The Location-Scheduled Control Methodology as Applied to Nanosatellites

By

MATTHEW CHARLES SORGENFREI
B.S. (California Polytechnic State University, San Luis Obispo) 2007
M.S. (University of California, Davis) 2009

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Mechanical and Aerospace Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

Sanjay S. Joshi, Chair

Amit K. Sanyal

Nasli S. Gündeş

Committee in Charge
2013

Acknowledgements

This research represents the culmination of a long journey which never would have been possible without the support of countless friends, family, and mentors. I am forever indebted to my PhD advisor Sanjay Joshi, whose tireless support and encouragement over the past five years made my graduate school experience not only successful, but nearly always enjoyable. It was not only Sanjay who provided the mentorship I needed, but numerous professors in multiple departments both at UC Davis and other universities. Amit Sanyal, Nasli Gundes, Mason Peck, Fidelis Eke, Ron Hess, and Bernad Levy–just to name a few–instilled the foundational knowledge that was critical to my development as an engineer. My friend and colleague Laura Jones deserves special mention, as she unwittingly wandered into a complicated and crazy project that represents a major portion of this work. I am also grateful to my research sponsors at NASA Ames Research Center, including Elwood Agasid and John Hines. This body of work would never have come into being if it were not for John's willingness to gamble on an unknown graduate student and Elwood's dogged protection of my time and resources while serving as a graduate student researcher at NASA Ames.

If it takes a village to raise a child, then it likely took most of California to see me through to completion of my doctorate. My parents (all four of them), my brother Mike, my Godfather Donald James, the UC Davis Men's Water Polo team, and the many people who I am lucky enough to call my friends all own a piece of me crossing the finish line. If writing a dissertation is about becoming focused on one specialized topic within your field, surviving a dissertation is about having the proper support network to help you see the forest for the trees. This work was made possible by endless rides to and from the train station, thousands of hours in the pool at Schaal Aquatic Center, more beers bought for me than Sanjay cares to hear about, and the knowledge that I could call any of you at a moment's notice and get the help I need. Thank you for this love and support.

Abstract

The problem of controlling the behavior of a spacecraft in an optimal manner is one that has been studied since the beginning of the space era in the late 1950s. Recently, the complexity of such optimization problems has been increased by the introduction of spacecraft that are comparatively small in size and capable of being reconfigured, either in between missions or on-orbit. While such spacecraft have the potential to greatly expand the variety of missions that can be undertaken, they also increase the basic number of design variables that must be optimized. This dissertation presents a novel approach for the design of spacecraft control systems that optimizes both controller gain parameters and physical attributes of the spacecraft *in parallel*. The central design tool for this new strategy is a genetic algorithm, which applies concepts from evolutionary biology to search a complex design space in an efficient manner. Results are presented for spacecraft of various sizes, and the genetic algorithm design results are compared to a number of more traditional design approaches.

In the first part of this dissertation the genetic algorithm is applied to the problem of tuning the gain parameters of a nonlinear control law. This controller is used within a small spacecraft performing an attitude tracking maneuver, and must compensate for multiple environmental disturbance moments and the imposition of actuator saturation limits. While the control law under study has proven stability properties, no work has yet been done on optimizing the gain parameters for a specific application. The combined complexity of the controller itself and the spacecraft system make gain tuning via traditional approaches very difficult, and as such a genetic algorithm is utilized. The genetic algorithm can search a broad swath of the overall design space, and does so more efficiently than a human engineer applying their intuition in an "informed" trial-and-error approach to the same problem.

With the basic efficacy of the genetic algorithm established, in the next phase of the dissertation a novel controller optimization approach known as location-scheduled control is introduced. Under location-scheduled control, the use of the genetic algorithm is extended to not only optimize the gain parameters of a given control law but also the physical location of the control actuators within the spacecraft. This *dual* optimization of both controller gain parameters and physical properties of the spacecraft yields a catalog of design solutions that can be called upon from mission to mission, which significantly reduces the time required for control system design. The ability to easily relocate the hardware components of a spacecraft control system is enabled by a class of spacecraft known as CubeSats, which will be described in detail.

In the final portion of this dissertation the location-scheduled control methodology is applied to a real-word testbed system and hardware results are compared to those obtained via simulation. This system makes use of a unique property of superconducting physics known as flux-pinned interfaces that allows CubeSat-scale test articles to be easily reconfigured. The location-scheduled control approach is used to simultaneously determine the optimal configuration of the reconfigurable spacecraft system and the appropriate gains for a single-axis reorientation maneuver. It is shown through hardware testing that the genetic algorithm once again yields a combination of system configuration and controller gain values that outperforms those found by a control systems engineer.

Nomenclature

β = Weight factor used in fitness function
C_e = Control effort expended (Nms)
f = Fitness score for a given design solution
J = Spacecraft inertia tensor (kg m^2)
k = Scalar proportional gain for the controller of Chapter 4
k_P = PID proportional gain
k_I = PID integral gain
k_D = PID derivative gain
K = 3×3 proportional gain matrix for the controller of Chapter 3
L = 3×3 derivative gain matrix for the controller of Chapter 3
$\hat{\lambda}$ = Principal axis of initial rotational error
M_{ad} = 3×1 aerodynamic drag torque acting on the spacecraft (Nm)
M_g = 3×1 gravity gradient torque acting on the spacecraft (Nm)
Mut = Mutation probability in genetic algorithm
Num = Number of generations over which genetic algorithm is executed
Pop = Genetic algorithm design population size
P = 3×3 derivative gain matrix for the controller of Chapter 4
Q = 3×3 error in spacecraft attitude
R = 3×3 rotation matrix describing spacecraft attitude
R_d = 3×3 desired spacecraft rotation matrix
T_s = Settling time of spacecraft (s)
$\bar{\sigma}$ = Modified Rodrigues parameter representation of spacecraft rotational error
τ = 3×1 commanded control torque (Nm)
τ_{us} = 3×1 unsaturated control torque generated by the controller of Chapter 4 (Nm)
τ_{max} = 3×1 maximum control torque that can be supplied by the system actuators (Nm)
τ_f = Friction torque acting on spacecraft testbed (Nm)
θ = Angular position of spacecraft testbed (rad)
θ_d = Desired angular position of spacecraft testbed (rad)
θ_f = Final angular position of spacecraft testbed (rad)
Ω_0 = Spacecraft orbital rate (rad/s)
ω = Spacecraft angular velocity (rad/s)
ω_0 = Inital spacecraft angular velocity (rad/s)
ω_d = Desired spacecraft angular velocity (rad/s)
ω_e = Error in spacecraft angular velocity (rad/s)
Xover = Crossover probability in the genetic algorithm
ζ = 3×1 exponential mapping of the attitude error Q
ζ_{ref} = 3×1 reference error measure

