are generally much better than standard pointing abilities for underactuated CubeSats. Sequential trajectories for changing goals is also demonstrated with sub-degree pointing accuracy for a 6U CubeSat with only 100 seconds to change between goals.

While the work presented here marks a significant advancement in trajectory planning, opportunities for future work remain. This approach has many parameters and hyperparameters, and the author has only explored some of them. Investigation is needed into what setup (weights, time horizons, optimization parameters, etc.) results in the best trajectories and lowest computational effort. Similarly, the effect of different trajectory iterations in the planner (coarse versus fine, disturbance inclusion, etc.) should be considered further. Simple heuristics for determining when a trajectory is acceptable or when the planner should be re-run need to be developed, and it should be considered whether it is possible to predict mid-optimization if the trajectory is likely to converge.[8] Various improvements to the planner's cost function could be made, particularly for the underactuated case. One example might be to focus not just on minimizing the base cost but to include a term that weights the cost's stability over the state (perhaps using the cost-to-go derivative) as part of the objective function. This could result in trajectories that are less perfect but more stable to estimation error or disturbances. Other variations to the objective function should be explored, such as using arccosine (or its polynomial expansion) in the angular cost. Another improvement could be including multiple objectives in the objective function, and developing a method to prioritize or weight the objectives differently based on what is feasible (such as relying on a goal set to align two vectors when necessary, but fully specifying an orientation when possible and not costly). The trajectory overlap could be better used, such as using the prior trajectory (at least that which overlaps) as a warm-start for the next trajectory, or hybridizing between the trajectories when there is overlap to smooth the transition. Greater exploration of the methods for following a generated trajectory, particularly for underactuated satellites, is certainly needed. One possibility would be the incorporation of directional error (for example, allowing the satellite to be off around the imaging axis) based on the directional goals used in this planner and detailed in Chapter 5. There are further modifications to make the planner interface better with the real world. While the blips caused by orbit propagation in Section 7.5.4 are interesting, a flight-ready version should take that into account, along with understanding other potential divergences, such as local magnetic field deviations (this could be tracked as a variable). Lastly, real-time implementations using hardware-in-the-loop (HIL) systems or onboard testing would provide critical validation for operational readiness.

The concepts and work laid out in this chapter tie to the key components of this thesis: starting from the areas of work in Section 3.2, it builds the contributions in Section 3.3 to address the technical gaps in Section 2.3.2. This chapter directly comes from the **Attitude Trajectory Planning** area of work (Section 3.2.4) and is the most direct application of the **Trajectory Planning** contribution (Section 3.3.3), directly targeting the **ADCS Trajectory Planning** technical gap (Section 2.3.2.c). By using efforts from the **Generalized ADCS** area of work in Section 3.2.5, this planner is applicable to a wide variety of satellites and missions, helping to bridge the **General ADCS Approach** technical gap in Section 2.3.2.a.

[8] Having watched the planner iterate over trajectories, the author feels he developed an instinct for when the planner looked to be heading in a bad (or inefficient) direction. Further study could result in an autonomous version of this.

The planner works on satellites independent of their size and scale, actuators, sensors, goals, orbits, or disturbances. In any of these cases, it can generate trajectories that attempt to meet the goals and constraints and typically meet them quite well, with a performance and flexibility exceeding most typical control laws, especially for satellites that have actuation difficulties. Therefore, the work in this chapter is a clear part of the **ADCS Architecture for Weak or Underactuated Satellites** contribution in Section 3.3.5. When applied to magnetically-controlled satellites, this serves as a big step towards closing the **Effective Magnetic Control** technical gap. By developing a trajectory planning framework that supports time-varying goals and dynamic adaptability, the work forms a key component of the **Improved Autonomy for ADCS** contribution, contributing to the goals of enabling autonomous mission execution and reducing reliance on pre-planned ground control operations. This autonomy, combined with the rapid goal-switching seen in Section 7.5.4, helps to close the **Autonomous Rapid Retargeting** technical gap, providing methods to bridge the divide between static planning or control laws and the requirements of dynamic, multi-objective missions.

The integration of trajectory planning with the estimation, goal specification, and control frameworks developed in Chapters 4 - 6 highlights the interconnected nature of the contributions in this dissertation. Together, these components support the dissertation's overarching vision of a fully autonomous, adaptable ADCS framework capable of executing complex satellite missions with minimal operator intervention for a range of missions and spacecraft. This chapter advances the state of the art in trajectory planning and opens pathways for continued innovation in autonomous satellite systems.

Chapter 8

Summary and Future Work

This work explores a variety of methods to improve satellite ADCS, including ways to make these systems more generalizable, more adaptive, and more autonomous, even for systems that are underactuated, have lower-cost sensors and actuators, or have many disturbances. The individual results are summarized in Section 8.1 and tied to contributions in Section 8.2. Future work and extensions are in Section 8.3. Final remarks can be found in Section 8.4.

8.1 Result Summary

This work presents computational methods that significantly enhance the performance of Attitude Determination and Control Systems (ADCS) across a wide spectrum of missions and applications. By optimizing core ADCS components, such as control laws, estimators, and trajectory planning, this research delivers substantial performance improvements, particularly for satellites constrained by cost, disturbances, or underactuation. The innovations presented here establish a robust and adaptable ADCS framework suitable for diverse satellite configurations.

In Chapter 4, a framework for satellite modeling and state estimation is developed, achieving notable gains in accuracy and versatility. For satellites equipped with low-quality sensors and subjected to disturbances, the estimators demonstrate an improvement of more than two orders of magnitude in state estimation accuracy, as evidenced by the TRMM satellite results in Sections 4.4.2 and 4.4.3. These cases validate sub-degree pointing accuracy even under challenging conditions. This is extended to impressive $\approx 0.1^\circ$ level estimation for a CubeSat (a very different type of satellite, with lower-quality sensors) in Sections 4.4.4 and 4.4.5. Furthermore, Section 4.4.8 highlights the system's ability to simultaneously track diverse parameters, including sensor and actuator biases, propulsion disturbances, and internal dynamics. This capability enhances ADCS performance and provides critical insights for satellite debugging and optimization, representing a significant advancement over traditional estimation techniques.

Chapter 5 explores strategies to simplify high-level ADCS operations. The novel approach of specifying goals as vector alignments, rather than overly-defined 3D orientations, proves effective. Sections 5.3.5.a and 5.3.5.b demonstrate the utility of this method, finding simple slews while incorporating constraints like keep-out zones. These results support more intuitive

and flexible mission planning, enabling satellites to adapt dynamically to evolving mission requirements and environments. This approach also expands the operational envelope of underactuated systems, particularly magnetically-controlled satellites, by optimizing goal specification.

In Chapter 6, the research addresses the challenge of disturbance management in satellite control. By incorporating disturbances into control laws as feed-forward terms, this work achieves improvements in pointing accuracy—reducing errors by up to 20 degrees, as detailed in Sections 6.5.1.c, 6.5.2.a, and 6.5.2.c. Additionally, the introduction of the "all-in-one" disturbance term in Section 6.3 provides a practical solution for environments with incomplete disturbance data, with its efficacy shown in several sections, including Sections 6.5.1.b, 6.5.2.a, and 6.5.2.c. While slightly less precise than detailed disturbance models, this approach offers robust enhancements without requiring prior knowledge, making it applicable to a range of missions.

Chapter 7 introduces an adaptive approach to trajectory planning tailored for diverse satellite configurations. Extensive testing across scenarios demonstrates strong performance. Section 7.5.1 showcases its effectiveness for underactuated, magnetically-controlled systems, including multiple goals, while Section 7.5.4 highlights its ability to handle sequential trajectories. The planner's ability to integrate with real-world architectures and execute pre-generated trajectories with precision establishes it as a valuable tool for autonomous ADCS.

The results of this work demonstrate significant progress across all components. From improved state estimation accuracy to innovative goal specification methods, robust disturbance handling, and advanced trajectory planning, this research provides a strong foundation for autonomous, generalizable, and high-performance satellite systems. These advancements address key challenges in modern ADCS design and offer practical solutions for future satellite missions.

8.2 Contributions Revisited

The results shown in Chapters 4-7 and summarized in Section 8.1 relate to the various contributions from Section 3.3. These contributions are reviewed here, with specific examples from test cases to further emphasize them and their real-world applicability.

8.2.1 Generalized Disturbance- and Dynamics-Aware Estimator

The first contribution is a generalized and modular attitude estimation system that incorporates disturbances, dynamics, and control effort for a wide variety of spacecraft, sensors, actuators, and disturbances. This is a direct effort towards addressing the need for a **General ADCS Approach** to estimation, described in Section 2.3.2.a. The relevant work (from the **Improved Attitude State Estimation** and **Generalized ADCS** areas of work in Sections 3.2.1 and 3.2.5, respectively) can largely be found in Chapter 4, which includes information on:

- the derivation of modeling satellite state, including angular velocity, orientation represented as a quaternion, stored angular momentum, time-varying disturbing torques and magnetic moments, and sensor and actuator biases;

- methods to encompass various sources of uncertainty in satellite modeling, including sensor and actuator noise and biases alongside integration error; and
- methods to easily create estimators (including different types of nonlinear Kalman filters) for systems with a variety of physical properties, actuators, sensors, and disturbances, based on the properties and noise behavior of differing satellites.

The results in Chapter 4 demonstrate the flexibility and accuracy of these estimators and their potential value to small spacecraft with lower-quality sensors. For instance, Sections 4.4.2 and 4.4.3 show sub-degree pointing accuracy for the TRMM satellite, and Sections 4.4.4 and 4.4.5 show $\approx 0.1^\circ$ accuracy for a CubeSat. These outcomes demonstrate their potential value for both large spacecraft and small spacecraft with lower-quality sensors. Furthermore, Section 4.4.8 highlights successful tracking of valuable dynamic variables like actuator biases and propulsion disturbances.

8.2.2 Disturbance Modeling for Control Inclusion

The second contribution is a set of methods for including a variety of expected disturbances in satellite attitude control. This is a specific effort toward the **General ADCS Approach** Technical Gap in Section 2.3.2.a, with the idea of making disturbance inclusion easier to do in the general case, not just for highly specific missions or control laws. The relevant work (largely connected to the **Direct Countering of Disturbances** area of work in Section 3.2.3) is largely in Chapter 6, and includes:

- an analysis of how different disturbances can be effectively and efficiently modeled for use in real-time algorithms;
- a method of real-time estimation of time-varying disturbances (magnetic dipole and bias) and methods to include these in planning or correct during control;
- methods to include analytical disturbances (propulsion and gravity gradient) in planning, estimation, and control;
- consideration of a potential "all-in-one" torque term calculated by an estimator (see the generalized estimator contribution in Section 8.2.1) for cases where disturbances cannot be precisely modeled; and
- inclusion of both modeled and all-in-one disturbances in a trajectory planning algorithm and other control approaches to improve pointing.

The results in Chapter 6 demonstrate the use of this feed-forward disturbance control term in various (simulated) satellite missions. For example, Sections 6.5.1.c, 6.5.2.a, and 6.5.2.c show a 20-degree reduction in pointing errors when disturbances are included. Additionally, the "all-in-one" disturbance term is validated in several sections, including Sections 6.5.1.b, 6.5.2.a, and 6.5.2.c.

8.2.3 Trajectory Planning

The third contribution is the creation of an attitude trajectory planner and accompanying closed-loop controller for spacecraft with changing pointing goals that include disturbances (gravity gradient, SRP, drag, propulsion, and a general form) and that can be used for a variety of actuator arrangements, directly addressing the **Trajectory Planning** Technical Gap described in Section 2.3.2.c. Tying directly to the **Attitude Trajectory Planning** (Section 3.2.4) and **Generalized ADCS** (Section 3.2.5) areas of work, it is detailed largely in Chapter 7, which includes information on:

- the trajectory planner details, including problem formulation and algorithmic solving approach, as well as the seed trajectory and constraint enforcement;
- the variability across satellites with different disturbances and actuator arrangements;
- the primary use of being always-on, continuously planning trajectories to meet a set of (potentially varying) goals, instead of being used for specific slews or other operations;
- closed-loop control methods; and
- the overarching control system that includes command interpretation, scheduling of planning, management of calculation time and exceptions, and that deals with convergence failures.

The results in Chapter 7 demonstrate the planner's use in a variety of scenarios, with impressive results for many underactuated satellite control challenges. The results shown in Section 7.5 show the versatility and ability of the planner-based ADCS, including meeting multiple goals while underactuated (Section 7.5.1), meeting goals under an overwhelming disturbance (Section 7.5.2), and meeting many goals while sequentially planning based on estimated data (Section 7.5.4). The results in Sections 7.5.1 and 7.5.3 also indicate progress on the **Effective Magnetic Control** technical gap (Section 2.3.2.b), while the multi-target results in Sections 7.5.1, 7.5.3, and 7.5.4 demonstrate quick changes between pointing goals, showing development towards closing the **Autonomous Rapid Retargeting** technical gap (Section 2.3.2.d).

8.2.4 Improved Autonomy for ADCS

The previously mentioned contributions have benefits on their own, but they can also be used together to create greater synergistic contributions. One such contribution is an ADCS approach that is more autonomous, general, and flexible across different satellite architectures, scales, and missions. This approach can take into account the spacecraft disturbance environment, actuation limitations, and specific goals to manage the satellite's attitude in a reasonable (though perhaps not provably optimal) way. The work for this contribution is spread across several chapters, as it takes parts from Chapter 4, 6, 5, and 7. These chapters each show part of the method, and some show results of the combination. It draws from all areas of work, including the **Improved Attitude State Estimation** (Section 3.2.1), **Generalized ADCS** (Section 3.2.5), and **Attitude Trajectory Planning** (Section 3.2.4). This contribution includes:

- an ADCS framework that combines a generalized estimator and planner (or other control approaches) in a manner that includes disturbances and considers the ADCS goals;
- methods to create this framework for a variety of satellite architectures with various sensors and actuators of different limits and quality, and a variety of disturbances;
- incorporation of a variety of types of pointing goals, at varying times, with methods to include them in various control laws;
- incorporation of a variety of control laws (with the potential for many more) and specific schedules for using them; and
- software to combine these methods and operate onboard satellites.

The results showing this contribution can be found in Chapter 7, particularly Sections 7.5.4. In both of those results, simulations are shown where (simulated) spacecraft use the aforementioned estimators and planner to control towards the goal, following operations and goals as described in Chapter 5 and live-countering various disturbances. The increased autonomy ties directly to the **Autonomous Rapid Retargeting** technical gap (Section 2.3.2.d).

8.2.5 ADCS Architecture for Weak or Underactuated Satellites

The final contribution is also a "combined" contribution, as described in Section 8.2.4, and is exactly what is described in Section 2.4–an ADCS approach for satellites that are weak or underactuated and in a difficult disturbance environment. This contribution is a direct result of efforts on the **Effective Magnetic Control** technical gap in Section 2.3.2.b. In many cases, this contribution uses the methods laid out in this dissertation (across Chapters 4, 6, 5, and 7; tying to all of the areas of work in Section 3.2) to fully address that technical gap. However, due to efforts on making each component generalizable, the final product is also useful for many other satellites and is a step towards closing the **General ADCS Approach** technical gap (Section 2.3.2.a), especially for small satellites. While results can be found across Chapters 4-7, the clearest results, combining all of the parts to show the improved performance of limited systems, can be found in Chapter 7, particularly in Sections 7.5.1 and 7.5.3. This contribution includes:

- the generalized code for different ADCS components that can be applied to various small or underactuated satellites, and
- results showing the possibility of novel actuator combinations, such as constant desaturation or the combination of one reaction wheel and three magnetorquers.

The results showing this contribution can be best found in Chapter 7, particularly Section 7.5.4. In these results, simulations are shown where (simulated) poorly-equipped and underactuated spacecraft still manage degree-level pointing in the presence of disturbances.

8.3 Future Work

This work demonstrates significant advancements in the modeling, estimation, control, and planning of attitude determination and control systems (ADCS). However, numerous opportunities remain to extend and enhance the contributions made here. This section outlines several avenues for future exploration, divided into key areas of software improvements, control methodologies, real-world implementation, and advanced concepts.

8.3.1 Software and Usability Improvements

To improve the usability and robustness of the ADCS framework, several software enhancements are recommended:

1. **Sensor and Actuator Additions**: Add support for additional sensors and actuators, such as star trackers, Earth horizon sensors, and control moment gyroscopes. These would increase the range of use cases and missions for which this framework can be applied. Given the class-based structure of this framework, such modifications should be feasible.

2. **Control Law Optimization**: Streamline the structure of control laws and work with software engineers to improve modularity, efficiency, and robustness of the object-oriented framework. This could include:

    - Defining trajectories at an abstract level (perhaps as its own object), applicable across time steps. This could allow for better control laws to follow the trajectories.
    - Representing control goals and system states as discrete objects or classes rather than dictionaries and vectors.
    - Transitioning more portions of the framework to C++ for improved computational efficiency and integration into flight-ready systems.

3. **Integration and Simplification**: Simplify the overall software structure and enhance its object-oriented nature to enable intuitive interaction with state vectors, control goals, and trajectories. This will make the system more accessible to users with varying levels of experience.