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Approach . . . 2

1.3 The Small Spacecraft Movement . . . 3

1.4 The Location-Scheduled Control Approach . . . 5

1.5 Spacecraft Dynamics and Control . . . 6

1.5.1 Relevant Reference Frames . . . 7

1.5.2 Spacecraft Attitude Dynamics Model . . . 8

1.5.3 Spacecraft Control Strategies . . . 11

1.6 Sensitivity Issues . . . 12

2 Genetic Algorithms for Control System Design **14**

2.1 Gain Tuning Approaches . . . 14

2.2 The Genetic Algorithm . . . 16

2.2.1 Algorithm Overview . . . 16

2.2.2 The Chromosome Structure . . . 17

2.2.3 Evolving the Design Population . . . 18

2.3 The Fitness Function . . . 19

2.4 The Death Penalty . . . 21

3 Control Law Gain Optimization **22**

3.1 Overview . . . 22

3.2 The Hawai'iSat Spacecraft . . . 23

3.3 Candidate Control Law . . . 24

3.4 GA Implementation . . . 26

3.4.1 Fitness Function Design . . . 27

3.5 Numerical Simulation . . . 29

3.5.1 Simulation Description . . . 29

3.5.2 Simulation Results . . . 31

3.6 GA Performance Assessment . . . 36

3.7 Sensitivity Analysis . . . 37

4 The Location-Scheduled Control Methodology **40**

4.1 Nanosatellites for Scientific Research . . . 40

4.2 Location-Scheduled Control . . . 41

4.3 Detumble Maneuver Optimization . . . 44

4.3.1 Fitness Function Selection . . . 44

4.3.2 Control Law Selection . . . 45

4.3.3 Problem Formulation 47
4.4 Simulation Results 49
4.5 Initial Assessment of the LSC Methodology 52
4.6 Sensitivity Analysis 54

5 Reconfigurable Spacecraft Testbed **56**
5.1 Flux-Pinned Interfaces for Reconfigurable Spacecraft 58
5.1.1 The Physics of Magnetic Flux Pinning 58
5.1.2 Applications of FPIs to Spacecraft 59
5.1.3 Experiment Design with FPIs 59
5.2 The Reconfigurable Spacecraft Testbed 60
5.2.1 Hardware Description 60
5.2.2 CubeSat Module Payloads 62
5.2.3 Experiment Procedure 64
5.3 Application of Location-Scheduled Control 65
5.4 Model Development and Controller Implementation 68
5.4.1 System Model 68
5.4.2 Data Processing and Controller Implementation 71
5.5 Reorientation Maneuver Optimization Results 73
5.5.1 Collected Data Overview 73
5.5.2 Comparison Between Simulated and Hardware Results 74
5.5.3 Comparison of Gain Tuning Methods 77
5.6 Assessment 80

6 Conclusion **82**
6.1 The LSC Approach 82
6.2 GAs for Design Optimization 83
6.3 Future Work 84
6.4 Concluding Remarks 86

Bibliography **88**

List of Figures

1.1 The P-POD . . . 4
1.2 GeneSat-1 . . . 4
1.3 MIST . . . 5
1.4 6U Spacecraft . . . 6
1.5 Reference Frames . . . 8
1.6 Body-Fixed Frame . . . 8

2.1 GA Flow Diagram . . . 17
2.2 Chromosome Structure . . . 18
2.3 Crossover Operator . . . 19

3.1 Hawai'iSat . . . 24
3.2 Chromosome Structure for Chapter 3 . . . 27
3.3 GA Time Response, $\beta = 1$ . . . 32
3.4 GA Time Reponse, $\beta = 1000$ . . . 34
3.5 k_1 - k_2 Sensivity Plot . . . 35
3.6 l_1 - l_2 Sensivity Plot . . . 36
3.7 Inertia Sensitivity Plot . . . 38

4.1 The 6U Spacecraft Under Study . . . 42
4.2 The LSC Design Process . . . 43
4.3 Chromsome Structure for Chapter 4 . . . 48
4.4 f = -C_e Time Response . . . 50
4.5 f = -C_e Comparison . . . 51
4.6 $f = -(T_s + \beta C_e)$ Time Response . . . 52
4.7 $f = -(T_s + \beta C_e)$ Comparison . . . 53

5.1 Flux-Pinning Example . . . 58
5.2 The Candidate Configurations . . . 60
5.3 The Reorientation Maneuver . . . 61
5.4 The FluxCraft/FloatCube Testbed . . . 61
5.5 The Active Cube . . . 63
5.6 The Passive Cube . . . 63
5.7 Chromosome Structure for Chapter 5 . . . 66
5.8 Simulink Model . . . 70
5.9 FluxCraft Testbed Data Flow . . . 71
5.10 FluxCraft Data Integration Scheme . . . 72
5.11 Example Hardware Time Response . . . 74
5.12 Comparison of Simulated Responses . . . 77

5.13 Comparison of Hardware Responses 78

6.1 FluxCraft Internal View 85

List of Tables

3.1 GA Performance Values, $\beta = 1$ 31
3.2 Gain Values, $\beta = 1$ 32
3.3 GA Performance Values, $\beta = 1000$ 33
3.4 Gain Values, $\beta = 1000$ 33

4.1 Results using f = -C$_e$ 50
4.2 Results using $f = -(T_s + \beta C_e)$ 52

5.1 FPI Gain Values 68
5.2 Simulation Parameters 70
5.3 Simulation and Hardware Results for the GA (1/2) 75
5.4 Simulation and Hardware Results for the GA (2/2) 75
5.5 Settling Time Comparison 78
5.6 Final Settling Angle Comparison 79
5.7 Control Effort Comparison 79
5.8 Fitness Score Comparison 79

Chapter 1

Introduction

1.1 Motivation

This dissertation presents a novel approach to the design of the attitude determination and control subsystem (ADCS) of small spacecraft. Over the past ten years the aerospace community has been profoundly impacted by the introduction of inexepensive, commerically-available components that are smaller and less power-hungry than their predecessors. This commerical-off-the-shelf (COTS) hardware has resulted in smaller spacecraft which can be developed in a shorter period of time, necessitating a new strategy for the development of each subsystem within the spacecraft. In this research a new design approach will be introduced which considers multiple facets of the spacecraft ADCS design problem *in parallel*. This strategy has the potential to greatly reduce the amount of time required to design a spacecraft ADCS, and complements the rapidly emerging field of small, inexpensive spacecraft.

One exciting consequence of the evolution of very small spacecraft is new possibilities for systems that can be easily reconfigured, either on the ground in between missions or in real time on-orbit. One approach to reconfiguration enabled by recent technology advances is to place multiple microprocessors throughout the spacecraft volume, creating discrete modules with a certain measure of intelligence. A spacecraft designer can then interchange these modules from mission to mission, or can introduce additional actuation systems that allow the modules to reconfigure while in space depending on the needs of the mission. Spacecraft reconfiguration drastically increases the number of design options available to an engineer, which in turn increases the complexity of the overall design problem. Furthermore, reconfigurable spacecraft may be applied to a wide range of tasks, from robotic assembly to sparse aperture arrays, placing higher performance demands on the ADCS. This dissertation develops a new design strategy that optimizes both controller gains and the physical location of the control elements within a reconfigurable spacecraft, a methodology which can be applied to a wide range of spacecraft and missions.

1.2 Approach

While the overall design strategy is widely applicable, this dissertation is concerned with the design and optimization of spacecraft operating in low Earth orbit (LEO). This is due in part to the wide range of missions that LEO spacecraft can carry out, and also because new, highly reconfigurable "CubeSats" have been used for LEO missions in the past. For such missions, an engineer may with equal likelyhood be asked to design a control system for a high-precision imaging payload or for a spacecraft which points a biological system to the sun. Both of these applications require active attitude control, and stand to benefit from the application of the location-scheduled control approach introduced in this dissertation. From a practical standpoint, it is generally cheaper and quicker to build spacecraft operating in LEO, and launch access to low earth orbits comes much more regularly than launches to Geostationary orbit or interplanetary trajectories.

The design strategy introduced in this dissertation is developed through three distinct application examples. The first example serves as a foundation for the main contribution of the dissertation, in that it is only considers optimization of controller gain parameters for a spacecraft operating in LEO. Once the baseline gain tuning problem has been solved, the design tool is extended to problems in which both the control parameters and physical characteristics of the system are optimized *in parallel.* This dual optimization is undertaken for a small spacecraft operating in LEO and then is applied to a real-world hardware testbed for reconfigurable spacecraft. While many examples exist for gain tuning in LEO, no work has yet been done on coupling gain tuning with physical parameter optimization, and this serves as the cornerstone for the research presented herein.

The basic problem of optimizing control law parameters for a spacecraft system has been widely studied (see, for example, the review provided in [2]). If the system to be controlled is linear or can be linearized about an operating point, then it is possible to apply classical root locus techniques for pole placement [1] or more advanced linear optimal control techniques [30]. If the system is nonlinear, however, the techniques to tune controller performance become more complicated. Previously-demonstrated approaches include application of physics-based control strategies to a simplified version of the system [31] or nonlinear programming techniques [6], [8]. In contrast, this research proposes a new methodology for optimizing control system parameters using a genetic algorithm (GA). As will be detailed in the next chapter, implementation of a GA has advantages over these other strategies in that no reduction or simplification of the underlying equations of motion is required, a large swath of the design space can be rapidly searched, and the algorithm is less likely to settle on local optima [10, 18]. Moreover, the GA can be readily extended to the *dual* optimization problem of locating ADCS components and tuning gain parameters, which is the central focus of this research. The previously-published methods described above bring great value to gain parameter optimization, but are less applicable to the broader design approach presented in this work.

1.3 The Small Spacecraft Movement

A central motivating factor behind the design concept that is at the heart of this dissertation is the advent of small, inexpensive, and largely modular spacecraft. Of particular interest are spacecraft known as CubeSats or nanosatellites. Widespread development of nanosatellites first began with the creation of the CubeSat standard by Stanford University and California Polytechnic State University, San Luis Obispo, in 1999 [21]. This standard, coupled with the Cal Poly-built Poly Picosatellite Orbital Deployer (P-POD–seen in Figure 1.1), enabled a wide range of universities, researchers, and government organizations from around the world to start developing low-cost spacecraft. To date, nanosatellite missions have ranged from simple image-capture operations while in LEO to testing of complex thruster systems and implementation of astrobiology experiments. The number and complexity of nanosatellites in orbit has steadily increased, however very few of these spacecraft have attempted anything more than the simplest of science missions.

The standard nomenclature employed by the nanosatellite community is to refer to spacecraft sizing in terms of the number of 10 cm cubes of which the spacecraft is comprised. For example, a nanosatellite which measures 10 cm x 10 cm x 10 cm would be referred to as 1U, a nanosatellite

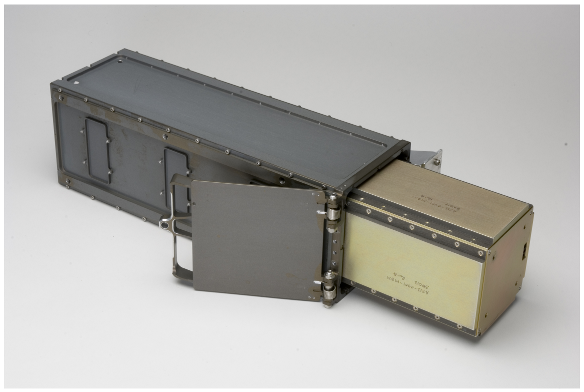

Figure 1.1 – The Poly Picosatellite Orbital Deployer (P-POD) shown holding a spacecraft mass model.

which measures 10 cm x 10 cm x 20 cm would be 2U, and so on. The Mission Design Division at NASA Ames Research Center has successfully launched four different 3U nanosatellite science missions, including the GeneSat-1 spacecraft, seen in Figure 1.2. This spacecraft was passively stabilized (as opposed to actively controlled) because the science payload did not levy stringent pointing requirements on the mission design.

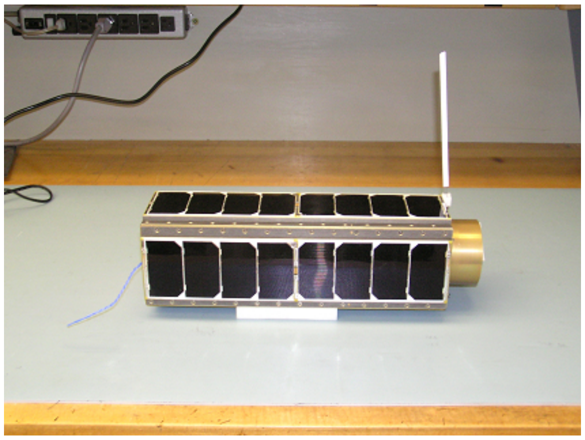

Figure 1.2 – The GeneSat-1 spacecraft, a 3U nanosatellite developed by NASA Ames Research Center.

With recent advances in miniaturization of ADCS sensors and actuators, it is now possible to control CubeSat-class spacecraft to a relatively high level of accuracy. In order to leverage these technology advances, NASA Ames is currently developing the first-ever 6U nanosatellite. Such a spacecraft could house both a full ADCS for active control and a wide range of science payloads, which would greatly broaden the range of missions that can be undertaken by nanosatellites. The 6U form factor will be considered as an application example in Chapter 4 of this dissertation. An early concept for a 6U nanosatellite is presented in Figure 1.3.

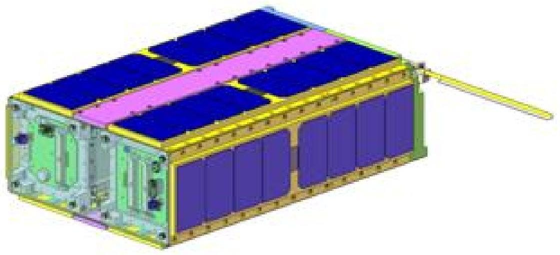

Figure 1.3 – The 6U MIST spacecraft, an actively controlled mission concept.

1.4 The Location-Scheduled Control Approach

While the location-scheduled control methodology will be developed in greater detail later in this dissertation, it is worthwhile to provide a preliminary overview here. The overall goal is to optimize both the gain parameters of the control law being used as well as certain physical properties of the spacecraft system *in parallel*. This concept is enabled in part by the CubeSat standard itself, whereby entire spacecraft are defined in discrete units of volume. If one can leverage this inherent discretization to consider the ADCS as a stand-alone module, then it may be possible to relocate this module within the overall spacecraft volume.

Take as an example the 6U spacecraft seen in Figure 1.4. If one assumes that the ADCS occupies 1U of volume and that the rest of the spacecraft bus operations are contained in another 1U of volume, then there is 4U of volume that can be allocated for the science payload. Depending on the requirements of this payload, it is likely that there is a specific location within the spacecraft where the ADCS could be placed to help optimize the performance of that payload. Under location-scheduled control both the physical location of the ADCS and the gain parameters of the controller being used within the ADCS are tuned in parallel to generate the best-possible overall design for the mission at hand. A genetic algorithm (GA) is used to faciliate this optimization process. The GA is an apt tool because the combined design space of controller gain values and candidate spacecraft layouts can become quite large, and it will be shown that the GA is capable of searching this design space in an efficient manner.