4. **Efficiency Improvements**: Investigate the trade-off between accuracy and computational complexity, including the use of C++ or other compiled languages to accelerate computations. The dynamics-aware filter framework from Chapter 4, for example, is implemented in Python, and it can likely be made much faster using C++ or other compiled languages. Explore adjustments to time steps and resource allocation for improved performance. The more advanced version of ALTRO, named ALTRO-C and using conic-based formulations, might be useful for this [100].

8.3.2 Control Methodology Extensions

Future efforts could explore the development of more general and sophisticated control laws:

1. **Quaternion Set Heuristics**: Extend control laws to operate on quaternion sets rather than single quaternion goals, allowing the system to align vectors in arbitrary orientations within specified constraints. Specifying an exact orientation may be an unnecessary constraint for pointing spacecraft if the goal is based on a single axis. The extra degree of freedom provided by using a set of orientations allows for consideration of additional criteria, such as avoiding a keep-out zone. The author has begun exploring various methods, including minimal control effort, nearest point, or convergence rate estimates using a Lyapunov-style approach.

2. **Mixed Time Horizons**: Investigate trajectory optimization algorithms using mixed time horizons (and time step) to improve planning efficiency and performance. Techniques such as funnel methods and tangent bundle-based analysis could enhance trajectory optimization (and more general control laws).

3. **Convergence and Stability**: Develop more rigorous mathematical proofs, including Lyapunov analyses, to demonstrate stability and convergence properties of the configurable estimator, the generalizable control methods, and the inclusion of bias and other extra state parameters. One example would be a Lyapunov stability analysis of the adaptive control methods, such as the all-in-one torque model, for various disturbance conditions. It would be good to extend these analyses to include specific effects like momentum saturation in reaction wheels. Further Monte Carlo simulations (varying in mass, error, noise, and more) to verify the individual components–particularly the estimator–as well as the combined system are also important.

4. **Disturbance Inclusion in Control**: For attitude control systems as defined in this work, including disturbances as feedforward control have been shown to be helpful, especially in underactuated systems or those with significant disturbances. However, there are risks to such an approach. It would be worthwhile to explore the effects of inaccurate disturbance models and the trade-offs between model accuracy and system performance.

8.3.3 Real-World Performance

Field testing and implementation of the proposed methods will provide valuable insights and help validate the theoretical contributions. Key areas for real-world analysis include:

1. **Timing and Computational Complexity**: Conduct timing analysis and compare the computational complexity of different control and estimation methods across varying trajectory lengths and other hyperparameters.

2. **Hardware-in-the-Loop (HIL) Testing**: Integrate the ADCS framework with physical hardware systems, simulating disturbances and validating control performance. Testing it on an actual satellite, on-orbit, would be the most compelling validation of its performance.

3. **Real Satellite Deployment**: Apply the framework to operational satellite missions, analyzing performance in different orbits, disturbance environments, and under varying mission goals.

4. **Fine-Pointing Applications**: Many of the methods here can achieve attitude estimation and pointing accuracy with subdegree precision, even down to 0.01 degrees. For the approaches to be applicable to fine-pointing at the arcsecond level or lower, an alternative technique may be required. For example, weighting the pointing accuracy in the trajectory planner would require a different objective function (perhaps a more robust multi-objective method, as described in Section 7.6), as using a large weight on the pointing accuracy as currently formatted could cause numerical stability issues.

5. **Model Limitations**: The methods used in this work, particularly the ALTRO-based planner, are sensitive to the models that are used to provide environmental information. As described in Section 7.5.4, there can be issues even from basic orbit propagation. The expected accuracy of the various relevant models, including orbit propagation, celestial dynamics, and the IGRF, should be considered and incorporated into future versions of these methods. There are solutions to deal with expected limitations. For example, if magnetic field divergence is expected, the local deviation could be tracked in the estimator. If orbit propagation pre-planning has limited accuracy, the planned trajectory could be described in such a way that the plan-following control law incorporates the actual orbit positions and goals.

6. **Non-Rigid Bodies**: The dynamics which underpin this work could be extended to non-rigid bodies, perhaps using DRAKE. There has already been some work incorporating non-rigid dynamics into ALTRO for satellite attitude control [89].

7. **Sensitivity to Inaccuracies**: Investigate approaches to model inaccuracies and their potential impact, such as incorrect inertia matrices or wrong estimates for noise variances. If the estimator has enough "slack," small modeling errors or inaccuracies should be handled well. However, wild mis-estimates, or those that interact with other aspects of the dynamics, may have a greater effect. This includes more complex or specific questions such as how solar panel degradation can be modeled as bias and the effect of a spacecraft's own dipole (inherent, or from magnetorquers, and potentially electric activity) on magnetometer bias. Similar questions exist for mis-estimating variances and noise on sensors, actuators, disturbances, and other parameters. All of this also ties into classic estimation questions regarding choosing parameters, covariances, and noise, and how to tune these values for a given application, as well as more modern questions regarding choosing correct error and evolution models, and cases of dependency.

8.3.4 Advanced Concepts

Future research could investigate more innovative and exploratory ideas, such as:

1. **Adaptive Estimation**: Extend adaptability of the dynamics-aware estimator to be more capable of dynamically adjusting its parameters to account for slowly varying

biases, nonlinearities, and drift rates in sensors and actuators. Something like onboard system identification or techniques from adaptive control would be worthwhile, if an accurate and computationally inexpensive approach could be found. Any approach, including the one from this work, should be further investigated to determine how the estimated noise and bias drift rates affect robustness under changing conditions.

2. **AI-Driven Planning**: Consider artificial intelligence or heuristic decision-making systems for selecting between control laws and determining when trajectory planning is necessary or beneficial

3. **Parameter Estimation and Unknowns**: Extend the estimator to incorporate mass and inertia estimation and other unknown parameters critical for accurate dynamics modeling. Consider expectation-maximization techniques to alternate between state estimation and system property estimation. Combining this work's efforts with those in [35] would be a good first step.

4. **Mission-Driven Control Optimization**: Analyze the trade-offs between time steps, trajectory durations, and overlaps to optimize for specific mission objectives.

5. **Safe Mode Integration**: Create robust decision-making frameworks for transitioning between control modes, such as reinitializing the estimator, terminating planning, or entering safe mode under anomalous conditions.

6. **Particle Filters**: Investigate the use of particle filters to capture nonlinearities in the estimation process. These may better handle complex dynamics and multi-modal likelihoods and provide robust alternatives to Kalman filtering.

The proposed future directions reflect the broad potential of this framework to impact both theoretical advancements and practical applications in satellite attitude determination and control. By addressing these areas, the ADCS field can continue to evolve, enabling more capable and autonomous spacecraft.

8.4 Closing

This dissertation explores computational methods to enhance the autonomy, generalizability, and efficacy of satellite attitude determination and control systems (ADCS), with a specific focus on addressing the challenges of underactuation, weak actuators, and dynamic disturbances. Throughout this work, the aim is not only to improve the immediate performance of ADCS but also to present a framework that can adapt to the diverse and evolving needs of modern space missions.

At its core, this research demonstrates that the combination of advanced estimation methods, trajectory planning, and disturbance modeling can fundamentally shift the capabilities of satellite control systems. By fully modeling satellite dynamics and incorporating disturbances directly into control laws, this framework enables satellites to operate effectively even in constrained environments with low-quality sensors or limited actuation. In doing so,

it expands the possibilities for satellite architectures and missions, especially in the growing fields of nanosatellites and mega-constellations.

The development of a generalized disturbance- and dynamics-aware estimator exemplifies the integration of theoretical rigor and practical application. Capable of accurately estimating orientation, angular velocity, biases, and even time-varying disturbances, this estimator provides a robust foundation for control and planning. Similarly, the incorporation of trajectory planning, adapted to real-world implementation constraints, allows for seamless goal-switching and efficient attitude management, paving the way for truly autonomous satellite operations.

In addressing the challenges of magnetic-only control and weak or underactuated systems, this work also highlights the potential for new mission paradigms. From enabling high-precision tasks with magnetically controlled satellites to improving the performance of commercial off-the-shelf components, these advancements not only reduce costs but also democratize access to space technology. This democratization, in turn, fosters innovation across disciplines and industries, fueling the continued growth of space exploration and its applications on Earth. The generalizability of the approaches here also allows for novel architectures, such as the high-accuracy MTQ+1RW framework described in Section 7.5.3.

Beyond its technical achievements, this dissertation embodies a broader vision: the pursuit of systems that are not only efficient and effective but also resilient, mission-agnostic, and adaptable. In the dynamic and often unpredictable realm of space, such systems are key to unlocking new possibilities, from scientific discovery to global connectivity and beyond. It is my hope that this work serves as a foundation upon which others may build, continuing the journey toward a more autonomous and accessible space future.

References

[1] C. Pong, "On-Orbit Performance & Operation of the Attitude & Pointing Control Subsystems on ASTERIA," Logan, UT, Aug. 2018. URL: https://digitalcommons.usu.edu/smallsat/2018/all2018/361.

[2] W. Li and M. Hudson, "Earth's Van Allen Radiation Belts: From Discovery to the Van Allen Probes Era," en, *Journal of Geophysical Research: Space Physics*, vol. 124, no. 11, pp. 8319–8351, 2019, ISSN: 2169-9402. DOI: 10.1029/2018JA025940. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2018JA025940 (Last visited on 10/28/2024).

[3] P. W. Likins, "Effects of Energy Dissipation on the Free Body Motions of Spacecraft," Tech. Rep. JPL-TR-32-860, Jul. 1966, NTRS Author Affiliations: Jet Propulsion Lab., California Inst. of Tech. NTRS Document ID: 19660020833 NTRS Research Center: Legacy CDMS (CDMS). URL: https://ntrs.nasa.gov/citations/19660020833 (Last visited on 10/28/2024).

[4] *JWST Fine Guidance Sensor - JWST User Documentation*, JWST User Documentation (JDox) Website. URL: https://jwst-docs.stsci.edu/jwst-observatory-hardware/jwst-fine-guidance-sensor/#gsc.tab=0 (Last visited on 10/28/2024).

[5] P. W. Fortescue, G. Swinerd, and J. P. W. Stark, *Spacecraft systems engineering*, English. Chichester: Wiley, 2011, OCLC: 1166768290, ISBN: 978-0-470-75012-4.

[6] C. Cappelletti, S. Battistini, and B. Malphrus, *CubeSat Handbook: From Mission Design to Operations*. Sep. 2020, ISBN: 978-0-12-817884-3.

[7] F. L. Markley and J. L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*, en. New York, NY: Springer New York, 2014, ISBN: 978-1-4939-0802-8. DOI: 10.1007/978-1-4939-0802-8. URL: http://link.springer.com/10.1007/978-1-4939-0802-8 (Last visited on 07/05/2021).

[8] F. L. Markley, "Statistical Attitude Determination," en, in *Encyclopedia of Aerospace Engineering*, R. Blockley and W. Shyy, Eds., Chichester, UK: John Wiley & Sons, Ltd, Dec. 2010, eae298, ISBN: 978-0-470-68665-2. DOI: 10.1002/9780470686652.eae298. URL: https://onlinelibrary.wiley.com/doi/10.1002/9780470686652.eae298 (Last visited on 06/28/2023).

[9] J. L. C. Junkins John L., *Optimal Estimation of Dynamic Systems*, 2nd ed. New York: Chapman and Hall/CRC, Nov. 2011, ISBN: 978-0-429-10560-9. DOI: 10.1201/b11154.

[10] L. Markley, "Attitude Determination Using Vector Observations and the Singular Value Decomposition," *J. Astronaut. Sci.*, vol. 38, Nov. 1987.

[11] *Quarternion feedback regulator for spacecraft eigenaxis rotations | Journal of Guidance, Control, and Dynamics*. URL: https://arc-aiaa-org.libproxy.mit.edu/doi/abs/10.2514/3.20418 (Last visited on 03/01/2021).

[12] L. Markley, "Fast Quaternion Attitude Estimation from Two Vector Measurements," *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 25, pp. 411–414, Mar. 2002. DOI: 10.2514/2.4897.

[13] E. Lefferts, F. Markley, and M. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/3.56190. URL: https://doi.org/10.2514/3.56190 (Last visited on 07/05/2021).

[14] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," en, in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, Lake Louise, Alta., Canada: IEEE, 2000, pp. 153–158, ISBN: 978-0-7803-5800-3. DOI: 10.1109/ASSPCC.2000.882463. URL: http://ieeexplore.ieee.org/document/882463/ (Last visited on 10/05/2022).

[15] S. Julier, "The scaled unscented transformation," en, in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, Anchorage, AK, USA: IEEE, 2002, 4555–4559 vol.6, ISBN: 978-0-7803-7298-6. DOI: 10.1109/ACC.2002.1025369. URL: http://ieeexplore.ieee.org/document/1025369/ (Last visited on 06/17/2023).

[16] H. M. T. Menegaz, J. Y. Ishihara, G. A. Borges, and A. N. Vargas, "A Systematization of the Unscented Kalman Filter Theory," en, *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2583–2598, Oct. 2015, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2015.2404511. URL: http://ieeexplore.ieee.org/document/7042740/ (Last visited on 06/27/2023).

[17] J. Crassidis, S. Andrews, L. Markley, and K. Ha, "Contingency designs for attitude determination of TRMM," Jan. 1995.

[18] J. L. Crassidis and F. L. Markley, "Unscented Filtering for Spacecraft Attitude Estimation," en, *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 4, pp. 536–542, Jul. 2003, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.5102. URL: https://arc.aiaa.org/doi/10.2514/2.5102 (Last visited on 05/31/2023).

[19] S. Andrews and S. Bilanow, "Recent Flight Results of the TRMM Kalman Filter," en, in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, California: American Institute of Aeronautics and Astronautics, Aug. 2002, ISBN: 978-1-62410-108-3. DOI: 10.2514/6.2002-5047. URL: https://arc.aiaa.org/doi/10.2514/6.2002-5047 (Last visited on 06/02/2023).

[20] R. Van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, ISSN: 1520-6149, vol. 6, May 2001, 3461–3464 vol.6. DOI: 10.1109/ICASSP.2001.940586.

[21] F. L. Markley, "Attitude Error Representations for Kalman Filtering," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 311–317, 2003, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/2.5048. URL: https://doi.org/10.2514/2.5048 (Last visited on 07/05/2021).

[22] B. Jackson, "Accelerating Numerical Methods for Optimal Control," en, Ph.D. dissertation, Carnegie Mellon University, 2022.

[23] M. Andrle and J. Crassidis, "Geometric Integration of Quaternions," en, in *AIAA/AAS Astrodynamics Specialist Conference*, Minneapolis, Minnesota: American Institute of Aeronautics and Astronautics, Aug. 2012, ISBN: 978-1-62410-182-3. DOI: 10.2514/6.2012-4421. URL: https://arc.aiaa.org/doi/10.2514/6.2012-4421 (Last visited on 10/19/2022).

[24] J. Solà, "Quaternion kinematics for the error-state Kalman filter," *arXiv:1711.02508 [cs]*, Nov. 2017, arXiv: 1711.02508. URL: http://arxiv.org/abs/1711.02508 (Last visited on 08/18/2021).

[25] N. Trawny and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," en, Ph.D. dissertation, University of Minnesota, Mar. 2005.

[26] J. Cemenska, "Sensor Modelling and Kalman Filtering Applied to Satellite Attitude Determination," en,

[27] J. Tuthill, "DESIGN AND SIMULATION OF A NANO-SATELLITE ATTITUDE DETERMINATION SYSTEM," en, Ph.D. dissertation, Naval Postgraduate School, Dec. 2009.

[28] S. Esteban, J. M. Girón-Sierra, Ó. R. Polo, and M. Angulo, "Signal Conditioning for the Kalman Filter: Application to Satellite Attitude Estimation with Magnetometer and Sun Sensors," en, *Sensors*, vol. 16, no. 11, p. 1817, Nov. 2016, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s16111817. URL: https://www.mdpi.com/1424-8220/16/11/1817 (Last visited on 05/22/2023).

[29] H. E. Soken, C. Hacizade, and S.-i. Sakai, "Simultaneous Adaptation of the Process and Measurement Noise Covariances for the UKF Applied to Nanosatellite Attitude Estimation," en, *IFAC Proceedings Volumes*, 19th IFAC World Congress, vol. 47, no. 3, pp. 5921–5926, Jan. 2014, ISSN: 1474-6670. DOI: 10.3182/20140824-6-ZA-1003.00773. URL: https://www.sciencedirect.com/science/article/pii/S1474667016425383 (Last visited on 10/05/2022).

[30] H. E. Soken and C. Hajiyev, "Pico satellite attitude estimation via Robust Unscented Kalman Filter in the presence of measurement faults," en, *ISA Transactions*, vol. 49, no. 3, pp. 249–256, Jul. 2010, ISSN: 0019-0578. DOI: 10.1016/j.isatra.2010.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0019057810000339 (Last visited on 10/05/2022).