The location-scheduled control concept is intended not only for specific spacecraft but also for entire ranges of missions. If a spacecraft developer wanted to reuse the same suite of ADCS sensors and actuators across multiple missions, then the same 1U module could be tested in multiple

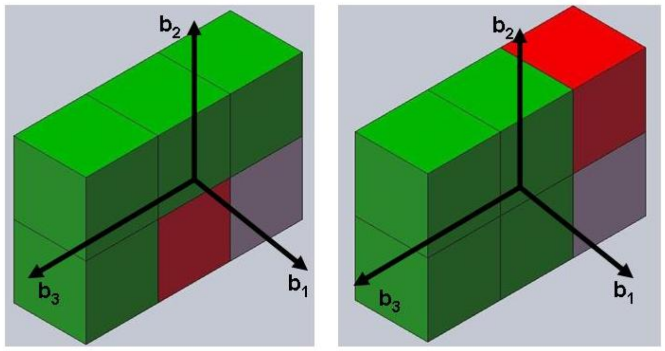

Figure 1.4 – Two conceptual 6U spacecraft layouts, demonstrating how CubeSat modularity enables hardware reconfiguration.

spacecraft layouts *a priori*. Optimizing ADCS location and controller gain values for a specific maneuver across multiple spacecraft layouts gives rise to a *location*-based schedule of gains. This gain schedule can be called upon from mission to mission in order to speed up the ADCS design process, which will in turn greatly reduce the development life cycle for a given mission.

1.5 Spacecraft Dynamics and Control

The fundamental spacecraft building block that is considered in this work is the attitude determination and control subsystem. As its name implies, the ADCS is responsible for determining the orientation (the attitude) of the spacecraft with respect to a given reference frame and then imparting control torques upon the spacecraft to change that attitude as desired. Prior to developing control laws to affect changes in the spacecraft attitude, it is necessary to understand the dynamic equations which govern the spacecraft motion. The same basic set of equations describing the rotational motion of a spacecraft (or some other rigid body) will be used throughout this work, and it is worthwhile to first consider a very general case in which one wishes to control the motion of a spacecraft orbiting the Earth. In order to do so, it is necessary to first define the set of reference frames that will be used to describe the motion of the spacecraft. Throughout this development it is assumed that the spacecraft in question is orbiting the Earth in a circular, equatorial orbit, which simplifies the motion of these reference frames somewhat.

1.5.1 Relevant Reference Frames

For a body in LEO, one should first define an inertial reference frame with respect to which equations of motion can be defined. The inertial reference frame N used in this research has its orgin at the center of mass of the Earth, with the $\hat{n}_2$ unit vector normal to the orbit plane, the $\hat{n}_1$ unit vector pointing towards vernal equinox, and the $\hat{n}_3$ unit vector completing the right-handed system (as depicted in Figure 1.5). It is important to note that this reference frame is not inertial when considering the motion of the Earth about the Sun, however for spacecraft operating in LEO it is a generally accepted practice to make the simplifying assumption that this is in fact an inertial frame [44]. Later work in this dissertation considers the operation of a spacecraft testbed on the surface of the Earth, and for that research this same inertial frame is perfectly acceptable for defining equations of motion.

A typical LEO spacecraft control problem arises from wanting a payload or instrument on a spacecraft to point towards the surface of the Earth. This can be accomplished by requiring a certain body-fixed axis to be in alignment with the nadir vector, which is a vector pointing towards the center of mass of the Earth. To accomplish this control task, it can be convenient to define an intermediate reference frame whose origin is located at the center of mass of the spacecraft and which orbits the Earth at a constant rate. This orbit frame, often called the Local Vertical Local Horizontal (LVLH) frame, is defined such that the $\hat{l}_2$ unit vector is normal to the orbit plane, the $\hat{l}_1$ vector points in the spacecraft velocity direction, and the $\hat{l}_3$ vector points towards zenith. The LVLH frame rotates about the inertial frame at a rate equal to the spacecraft orbital rate Ω_0, which is a function of the orbit altitude [45].

The final reference frame required for a spacecraft control problem is a body-fixed reference frame B. Regardless of the specific application, in this research the origin of the B frame is located at the geometric center of the spacecraft. For applications in orbit, the axes of the B frame are defined such that the body frame and the LVLH frame are aligned when the rotation matrix (to be defined shortly) from the B frame to the LVLH frame equals identity. For applications on the surface of the Earth, the axes of the B frame will be defined in a manner appropriate to the specific hardware under study. A visualization of the $N = \{\hat{n}_1, \hat{n}_2, \hat{n}_3\}$ and $LVLH = \{\hat{l}_1, \hat{l}_2, \hat{l}_3\}$ frames is given in Figure 1.5, while a depiction of the spacecraft B frame (with axes b_1, b_2, and b_3) for the first application example is given in Figure 1.6.

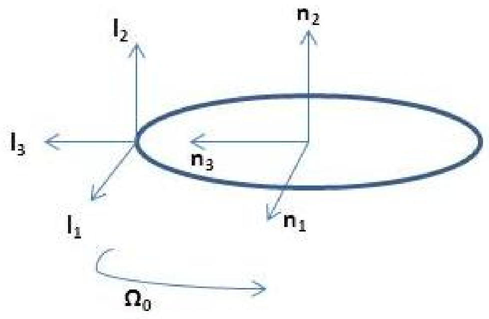

Figure 1.5 – Inertial and LVLH reference frames with the sense of the orbital angular velocity Ω_0 also shown

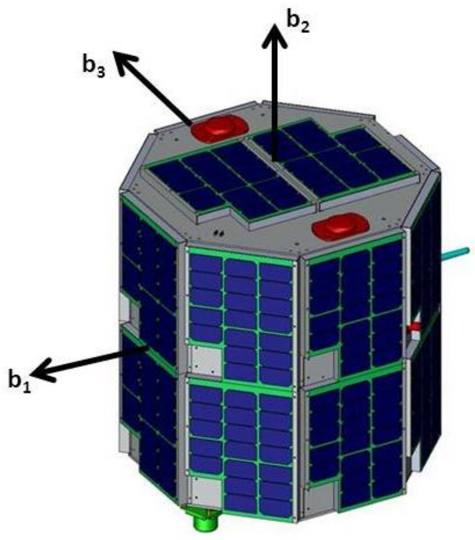

Figure 1.6 – Artists rendition of the Hawai'iSat spacecraft and the body-fixed reference frame B

1.5.2 Spacecraft Attitude Dynamics Model

This dissertation is only concerned with the rotational motion of a spacecraft. While many LEO spacecraft are equipped with sensors and actuators to enable translational motion in orbit, there are a host of applications for which rotational motion control only is adequate. This motion is governed by Euler's equation [13], which considers the rate of change of the spacecraft angular momentum as a function of control and disturbance moments acting on the spacecraft. As outlined in [45], typical sources of disturbance moments in LEO include aerodynamic drag, gravity gradient effects, solar pressure drag, and residual magnetism. In this research only the disturbance moments contributed by aerodynamic drag and the gravity gradient of the spacecraft will be considered. For typical LEO missions the moment contributed by aerodynamic drag is the largest disturbance by at least an order of magnitude [45], so it is a reasonable simplification to only consider these two disturbance sources. In this general case, the equations of motion for the rotation of the spacecraft are derived from Euler's equation [13] as

$$J\dot{\omega} + \omega \times J\omega = \tau + M_{ad} + M_g \tag{1.1}$$

where J is the inertia tensor for the spacecraft in kg m^2, ω is the angular velocity (in rad/s) of the spacecraft defined in the B frame, τ is the control torque, M_{ad} is the aerodynamic drag disturbance, and M_g is the gravity gradient disturbance, all given in Nm. Note that both disturbance sources are dependent upon the orientation of the spacecraft with respect to LVLH reference frame [45], such that