[31] H. E. Soken and C. Hajiyev, "Fault Tolerant Attitude Estimation for Pico Satellites Using Robust Adaptive UKF," en, *IFAC Proceedings Volumes*, 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, vol. 45, no. 20, pp. 726–731, Jan. 2012, ISSN: 1474-6670. DOI: 10.3182/20120829-3-MX-2028.00064. URL: https://www.sciencedirect.com/science/article/pii/S1474667016348418 (Last visited on 10/05/2022).

[32] A. Weiss, F. Leve, I. V. Kolmanovsky, and M. Jah, "Reaction Wheel Parameter Identification and Control through Receding Horizon-Based Null Motion Excitation," en, in *Advances in Estimation, Navigation, and Spacecraft Control*, D. Choukroun, Y. Oshman, J. Thienel, and M. Idan, Eds., Berlin, Heidelberg: Springer, 2015, pp. 477–494, ISBN: 978-3-662-44785-7. DOI: 10.1007/978-3-662-44785-7_25.

[33] J. L. Crassidis, "Sigma-Point Kalman Filtering for Integrated GPS and Inertial Navigation," en,

[34] J. W. Chia, M. S. C. Tissera, K. S. Low, S. T. Goh, and Y. T. Xing, "A low complexity Kalman filter for improving MEMS based gyroscope performance," in *2016 IEEE Aerospace Conference*, Mar. 2016, pp. 1–7. DOI: 10.1109/AERO.2016.7500795.

[35] H. Yoon, K. M. Riesing, and K. Cahoy, "Kalman Filtering for Attitude and Parameter Estimation of Nanosatellites Without Gyroscopes," en, *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2272–2288, Sep. 2017, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G002649. URL: https://arc.aiaa.org/doi/10.2514/1.G002649 (Last visited on 12/24/2024).

[36] N. Chaturvedi, D. Bernstein, J. Ahmed, F. Bacconi, and N. McClamroch, "Globally convergent adaptive tracking of angular velocity and inertia identification for a 3-DOF rigid body," en, *IEEE Trans. Contr. Syst. Technol.*, vol. 14, no. 5, pp. 841–853, Sep. 2006, ISSN: 1063-6536. DOI: 10.1109/TCST.2006.876908. URL: http://ieeexplore.ieee.org/document/1668147/ (Last visited on 12/24/2024).

[37] J. Ahmed, V. T. Coppola, and D. S. Bernstein, "Adaptive Asymptotic Tracking of Spacecraft Attitude Motion with Inertia Matrix Identification," en, *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 5, pp. 684–691, Sep. 1998, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4310. URL: https://arc.aiaa.org/doi/10.2514/2.4310 (Last visited on 09/21/2021).

[38] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space mission engineering : the new SMAD*, English. Hawthorne, CA: Microcosm Press : Sold and distributed worldwide by Microcosm Astronautics Books, 2011, ISBN: 978-1-881883-16-6.

[39] M. L. Psiaki, "Magnetic Torquer Attitude Control via Asymptotic Periodic Linear Quadratic Regulation," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 386–394, Mar. 2001, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/2.4723. URL: http://arc.aiaa.org/doi/10.2514/2.4723 (Last visited on 11/03/2021).

[40] J. R. Wertz, "Attitude Hardware," en, in *Spacecraft Attitude Determination and Control*, ser. Astrophysics and Space Science Library, J. R. Wertz, Ed., Dordrecht: Springer Netherlands, 1978, pp. 155–216, ISBN: 978-94-009-9907-7. DOI: 10.1007/978-94-009-9907-7_6. URL: https://doi.org/10.1007/978-94-009-9907-7_6 (Last visited on 03/19/2021).

[41] *Gen 1: CubeWheel Small*, en-US. URL: https://www.cubesatshop.com/product/gen-1-cubewheel-small/ (Last visited on 12/24/2024).

[42] B. Wie and P. M. Barba, "Quaternion feedback for spacecraft large angle maneuvers," en, *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 3, pp. 360–365, May 1985, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.19988. URL: https://arc.aiaa.org/doi/10.2514/3.19988 (Last visited on 01/27/2021).

[43] B. Wie and J. Lu, "Feedback control logic for spacecraft eigenaxis rotations under slew rate and control constraints," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 6, pp. 1372–1379, 1995, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/3.21555. URL: https://doi.org/10.2514/3.21555 (Last visited on 08/24/2021).

[44] B. Wie, H. Weiss, and A. Arapostathis, "Quarternion feedback regulator for spacecraft eigenaxis rotations," en, *Journal of Guidance, Control, and Dynamics*, vol. 12, no. 3, pp. 375–380, May 1989, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.20418. URL: https://arc.aiaa.org/doi/10.2514/3.20418 (Last visited on 01/27/2021).

[45] R. Kristiansen and P. Nicklasson, "Satellite attitude control by quaternion-based backstepping," in *Proceedings of the 2005, American Control Conference, 2005.*, ISSN: 2378-5861, Jun. 2005, 907–912 vol. 2. DOI: 10.1109/ACC.2005.1470075.

[46] Q. Hu, B. Chi, and M. R. Akella, "Reduced Attitude Control for Boresight Alignment With Dynamic Pointing Constraints," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 6, pp. 2942–2952, Dec. 2019, Conference Name: IEEE/ASME Transactions on Mechatronics, ISSN: 1941-014X. DOI: 10.1109/TMECH.2019.2944431.

[47] M. Ovchinnikov and D. Roldugin, "A survey on active magnetic attitude control algorithms for small satellites," *Progress in Aerospace Sciences*, vol. 109, Jun. 2019. DOI: 10.1016/j.paerosci.2019.05.006.

[48] C. Chasset, R. Noteborn, P. Bodin, R. Larsson, and B. Jakobsson, "3-Axis magnetic control: Flight results of the TANGO satellite in the PRISMA mission," en, *CEAS Space Journal*, vol. 5, no. 1, pp. 1–17, Sep. 2013, ISSN: 1868-2510. DOI: 10.1007/s12567-013-0034-9. URL: https://doi.org/10.1007/s12567-013-0034-9 (Last visited on 11/02/2021).

[49] G. Sechi, G. André, D. Andreis, and M. Saponara, "Magnetic Attitude Control of the Goce Satellite," vol. 606, p. 37, Dec. 2005.

[50] M. Lovera, "Magnetic satellite detumbling: The b-dot algorithm revisited," in *2015 American Control Conference (ACC)*, ISSN: 2378-5861, Jul. 2015, pp. 1867–1872. DOI: 10.1109/ACC.2015.7171005.

[51] R. Wiśniewski and M. Blanke, "Fully magnetic attitude control for spacecraft subject to gravity gradient," en, *Automatica*, vol. 35, no. 7, pp. 1201–1214, Jul. 1999, ISSN: 00051098. DOI: 10.1016/S0005-1098(99)00021-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109899000217 (Last visited on 03/19/2021).

[52] C. Böhm, M. Merk, W. Fichter, and F. Allgöwer, "Spacecraft Rate Damping with Predictive Control Using Magnetic Actuators Only," en, in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds., Berlin, Heidelberg: Springer, 2009, pp. 511–520, ISBN: 978-3-642-01094-1. DOI: 10.1007/978-3-642-01094-1_41. URL: https://doi.org/10.1007/978-3-642-01094-1_41 (Last visited on 11/03/2021).

[53] M. Lovera and A. Astolfi, "Global magnetic attitude control of spacecraft in the presence of gravity gradient," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 3, pp. 796–805, Jul. 2006, Conference Name: IEEE Transactions on Aerospace and Electronic Systems, ISSN: 1557-9603. DOI: 10.1109/TAES.2006.248214.

[54] I. V. Belokonov, I. A. Timbai, and P. N. Nikolaev, "Analysis and Synthesis of Motion of Aerodynamically Stabilized Nanosatellites of the CubeSat Design," en, *Gyroscopy and Navigation*, vol. 9, no. 4, pp. 287–300, Oct. 2018, ISSN: 2075-1109. DOI: 10.1134/S2075108718040028. URL: https://doi.org/10.1134/S2075108718040028 (Last visited on 11/03/2021).

[55] B. Stuurman and K. Kumar, "RyeFemSat: Ryerson University Femtosatellite Design and Testing," in *SpaceOps 2010 Conference*, ser. SpaceOps Conferences, American Institute of Aeronautics and Astronautics, Apr. 2010. DOI: 10.2514/6.2010-2157. URL: http://arc.aiaa.org/doi/10.2514/6.2010-2157 (Last visited on 11/03/2021).

[56] M. A. Frik, "Attitude stability of satellites subjected to gravity gradient and aerodynamic torques," *AIAA Journal*, vol. 8, no. 10, pp. 1780–1785, Oct. 1970, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0001-1452. DOI: 10.2514/3.5990. URL: http://arc.aiaa.org/doi/10.2514/3.5990 (Last visited on 07/05/2021).

[57] M. Y. Ovchinnikov, D. S. Roldugin, S. S. Tkachev, and S. O. Karpenko, "New one-axis one-sensor magnetic attitude control theoretical and in-flight performance," en, *Acta Astronautica*, vol. 105, no. 1, pp. 12–16, Dec. 2014, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2014.08.017. URL: https://www.sciencedirect.com/science/article/pii/S0094576514003245 (Last visited on 11/03/2021).

[58] M. Lovera and A. Astolfi, "Spacecraft attitude control using magnetic actuators," en, *Automatica*, vol. 40, no. 8, pp. 1405–1414, Aug. 2004, ISSN: 00051098. DOI: 10.1016/j.automatica.2004.02.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109804000767 (Last visited on 10/20/2021).

[59] M. Lovera and A. Astolfi, "Global Magnetic Attitude Control of Inertially Pointing Spacecraft," en, *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 5, pp. 1065–1072, Sep. 2005, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.11844. URL: https://arc.aiaa.org/doi/10.2514/1.11844 (Last visited on 11/12/2021).

[60] A. M. Zanchettin, A. Calloni, and M. Lovera, "Robust Magnetic Attitude Control of Satellites," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1259–1268, Aug. 2013, Conference Name: IEEE/ASME Transactions on Mechatronics, ISSN: 1941-014X. DOI: 10.1109/TMECH.2013.2259843.

[61] D. S. Ivanov, M. Y. Ovchinnikov, V. I. Penkov, D. S. Roldugin, D. M. Doronin, and A. V. Ovchinnikov, "Advanced numerical study of the three-axis magnetic attitude control and determination with uncertainties," en, *Acta Astronautica*, vol. 132, pp. 103–110, Mar. 2017, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2016.11.045. URL: https://www.sciencedirect.com/science/article/pii/S009457651630371X (Last visited on 11/01/2021).

[62] R. Wisniewski, *Satellite Attitude Control Using Only Electromagnetic Actuation*, Danish. Aalborg Universitetsforlag, 1997. URL: https://vbn.aau.dk/en/publications/satellite-attitude-control-using-only-electromagnetic-actuation (Last visited on 03/19/2021).

[63] P. Wang and Y. Shtessel, "Satellite attitude control using only magnetorquers," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, ISSN: 0743-1619, vol. 1, Jun. 1998, 222–226 vol.1. DOI: 10.1109/ACC.1998.694663.

[64] S. Janardhanan, M. un Nabi, and P. M. Tiwari, "Attitude control of magnetic actuated spacecraft using super-twisting algorithm with nonlinear sliding surface," in *2012 12th International Workshop on Variable Structure Systems*, ISSN: 2158-3986, Jan. 2012, pp. 46–51. DOI: 10.1109/VSS.2012.6163476.

[65] C. Chasset, S. Berge, P. Bodin, and B. Jakobsson, "3-axis magnetic control with multiple attitude profile capabilities in the PRISMA mission," in *57th International Astronautical Congress*, ser. International Astronautical Congress (IAF), American Institute of Aeronautics and Astronautics, Oct. 2006. DOI: 10.2514/6.IAC-06-C1.2.03. URL: http://arc.aiaa.org/doi/10.2514/6.IAC-06-C1.2.03 (Last visited on 11/03/2021).

[66] M. Ovchinnikov, D. Roldugin, V. Penkov, S. Tkachev, and Y. Mashtakov, "Fully magnetic sliding mode control for acquiring three-axis attitude," en, *Acta Astronautica*, vol. 121, pp. 59–62, Apr. 2016, ISSN: 00945765. DOI: 10.1016/j.actaastro.2015.12.031. URL: https://linkinghub.elsevier.com/retrieve/pii/S009457651530120X (Last visited on 10/20/2021).

[67] Y. Ma, B. Jiang, G. Tao, and Y. Cheng, "Actuator failure compensation and attitude control for rigid satellite by adaptive control using quaternion feedback," en, *Journal of the Franklin Institute*, vol. 351, no. 1, pp. 296–314, Jan. 2014, ISSN: 00160032. DOI: 10.1016/j.jfranklin.2013.08.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S0016003213003190 (Last visited on 10/20/2021).

[68] A. Sofyali and E. M. Jafarov, "Purely magnetic spacecraft attitude control by using classical and modified sliding mode algorithms," in *2012 12th International Workshop on Variable Structure Systems*, ISSN: 2158-3986, Jan. 2012, pp. 117–123. DOI: 10.1109/VSS.2012.6163488.

[69] A. Sofyalı, E. M. Jafarov, and R. Wisniewski, "Robust and global attitude stabilization of magnetically actuated spacecraft through sliding mode," en, *Aerospace Science and Technology*, vol. 76, pp. 91–104, May 2018, ISSN: 1270-9638. DOI: 10.1016/j.ast.2018.01.022. URL: https://www.sciencedirect.com/science/article/pii/S1270963816311427 (Last visited on 10/20/2021).

[70] T. Inamori, K. Otsuki, Y. Sugawara, P. Saisutjarit, and S. Nakasuka, "Three-axis attitude control by two-step rotations using only magnetic torquers in a low Earth orbit near the magnetic equator," en, *Acta Astronautica*, vol. 128, pp. 696–706, Nov. 2016, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2016.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0094576516306348 (Last visited on 11/01/2021).

[71] V. A. Bushenkov, M. Y. Ovchinnikov, and G. V. Smirnov, "Attitude stabilization of a satellite by magnetic coils," en, *Acta Astronautica*, vol. 50, no. 12, pp. 721–728, Jun. 2002, ISSN: 0094-5765. DOI: 10.1016/S0094-5765(02)00011-5. URL: https://www.sciencedirect.com/science/article/pii/S0094576502000115 (Last visited on 11/01/2021).

[72] G. V. Smirnov, "Attitude determination and stabilization of a spherically symmetric rigid body in a magnetic field," *International Journal of Control*, vol. 74, no. 4, pp. 341–347, Jan. 2001, Publisher: Taylor & Francis, ISSN: 0020-7179. DOI: 10.1080/00207170010010560. URL: https://doi.org/10.1080/00207170010010560 (Last visited on 11/01/2021).

[73] Y. W. Jan and J. .-. Tsai, "Active control for initial attitude acquisition using magnetic torquers," en, *Acta Astronautica*, vol. 57, no. 9, pp. 754–759, Nov. 2005, ISSN: 0094-5765. DOI: 10.1016/j.actaastro.2005.03.067. URL: https://www.sciencedirect.com/science/article/pii/S0094576505001657 (Last visited on 11/03/2021).

[74] J. Kim and K. Worrall, "Sun tracking controller for UKube-1 using magnetic torquer only*," en, *IFAC Proceedings Volumes*, 19th IFAC Symposium on Automatic Control in Aerospace, vol. 46, no. 19, pp. 541–546, Jan. 2013, ISSN: 1474-6670. DOI: 10.3182/20130902-5-DE-2040.00126. URL: https://www.sciencedirect.com/science/article/pii/S1474667015363801 (Last visited on 11/03/2021).

[75] S. O. Karpenko, M. Y. Ovchinnikov, D. S. Roldugin, and S. S. Tkachev, "One-axis attitude of arbitrary satellite using magnetorquers only," en, *Cosmic Research*, vol. 51, no. 6, pp. 478–484, Nov. 2013, ISSN: 1608-3075. DOI: 10.1134/S0010952513060087. URL: https://doi.org/10.1134/S0010952513060087 (Last visited on 11/03/2021).

[76] R. Wisniewski, *Three-Axis Satellite Attitude Control Based on Magnetic Torquing - Linear Optimal Approach*. 1996.

[77] R. Wisniewski, "Linear Time-Varying Approach to Satellite Attitude Control Using Only Electromagnetic Actuation," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 4, pp. 640–647, Jul. 2000, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/2.4609. URL: http://arc.aiaa.org/doi/10.2514/2.4609 (Last visited on 11/03/2021).

[78] R. Wisniewski and F. Landis Markley, "Optimal Magnetic Attitude Control," en, *IFAC Proceedings Volumes*, 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July, vol. 32, no. 2, pp. 7991–7996, Jul. 1999, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)57363-2. URL: https://www.sciencedirect.com/science/article/pii/S1474667017573632 (Last visited on 11/03/2021).

[79] H. Yan, I. M. Ross, and K. T. Alfriend, "Pseudospectral Feedback Control for Three-Axis Magnetic Attitude Stabilization in Elliptic Orbits," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1107–1115, Jul. 2007, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/1.26591. URL: http://arc.aiaa.org/doi/10.2514/1.26591 (Last visited on 10/20/2021).