$$M_{ad} = 0.5\rho C_d A V^2(\hat{v} \times \bar{S}_{cp}) \tag{1.2}$$

$$M_g = 3\Omega_0(-{}^B R^L \hat{l}_3 \times -J{}^B R^L \hat{l}_3) \tag{1.3}$$

Where in (1.2) ρ is the atmospheric density at the orbit altitude in kg/m^2, C_d is the drag coefficient of the spacecraft, A is the spacecraft area normal to the velocity direction in m^2, and V is the linear speed of the spacecraft (in m/s). The unit vector $\hat{v}$ points in the velocity direction, while $\bar{S}_{cp}$ is the vector from the center of mass to the center of pressure of the spacecraft (defined in the B frame). The rotation matrix ${}^B R^L$ in (1.3) defines the rotation from the LVLH frame to the B frame, and the unit vector $-\hat{l}_3$ is used because we require the nadir vector direction to define the gravity gradient moment.

Numerous methods exist for defining the attitude of the spacecraft body frame with respect to a given reference frame [13]. When considering two coordinate frames defined in $\mathbb{R}^3$, a straightforward approach is to make use of a rotation matrix, denoted R. Rotation matrices are members of the special orthogonal group $SO(3)$, defined as [5]

$$SO(3) = \{R \in \mathbb{R}^{3x3} : R^T R = I = RR^T, \det(R) = 1\}$$

One drawback to using rotation matrices is that they require nine pieces of information to be carried throughout any calculations (the entries of the 3×3 matrix), and as such many aerospace applications make use of certain reduced-parameter or minimal representations of attitude [46]. Examples of reduced parameter representations are Euler angles, Modified Rodrigues Parameters, or quaternions, which require three, three, and four pieces of information, respectively [40]. However, as outlined in [32], the major drawback of using any minimal representation such as Euler angles or quaternions is that they necessitate a local analysis of the system, since they cannot represent the attitude globally. Euler angles suffer from kinematic singularities which arise from certain trigonometric functions going to zero, and both quaternions and MRPs are subject to a phenomenon known as "unwinding" [32]. The effects of unwinding on total control effort required over a maneuver are detailed in [35], which compares a continuous attitude tracking control law based on quaternions that exhibits unwinding with the stabilizing attitude tracking control scheme that will be discussed in Chapter 3. This comparison shows that the control law that exhibits unwinding requires much more control effort compared to the stabilizing control law, where control effort is defined as a time integral of the Euclidean norm of the control torque over the maneuver duration.

The physical state space of rigid body attitude motion is technically defined as $SO(3) \times \mathbb{R}^3$, or the union of the space of rigid body rotations and translational motion in three dimensions [5]. As described in [35] and elsewhere, the state space $SO(3) \times \mathbb{R}^3$ can be parameterized as $\mathbb{R}^6$, and in this version of the state space there exists certain points that are very close to the desired physical state but which result in the feedback control system accelerating *away* from the equilibrium point, as opposed to converging towards it. The system will ultimately coverge towards the desired state, however this unwinding phenomenon will result in an unecessary loss of control effort. To avoid this problem, this research will largely use rotation matrices as the primary representation of the attitude of a spacecraft. The evolution of a given rotation matrix R in time is governed by the equation

$$\dot{R} = R\omega^{\times} \tag{1.4}$$

where $\omega^{\times}$ is the skew-symmetric operator that carries a 3×1 vector into a 3×3 matrix [5] as

$$\omega^{\times} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

Equations (1.1) and (1.4) fully describe the rotational motion of a spacecraft in LEO. We further parameterize this motion by introducing expressions for the error in attitude and angular velocity, denoted Q and ω_e respectively

$$Q = R_d^T R \tag{1.5}$$

$$\omega_e = \omega - Q^T \omega_d \tag{1.6}$$

where R$_d$ and ω_d are the desired spacecraft attitude and angular velocity, both of which are defined with respect to the inertial frame N. The rotational motion of the spacecraft (Equations (1.1) and (1.4)), parameterized by the rotational and angular velocity error (Equations (1.5) and (1.6)), serve as the foundation for the control problems considered in subsequent chapters.

1.5.3 Spacecraft Control Strategies

A very wide range of literature exists regarding the control of spacecraft in low Earth orbit and beyond. The purpose of this dissertation is not to synthesize new control algorithms for certain LEO applications, but rather to introduce a new design strategy which can be readily adapted to a wide array of existing controllers. Three different rotational maneuvers are studied in the applications included in this work, and three different control algorithms are used to perform these maneuvers. Even with such a diversity of approaches this still represents a small cross-section of the tasks spacecraft might peform, and the purpose of this section is to provide a brief overview of the range of rotational problems encountered on-orbit.

After a spacecraft has been delivered into orbit it separates from the launch vehicle and must be stabilized with respect to the LVLH frame. This stabilization maneuver is referred to as “detumble”, and the control objective is to drive the spacecraft from an arbitrary attitude and high angular velocity to a known attitude and little or no angular velocity. This can be accomplished through various means, but a common approach in LEO spacecraft is to use magnetic control actuators and the so-called B-dot control law [43]. It is not necessary to use magnetic actuators, but as the name implies a magnetic sensor is required to sense the rate of change of the Earth’s magnetic field (the “B-dot”). The detumble operation can also be a helpful control mode when an anomaly occurs and the spacecraft must be placed in a “safe” operating mode where it will await further commands from the ground station.

Once the spacecraft has been stablized with respect to the LVLH frame a number of different operating modes can be undertaken. A spacecraft may wish to keep its attitude fixed with respect to some point in the inertial frame (such as the sun or a collection of stars), and this type of motion would require small but continuous changes to the spacecraft attitude as the LVLH frame moves about the N frame. In order to calibrate sensors and other instruments included in the spacecraft payload it is sometimes necessary to reorient the spacecraft and the bring it back to rest, often referred to as a rest-to-rest reorientation. One challenge associated with this type of maneuver is that it requires the spacecraft to eliminate all of the angular velocity that it has built up during the reorientation, which can often lead to a certain amount of overshoot. Finally, a more complex collection of maneuvers are those which require either a constant or a time-varying spacecraft angular velocity. Such rotations are often required of Earth-observing spacecraft, which may either seek to continuously point a sensor towards the Earth or point and stare at a single location while the spacecraft passes over it.

Many factors contribute to what type of controller is appropriate for a specific maneuver, such as whether a large-angle rotation must be undertaken or whether there are known nonlinearities, including time delays or actuator saturation bounds. Three different cases are presented in this research: a constant angular velocity tracking maneuver which uses a nonlinear controller, a detumble maneuver which uses a different type of saturation-restricted nonlinear controller, and a rest-to-rest reorientation that makes use of a classical Proportional-Integral-Derivative (PID) controller.

1.6 Sensitivity Issues

Ultimately, no real-world system will behave in a truly optimal manner. Inaccuracies in the spacecraft model, uncertainty in the performance of sensors and actuators, and our limited ability to accurately model the LEO environment will all yield behavior that is sub-optimal in some way. The performance of any controller will be altered by changes in system properties, and it is important to understand how these changes affect the design strategy proposed in this work. In each of the following application examples a number of sensitivity studies will be undertaken, in which certain parameters are intentionally varied to understand the impact of these differences. A key strength of the genetic algorithm approach to control system design is that the algorithm can rapidly search a wide portion of the underlying design space, and multiple searches can be performed easily. Thus, even if it is impossible to predict the exact manner in which a model may differ from reality, it is relatively simple and quick for an engineer to test multiple sets of parameters to understand how they may impact optimality. The ability to design and test controller solutions rapidly is critical in the world of spacecraft development, and a genetic algorithm is ideal for these rapid simulations.