[80] Y. Yang, "An efficient algorithm for periodic Riccati equation with periodically time-varying input matrix," en, *Automatica*, vol. 78, pp. 103–109, Apr. 2017, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2016.12.028. URL: https://www.sciencedirect.com/science/article/pii/S0005109816305362 (Last visited on 10/20/2021).

[81] B. Zhou, "Global stabilization of periodic linear systems by bounded controls with applications to spacecraft magnetic attitude control," en, *Automatica*, vol. 60, pp. 145–154, Oct. 2015, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2015.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0005109815002794 (Last visited on 10/20/2021).

[82] S. Kumar, M. S. Varma, A. D. Rao, and V. K. Agrawal, "H∞ Tracking Control for Magnetically Controlled Nano-satellite," en, *IFAC-PapersOnLine*, 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016, vol. 49, no. 1, pp. 166–172, Jan. 2016, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.03.047. URL: https://www.sciencedirect.com/science/article/pii/S2405896316300477 (Last visited on 10/20/2021).

[83] J. Kulkarni and M. Campbell, "An approach to magnetic torque attitude control of satellites via 'H/sub /spl infin//' control for LTV systems," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, ISSN: 0191-2216, vol. 1, Dec. 2004, 273–277 Vol.1. DOI: 10.1109/CDC.2004.1428642.

[84] J. Liang, R. Fullmer, and Y. Q. Chen, "Time-optimal magnetic attitude control for small spacecraft," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, ISSN: 0191-2216, vol. 1, Dec. 2004, 255–260 Vol.1. DOI: 10.1109/CDC.2004.1428639.

[85] M. Wood and W.-H. Chen, "Regulation of Magnetically Actuated Satellites using Model Predictive Control with Disturbance Modelling," in *2008 IEEE International Conference on Networking, Sensing and Control*, Apr. 2008, pp. 692–697. DOI: 10.1109/ICNSC.2008.4525305.

[86] T. Krogstad, J. Gravdahl, and P. Tondel, "Explicit Model Predictive Control of a Satellite with Magnetic Torquers," in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, ISSN: 2158-9879, Jun. 2005, pp. 491–496. DOI: 10.1109/.2005.1467064.

[87] A. Gatherer and Z. Manchester, "MAGNETORQUER-ONLY ATTITUDE CONTROL OF SMALL SATELLITES USING TRAJECTORY OPTIMIZATION," en, *Proceedings of AAS/AIAA Astrodynamics Specialist Conference*, p. 14, Aug. 2019.

[88] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Nov. 2019, pp. 7674–7679. DOI: 10.1109/IROS40897.2019.8967788.

[89] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning With Attitude," en, *IEEE Robotics and Automation Letters*, pp. 1–1, 2021, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3052431. URL: https://ieeexplore.ieee.org/document/9326337/ (Last visited on 07/05/2021).

[90] L. Blackmore, M. Ono, and B. C. Williams, "Chance-Constrained Optimal Path Planning With Obstacles," en, *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, Dec. 2011, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2011.2161160. URL: http://ieeexplore.ieee.org/document/5970128/ (Last visited on 07/06/2021).

[91] C. Petersen, A. Jaunzemis, M. Baldwin, M. Holzinger, and I. Kolmanovsky, "Model predictive control and Extended Command Governor for improving robustness of relative motion guidance and control," *Advances in the Astronautical Sciences*, vol. 152, pp. 701–718, Jan. 2014.

[92] C. Petersen, M. Baldwin, and I. Kolmanovsky, "Model Predictive Control Guidance with Extended Command Governor Inner-Loop Flight Control for Hypersonic Vehicles," en, in *AIAA Guidance, Navigation, and Control (GNC) Conference*, Boston, MA: American Institute of Aeronautics and Astronautics, Aug. 2013, ISBN: 978-1-62410-224-0. DOI: 10.2514/6.2013-5028. URL: https://arc.aiaa.org/doi/10.2514/6.2013-5028 (Last visited on 07/25/2021).

[93] H. B. Hablani, "Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 6, pp. 759–767, Nov. 1999, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/2.4469. URL: http://arc.aiaa.org/doi/10.2514/2.4469 (Last visited on 08/24/2021).

[94] E. L. de Angelis, F. Giulietti, and G. Avanzini, "Single-Axis Pointing of Underactuated Spacecraft in the Presence of Path Constraints," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 1, pp. 143–147, 2015, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/1.G000121. URL: https://doi.org/10.2514/1.G000121 (Last visited on 08/24/2021).

[95] J. Wen, S. Seereeram, and D. Bayard, "Nonlinear predictive control applied to spacecraft attitude control," in *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, ISSN: 0743-1619, vol. 3, Jun. 1997, 1899–1903 vol.3. DOI: 10.1109/ACC.1997.610915.

[96] R. A. E. Zidek, C. D. Petersen, A. Bemporad, and V. Kolmanovsky, "RECEDING HORIZON DRIFT COUNTERACTION AND ITS APPLICATION TO SPACECRAFT ATTITUDE CONTROL," en, in *27th AAS/AIAA Space flight mechanics meeting*, Feb. 2017, p. 15.

[97] I. Kolmanovsky and C. Petersen, "Underactuated Spacecraft Control with Disturbance Compensation," en, University of Michigan, Ann Arbor, Dept of Aerospace Engineering, Tech. Rep., Aug. 2015, Section: Technical Reports. URL: https://apps.dtic.mil/sti/citations/ADA627451 (Last visited on 07/25/2021).

[98] C. D. Petersen, F. Leve, and I. Kolmanovsky, "Model Predictive Control of an Underactuated Spacecraft with Two Reaction Wheels," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 320–332, 2017, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/1.G000320. URL: https://doi.org/10.2514/1.G000320 (Last visited on 07/25/2021).

[99] B. Jackson, *AL_ilqr_tutorial.pdf*, Jun. 2019. URL: https://bjack205.github.io/papers/AL_iLQR_Tutorial.pdf (Last visited on 11/04/2021).

[100] B. E. Jackson, T. Punnoose, D. Neamati, K. Tracy, R. Jitosho, and Z. Manchester, "ALTRO-C: A Fast Solver for Conic Model-Predictive Control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 2577-087X, May 2021, pp. 7357–7364. DOI: 10.1109/ICRA48506.2021.9561438. URL: https://ieeexplore.ieee.org/document/9561438 (Last visited on 08/27/2024).

[101] K. D. Bilimoria and B. Wie, "Time-optimal three-axis reorientation of a rigid spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 3, pp. 446–452, 1993, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/3.21030. URL: https://doi.org/10.2514/3.21030 (Last visited on 08/24/2021).

[102] Y. Yang, "Spacecraft attitude determination and control: Quaternion based method," en, *Annual Reviews in Control*, vol. 36, no. 2, pp. 198–219, Dec. 2012, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2012.09.003. URL: https://www.sciencedirect.com/science/article/pii/S1367578812000387 (Last visited on 08/24/2021).

[103] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, "Rigid-Body Attitude Control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 30–51, Jun. 2011, Conference Name: IEEE Control Systems Magazine, ISSN: 1941-000X. DOI: 10.1109/MCS.2011.940459.

[104] F. Bullo, R. M. Murray, and A. Sarti, "Control on the Sphere and Reduced Attitude Stabilization," en, *IFAC Proceedings Volumes*, 3rd IFAC Symposium on Nonlinear Control Systems Design 1995, Tahoe City, CA, USA, 25-28 June 1995, vol. 28, no. 14, pp. 495–501, Jun. 1995, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)46878-9. URL: https://www.sciencedirect.com/science/article/pii/S1474667017468789 (Last visited on 11/04/2021).

[105] C. M. Pong and D. W. Miller, "Reduced-Attitude Boresight Guidance and Control on Spacecraft for Pointing, Tracking, and Searching," en, *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 6, pp. 1027–1035, Jun. 2015, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G000264. URL: https://arc.aiaa.org/doi/10.2514/1.G000264 (Last visited on 08/18/2021).

[106] J. Markdahl, J. Hoppe, L. Wang, and X. Hu, "A geodesic feedback law to decouple the full and reduced attitude," en, *Systems & Control Letters*, vol. 102, pp. 32–41, Apr. 2017, ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2017.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0167691117300099 (Last visited on 08/24/2021).

[107] M. Campbell and T. Tran, *BeaverCube CDR CAD*, Dec. 2019. (Last visited on 11/24/2021).