Chapter 2

Genetic Algorithms for Control System Design

2.1 Gain Tuning Approaches

A central concern in any control system design problem is determining what gain values should be used for the specific controller being implemented. The broad field of gain tuning has been widely researched, often with the goal of finding the "optimal" combination of gains for the problem under study. How one defines optimal gains is of course dependent on what the controls engineer is seeking to optimize. For simple linear systems, it is possible to apply root locus techniques to place the poles of the system such that time response characteristics (percent overshoot, settling time) are optimized [1]. These strategies are extended in the linear optimal control approach, whereby the gain values are optimized by minimizing a cost functional. This cost functional is typically a weighted combination of system time response and system energy consumption [22], which is another way one might define "optimal". Linear optimal control has been shown to be an effective method for spacecraft controller optimization [30], but it requires linearizing the system equations about an operating point.

Unfortunately, many spacecraft control problems are not linear. Spacecraft operating in LEO must often perform large-angle rotational maneuvers, eliminating the ability to linearize the equations of motion. Additionally, many control actuators used within spacecraft have limits on the amount of torque they can apply, creating saturation bounds on the applied control signal. A number of strategies have been developed for nonlinear gain tuning problems, including application of physics-based control strategies to a simplified version of the system [31] or nonlinear programming techniques [6], [8]. All of these strategies are more complex than their linear counterparts, and often still require that some simplifying assumptions be made.

Since there exist both linear and nonlinear spacecraft control applications, it would be very beneficial to develop a single framework for controller optimization which can be applied to *any* such problem. In this research three different approaches to gain tuning will be studied, all of which are appropriate for either linear or nonlinear controllers. The first method relies upon a human control systems engineer applying their intuition to the problem under study. In this approach, the human engineer undertakes an "informed" trial-and-error search strategy, testing a variety of gain combinations based upon the observed system response. One advantage to this method is that it can be applied to any type of controller being used with any type of physical system, and it leverages the intuition of the engineer in optimizing the gain terms. An obvious disadvantage is that it is somewhat slow, in that one engineer can only test one solution at a time.

One possible way to automate the human trial-and-error approach would be to write computer code which tests every possible solution for a particular control problem automatically. In this method the controls engineer would first determine what the ideal system response is and then write computer code to test all of the possible gain combinations automatically. Such an exhaustive search technique is advantageous in that it ensures that the truly optimal combination of gains is found for the controller being used because *all* gain combinations are tested. However, depending on the complexity of the problem under study it might be infeasible to actually test every possible solution in a reasonable amount of time. A number of different controller design problems are presented in this dissertation, and for some of them it would simply be impossible to actually test all possible gain combinations to find the globally optimal solution.

The last gain tuning method presented in this work makes use of a genetic algorithm. As will be embelished upon in later sections of this chapter a genetic algorithm is a tool for multi-objective design optimization. Genetic algorithms make use of operators from evolutionary biology to search a complex design space with the objective of finding an optimal solution. One advantage of the genetic algorithm approach is that it randomly creates an initial population of designs and then searches the design space using probablistic techniques. This helps the algorithm to avoid becoming stuck on local optima, although it does result in the occasional selection of poor-performing designs within the popluation. As compared to the hand-tuning and exhaustive search approaches, genetic algorithms tend to be able to test a broader cross-section of the design space in a shorter period of time.

In subsequent chapters of this dissertation a comparison will be made between the aforementioned design methods using a variety of control algorithms. The comparison will be based upon certain measures for optimality that are relevant to the spacecraft control problems being studied. Each method has its own advantages, but it will be shown that generally the genetic algorithm outperforms the other methods for both controller gain optimization and the broader problem of location-scheduled control.

2.2 The Genetic Algorithm

2.2.1 Algorithm Overview

A genetic algorithm is a computer-based tool for optimization of complex design problems. Past examples of problems studied using GAs within the aerospace community include launch vehicle design [3], missle system development [27], and flexible structure design [24]. In addition to these applications, genetic algorithms have also been used to tune the gain parameters of an LQR controller for aircraft control [14], as well as those of a nonlinear PID controller used for a robotic manipulator [25] and a neural network-based PID controller [23] for the same application. One unique attribute of GAs as compared to other optimization approaches is that they are heuristic, in that no mathematical proof exists demonstrating their ability to obtain an optimal solution for the problem under study. While lacking a formal proof of optimality, past GA research demonstrates that the algorithm will produce continuously improving design solutions from generation to generation as the population evolves. It will be shown in this dissertation that the algorithm is able to find ideal controller solutions in a timely manner as compared to the other optimization approaches considered.

As outlined in [18], GAs seek optimal solutions to multi-objective design optimization problems by applying a variety of concepts from evolutionary biology. Ultimately, the search process amounts to survival of the fittest, in which ill-performing design solutions are discarded and the desireable traits of well-performing designs become more likely to be selected for use. At the outset of the algorithm an initial population of candidate design solutions is randomly selected from the overall design space. The quality of each member of the population is evaluated with a user-defined fitness function, which allows for each design to be sorted based on its relative fitness. The best-performing solutions are selected to become “parents” for the next generation of design solutions, and the “children” designs are created by applying genetic operators known as crossover and mutation to these parents. The process of evaluating designs and then creating new children design solutions is repeated either until a certain fitness function value has been reached or until a pre-determined number of generations have been tested. Both the number of designs in the population and the number of generations over which the algorithm is executed can be controlled by the user, and multiple sets of these variables will be tested in this dissertation. A visualization of the overall flow of information within a genetic algorithm is given in Figure 2.1.

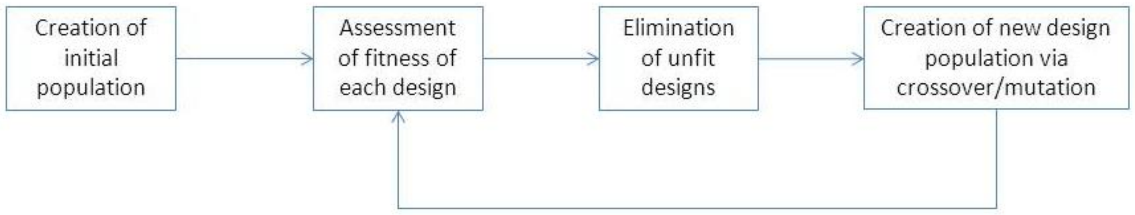
Creation of initial population
Assessment of fitness of each design
Elimination of unfit designs
Creation of new design population via crossover/mutation

Figure 2.1 – A visualization of the overall flow of a genetic algorithm.

2.2.2 The Chromosome Structure

A central component of any GA is the manner in which the members of the design population are represented within the algorithm. Each design is encoded as a string of binary digits referred to as genes, and each string of genes is called a chromosome. The number of binary digits and the meaning of each one is specific to the design problem under consideration. Take as an example the chromosome template used in the next chapter of this dissertation, in which the gain parameters of a complex control law are optimized for use in a small spacecraft. For this problem a total of 36 bits were used, six each for each of the gain parameters to be optimized. As can be seen in Figure 2.2, the first six genes define the value of the gain k_1, the next six correspond to the gain k_2, and so on. In their raw form each member of the design population simply looks like a random string of binary digits, but these strings are an easy format against which to apply the biological operators that drive design evolution. The structure of the chromosome and the number of genes used to represent each design trait is entirely up to the user, and in this work three different gene structures will be studied.

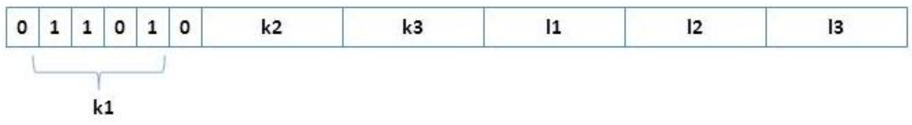

Figure 2.2 – A representative chromosome structure used for a controller gain optimization problem.

2.2.3 Evolving the Design Population

Once the fitness of each design has been evaluated using the fitness function (which will be described in Section 2.3), the algorithm must sort through all of the designs in the population and determine which ones are the most fit to become parents. Numerous methods have been studied for selection of parent designs [10,11], such as roulette wheel selection, random number selection, or tournament selection. In this research tournament selection is utilized, with the tournament size equal to two. Under tournament selection, after the fitness score has been calculated for all designs in the population these designs are selected at random to compete in a two-contestant tournament. Of the two designs, the one with the higher fitness score is deemed the winner, and is selected to become a parent for the next generation. Tournament selection is a quick and efficient strategy for parent selection in terms of computer computation time [10], and it guarantees that the poorest-performing solutions will not become parents for the next design generation. The process of parent selection is repeated until there is a sufficent number of parent pairs to create a new generation of designs. This number is also selectable by the user; one can choose to allow high-performing designs to be copied unchanged into the next generation, or one can simply replace the entire design population at each generation using children created from the best-performing designs. In this work, the entire population is replaced at every generation in order to encourage broad search across the design space.

After all of the parents have been selected for reproduction the actual process of creating children designs can be initiated. The mechanism for variation within the children designs is the application of two probabilistic operators, crossover and mutation. Crossover swaps sections of the parent chromosomes after they have been selected for reproduction, creating two children that will be a new variation of the parents. In contrast, mutation flips the value of a specific bit in a given child design. As seen in Figure 2.3 below, in the simplest case neither crossover nor mutation occurs (a), and the two parent designs are mapped directly to become two children designs. When crossover occurs (b) the parent designs are "cut" at a certain location and the bits from that location forward are swapped in the children designs. Under mutation (c) a specific bit on a child design is switched from a 1 to a 0 or vice versa, regardless of whether or not crossover has occured. The mutation can happen to one, both, or neither design, and the location of the mutation is also randomized. Crossover and mutation each occur with a user-controllable probability, which is what allows the GA to search through a broad swath of the overall design space by being randomly pushed to test different candidate solutions. A detailed study on the ideal combinations of crossover and mutation is provided in [9], and a variety of probability combinations will be presented in subsequent chapters.

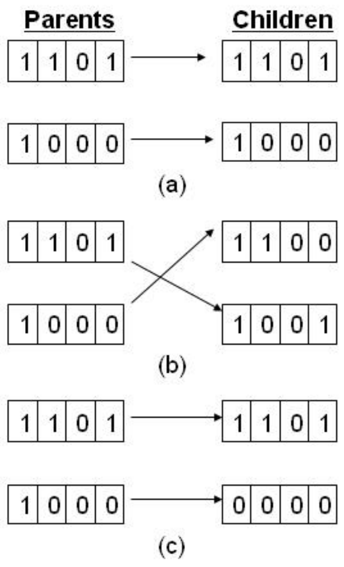

Figure 2.3 – Visualization of some possible ways that the crossover and mutation operators can be implemented.

2.3 The Fitness Function

The primary driver for evolution of the design population within a GA is the fitness function. This function is responsible for evaluating the relative quality of each member of the population, which influences which designs are selected for reproduction. The fitness function ultimately dictates which designs are considered to be "optimal," which greatly impacts the performance of the algorithm. This function is defined by the user, and must reflect the attribute or attributes of the design population which are to be optimized. As such, the design of the fitness function is a critical step in the overall creation of the algorithm; a fitness function which does not represent the needs of the designer will cause the GA to yield designs which are not optimal for the problem under study.

The focus of this dissertation is on optimizing controller gain parameters as well as spacecraft configurations for a variety of applications. For such design problems, one can call upon the canonical performance attributes of control systems to create appropriate fitness functions. System settling time, rise time, percent overshoot, and steady state error are classic metrics for control system tuning [19], all of which may be appropriate for inclusion in a fitness function. If one wishes to include multiple performance metrics in a fitness function, then it might be necessary to create a weighted combination, such as that seen in Equation (2.1).

$$f = -(T_s + \gamma OS) \tag{2.1}$$

In Equation (2.1) T$_s$ represents the settling time of the control maneuver in seconds and OS is the percent overshoot. The weight factor γ has been included to allow the user to adjust the relative contribution of each of these metrics in the fitness score. For example, if one was more concerned with how much a design overshoots its design target, then increasing γ could drive the fitness function to select designs which better satisfy this need. Note that in (2.1) the fitness values are all calculated as being negative. This is equivalent to seeking the minimum possible combination of settling time and percent overshoot, which is often a desireable outcome for control systems designers. Maximizing a negative fitness score is similar in theory to minimizing a positive cost functional, which is the approach typically taken in linear optimal control [22].

A number of different fitness function structures will be presented in subsequent chapters of this work. The overall objective is to find the best-possible controller designs for different rotational maneuvers, and depending on the problem under study one fitness function may prove more advantageous than another. The aforementioned performance metrics are all concerned with the time response of the system (e.g. settling time, rise time) but it is often also necessary to optimize the energy response of the system. There may be a finite amount of energy available to power the actuators of a given system, or there might be a known actuator saturation limit that cannot be exceeded. Minimizing both the energy response and the time response of the system is a classic control systems problem, one that will be visited multiple times in subsequent chapters.

2.4 The Death Penalty

An additional mechanism for guiding the evolution of the GA design population is a “death penalty.” As the name implies, a death penalty eliminates certain members of the design population from possibly becoming parents for the next generation. This is accomplished by assigning an extremely poor fitness score to any design which fails to satisfy a certain design criterion, which will in turn cause it to not be selected when the algorithm goes through the tournament selection process. The probabilistic nature of the crossover and mutation operations can sometimes cause the GA to create designs which do not perform terribly well, and one advantage of a death penalty is that it can weed out certain undesirable designs early in the natural selection process.

In this dissertation a death penalty will be combined with a fitness function to allow the end-user to not only optimize a certain set of parameters within the fitness function, but to also eliminate all designs which do not satisify a given performance metric. Two of the three applications in this dissertation make use of a death penalty that operates on the maximum torque being generated by the control actuators, while the third example utilizes a death penalty acting on settling time. The actuators used within small spacecraft are only capable of providing a comparatively low amount of total control torque, and a death penalty is used in the simulated system to ensure that this maximum torque level is not exceeded. Similarly, the death penalty that acts on settling time discards any designs that fail to satisfy a given settling time metric, allowing the fitness function to instead focus on other design features. It will be shown in the following chapters that like the fitness function itself, the death penalty structure must be fine-tuned somewhat to provide the desired performance.