[108] R. M. Fitzgerald and K. L. Cahoy, "Localized In-Situ Density Measurement in Low Earth Orbit via Drag Torque Estimation," *Journal of Spacecraft and Rockets*, vol. 56, no. 5, pp. 1564–1579, Sep. 2019, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0022-4650. DOI: 10.2514/1.A34338. URL: http://arc.aiaa.org/doi/10.2514/1.A34338 (Last visited on 11/04/2021).

[109] J. R. Wertz, "Introduction to Attitude Dynamics and Control," en, in *Spacecraft Attitude Determination and Control*, ser. Astrophysics and Space Science Library, J. R. Wertz, Ed., Dordrecht: Springer Netherlands, 1978, pp. 487–509, ISBN: 978-94-009-9907-7. DOI: 10.1007/978-94-009-9907-7_15. URL: https://doi.org/10.1007/978-94-009-9907-7_15 (Last visited on 03/19/2021).

[110] L. E. Wiggins, "Relative magnitudes of the space-environment torques on a satellite," en, *AIAA Journal*, vol. 2, no. 4, pp. 770–771, Apr. 1964, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.2426. URL: https://arc.aiaa.org/doi/10.2514/3.2426 (Last visited on 12/14/2024).

[111] J. R. Wertz, "Attitude Prediction," en, in *Spacecraft Attitude Determination and Control*, ser. Astrophysics and Space Science Library, J. R. Wertz, Ed., Dordrecht: Springer Netherlands, 1978, pp. 558–587, ISBN: 978-94-009-9907-7. DOI: 10.1007/978-94-009-9907-7_17. URL: https://doi.org/10.1007/978-94-009-9907-7_17 (Last visited on 03/19/2021).

[112] M. Lovera, E. De Marchi, and S. Bittanti, "Periodic attitude control techniques for small satellites with magnetic actuators," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 1, pp. 90–95, Jan. 2002, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/87.974341.

[113] D. Mostaza-Prieto and P. C. E. Roberts, "Methodology to Analyze Attitude Stability of Satellites Subjected to Aerodynamic Torques," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 437–449, Mar. 2016, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/1.G001481. URL: http://arc.aiaa.org/doi/10.2514/1.G001481 (Last visited on 07/05/2021).

[114] C. D. Petersen, F. Leve, M. Flynn, and I. Kolmanovsky, "Recovering Linear Controllability of an Underactuated Spacecraft by Exploiting Solar Radiation Pressure," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 4, pp. 826–837, 2016, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/1.G001446. URL: https://doi.org/10.2514/1.G001446 (Last visited on 07/25/2021).

[115] M. Flynn, F. Leve, C. Petersen, and I. Kolmanovsky, "Linear control of underactuated spacecraft with two reaction wheels made feasible by solar radiation pressure," in *2015 American Control Conference (ACC)*, ISSN: 2378-5861, Jul. 2015, pp. 3193–3198. DOI: 10.1109/ACC.2015.7171824.

[116] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '85, New York, NY, USA: Association for Computing Machinery, Jul. 1985, pp. 245–254, ISBN: 978-0-89791-166-5. DOI: 10.1145/325334.325242. URL: https://doi.org/10.1145/325334.325242 (Last visited on 11/21/2020).

[117] J. S. Dai, "Euler–Rodrigues formula variations, quaternion conjugation and intrinsic connections," en, *Mechanism and Machine Theory*, vol. 92, pp. 144–152, Oct. 2015, ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2015.03.004. URL: http://www.sciencedirect.com/science/article/pii/S0094114X15000415 (Last visited on 11/22/2020).

[118] D. Q. Huynh, "Metrics for 3D Rotations: Comparison and Analysis," en, *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, Oct. 2009, ISSN: 1573-7683. DOI: 10.1007/s10851-009-0161-2. URL: https://doi.org/10.1007/s10851-009-0161-2 (Last visited on 11/21/2020).

[119] D. Yuan, X. Ma, Y. Liu, Z. Shang, and S. Yan, "Statistical Modeling of Random Walk Errors for Triaxial Rate Gyros," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp. 286–296, Feb. 2016, Conference Name: IEEE Transactions on Instrumentation and Measurement, ISSN: 1557-9662. DOI: 10.1109/TIM.2015.2450358. URL: https://ieeexplore.ieee.org/abstract/document/7346479 (Last visited on 12/16/2024).

[120] M. Ghobadi, P. Singla, and E. T. Esfahani, "Robust Attitude Estimation from Uncertain Observations of Inertial Sensors Using Covariance Inflated Multiplicative Extended Kalman Filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 1, pp. 209–217, Jan. 2018, Conference Name: IEEE Transactions on Instrumentation and Measurement, ISSN: 1557-9662. DOI: 10.1109/TIM.2017.2761230.

[121] S. A. E.-H. Hegazy, A. M. Kamel, I. I. Arafa, and Y. Z. Elhalwagy, "MEMS Gyro Noise Estimation and Modeling for Precise Navigation Simulation," in *2020 8th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC)*, Dec. 2020, pp. 84–89. DOI: 10.1109/JAC-ECC51597.2020.9355936.

[122] Q. Lam, N. Stamatakos, C. Woodruff, and S. Ashton, "Gyro Modeling and Estimation of Its Random Noise Sources," vol. 5562, Aug. 2003, ISBN: 978-1-62410-090-1. DOI: 10.2514/6.2003-5562.

[123] S. Ye, Y. Bar-Shalom, P. K. Willett, and A. S. Zaki, "Statistically efficient estimation of noise variances for a Wiener process observed with measurement noise," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXIII*, vol. 13057, SPIE, Jun. 2024, pp. 72–82. DOI: 10.1117/12.3016575. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13057/130570A/Statistically-efficient-estimation-of-noise-variances-for-a-Wiener-process/10.1117/12.3016575.full (Last visited on 12/16/2024).

[124] P. McKeen and K. Cahoy, "Generalized Attitude Estimation for Spacecraft," *Small Satellite Conference*, Aug. 2024. URL: https://digitalcommons.usu.edu/smallsat/2024/all2024/37.

[125] J. L. Crassidis and F. L. Markley, "Sliding mode control using modified Rodrigues parameters," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 6, pp. 1381–1383, 1996, Publisher: American Institute of Aeronautics and Astronautics. DOI: 10.2514/3.21798. URL: https://doi.org/10.2514/3.21798 (Last visited on 08/24/2021).

[126] E. P. Babcock and T. Bretl, "CubeSat Attitude Determination via Kalman Filtering of Magnetometer and Solar Cell Data," en, p. 10,

[127] E. A. Hogan and H. Schaub, "Three-Axis Attitude Control Using Redundant Reaction Wheels with Continuous Momentum Dumping," en, *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 10, pp. 1865–1871, Oct. 2015, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G000812. URL: https://arc.aiaa.org/doi/10.2514/1.G000812 (Last visited on 08/06/2024).

[128] R. Wiśniewski, "Sliding Mode Attitude Control for Magnetic Actuated Satellite," *IFAC Proceedings Volumes*, 14th IFAC Symposium on Automatic Control in Aerospace 1998, Seoul, Korea, 24-28 August 1998, vol. 31, no. 21, pp. 179–184, Aug. 1998, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)41076-7. URL: https://www.sciencedirect.com/science/article/pii/S1474667017410767 (Last visited on 11/03/2021).

[129] R. Tedrake, *Ch. 1 - Fully-actuated vs Underactuated Systems*. URL: https://underactuated.csail.mit.edu/intro.html#section1 (Last visited on 09/20/2024).

[130] P. McKeen and K. Cahoy, "Method for Pointing Spacecraft While Minimizing Change in Orientation," *Journal of Guidance, Control, and Dynamics*, vol. 45, no. 8, pp. 1554–1557, 2022, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0731-5090. DOI: 10.2514/1.G006429. URL: https://doi.org/10.2514/1.G006429 (Last visited on 08/03/2022).

[131] C. E. Ghisi, C. Steiger, M. Romanazzo, and P. P. Emanuelli, "Drag-Free Attitude and Orbit Control System Performance of ESA's GOCE Mission during Low Orbit Operations and De-orbiting," en, in *SpaceOps 2014 Conference*, Pasadena, CA: American Institute of Aeronautics and Astronautics, May 2014, ISBN: 978-1-62410-221-9. DOI: 10.2514/6.2014-1906. URL: https://arc.aiaa.org/doi/10.2514/6.2014-1906 (Last visited on 11/02/2021).

[132] A. Sveier, A. M. Sjøberg, and O. Egeland, "Applied Runge–Kutta–Munthe-Kaas Integration for the Quaternion Kinematics," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 12, pp. 2747–2754, Dec. 2019, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0731-5090. DOI: 10.2514/1.G004578. URL: https://arc.aiaa.org/doi/10.2514/1.G004578 (Last visited on 10/19/2022).