Chapter 3

Control Law Gain Optimization

3.1 Overview

While the central focus of this dissertation is on a new design methodology known as location-scheduled control, it is worthwhile to begin with a motivating example for the use of GAs as a design tool. To that end, this chapter is concerned with optimizing the gain parameters for a spacecraft undertaking a detumble manuever, as described in Section 1.5.3. This example, which is concerned with tuning gain parameters only, provides a foundation for the more complex design problems undertaken later in this work. As outlined in the Introduction, numerous strategies exist for tuning controller gain parameters, but many of them are not applicable to the general rotation problem considered in this chapter. Barring the use of these various strategies, a less elegant approach is to simply test every possible design combination in simulation, seeking the optimum. While such an exhaustive search approach will ultimately yield a globally optimal solution (in terms of the problem under study), it is infeasible for very large design spaces. The space of candidate gain values associated with the controller under study is quite large, and as such exhaustive search of all candidate solutions is intractable.

This problem is an apt candidate for a GA-based approach, but it would be helpful to have a second design methodology against which to compare the performance of the GA. For this case study (and the other examples presented throughout this dissertation) the GA design strategy is compared to an "informed" trial-and-error search. Absent a particular design tool, a control systems engineer would undertake any controller optimization problem by tuning gain parameters through a trial-and-error search, in which gains are tuned based on the response observed either in simulation or through hardware tests. In a GA the observed response of the system is assessed using a fitness function, and throughout this dissertation the same fitness functions will be used to rate both GA-derived design solutions and those obtained via trial-and-error. A major advantage of the trial-and-error approach is that it allows the engineer to bring their full breadth of knowledge to bear on the problem at hand, as opposed to a GA which only can assess the value of a given design based on the fitness scores. However, trial-and-error tuning can be quite slow because it requires one engineer to test one gain solution one at a time and then make an informed guess about what the next test values should be. Both the time required by each design approach (GA vs. trial-and-error) and the optimality of the selected designs will be considered in the application examples that follow.

3.2 The Hawai'iSat Spacecraft

This chapter uses as an example the Hawai'iSat-1 spacecraft, a small (75 kg) spacecraft that was designed to operate in low Earth Orbit (LEO). A collaboration between the University of Hawai'i and NASA Ames Research Center, Hawai'iSat was to be tasked with observing certain characteristics of the Earth's oceans by means of a hyperspectral imager. The overall spacecraft design matured as far as the Critical Design Review, and in this work the commerically-available sensors and actuators from that design inform the spacecraft simulation. In particular, the spacecraft was equipped with three magnetic torque rods and one reaction wheel for control, as well as a three-axis gyroscope, a digital magnetometer, and analog sun sensors for attitude determination. For the purposes of this research only the attitude control problem will be considered; it is assumed that full state feedback is available.

Using a control algorithm first presented in [32], the spacecraft must track a time-varying attitude and maintain a constant angular velocity. This is a typical maneuver for a spacecraft operating in LEO, in which it can be desireable for the body-fixed B frame to track the LVLH orbit frame, which itself rotates about the inertial N frame. By tracking the attitude of the LVLH frame it is possible for nadir-facing instruments on the spacecraft (such as the Hawai'iSat hyperspectral imager) to continuously observe the surface of the Earth. The observation maneuver is accomplished by rotating the spacecraft body frame at a rate of Ω_0 about a specific body-fixed axis. Recall from the Introduction that Ω_0 is the orbital angular velocity of the spacecraft, which is the rate at which the LVLH frame orbits the N frame. The challenge for the control algorithm is to drive the spacecraft from an arbitrary initial state to tracking the LVLH frame in the presence of actuator saturation constraints and environmental disturbance moments. In particular, the torque rods used to provide the majority of the control torque for the spacecraft are capable of generating approximately 1.2×10^{-3} Nm of torque about each axis, and this saturation limit will be applied to the control algorithm.

The stability proof presented in [32] levies no requirements as to the inertia properties of the spacecraft, and the controller in question has been applied to a wide range of spacecraft sizes [32,34]. In this research we use the inertia tensor for the Hawai'iSat spacecraft as modeled by the design team, with $\mathrm{J} = \mathrm{diag}\begin{pmatrix} 5.69 & 5.69 & 4.26 \end{pmatrix}$ kg m^2. As depicted in Figure 3.1, this is the inertia tensor for a cylindrically shaped spacecraft measuring 0.66 m in diameter and 0.77 m tall.

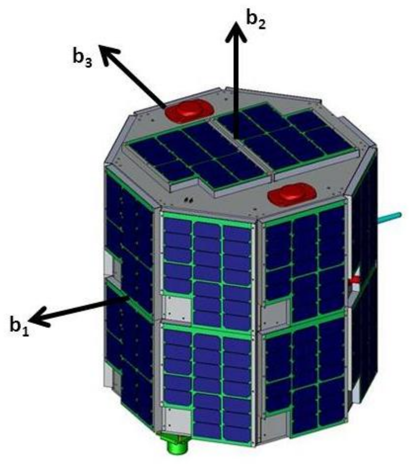

Figure 3.1 – The Hawai'Sat Spacecraft with the axes of the body-fixed reference frame B shown.

3.3 Candidate Control Law

The nonlinear control law that is studied in this chapter was originally developed by Dr. Amit Sanyal and Dr. Nalin Chartuvedi in [32], and was selected for use in the Hawai'iSat-1 mission. While many typical spacecraft applications make use of simpler control algorithms (such as a PID controller), the stability properties of this controller made it an ideal candidate for the large angle rotational maneuvers that were anticipated during the mission. Past research [32, 33] has shown that the controller under study is capable of almost globally stabilizing the attitude dynamics of a spacecraft that is acted upon by environmental disturbances. With stability established, the task for the GA is to determine what gain values result in the best performance of the spacecraft when performing the attitude tracking maneuver. It is important to observe that no accomodation is made for actuator saturation directly in the control algorithm, and as such the genetic algorithm will be used to impose such actuator restrictions. This is accomplished using a death penalty, which will be described in greater detail in the next section. The control law to asymptotically track the desired attitude and angular velocity of the spacecraft is defined as

$$\begin{aligned}\tau = -L\omega_e + JQ^T\dot{\omega}_d + (Q^T\omega_d)^{\times}JQ^T\omega_d + \phi'(\text{trace}(K - KQ))\\ [k_1e_1^{\times}Q^Te_1 + k_2e_2^{\times}Q^Te_2 + k_3e_3^{\times}Q^Te_3] - M_{ad}(R) - M_g(R)\end{aligned} \tag{3.1}$$

Note that the function $\phi(\cdot)$ in (3.1) is a $\mathcal{C}^2$ function used in the stability proof detailed in [32], which is based on Lyapunov methods [41]. While it is beyond the scope of this dissertation to fully replicate the proof of almost global stability for this controller, some interesting properties of Equation (3.1) are discussed here.

Let L and K be positive definite control gain matrices, with the restriction that the diagonal elements of K must be non-equal (e.g. $k_1 \neq k_2 \neq k_3$). As seen in Equation (3.1) the gain matrix L operates on the angular velocity error ω_e and the gain matrix K operates on the angular position error Q. Note also that the output control law is dependent upon the desired angular velocity ω_d and angular acceleration $\dot{\omega}_d$, and that the attitude-dependent aerodynamic drag moment M$_{ad}$ and the gravity gradient torque M$_g$ are included directly in the control algorithm. This implies that the ADCS is capable of resolving the attitude of the spacecraft and feeding that attitude forward to the controller in order to calculate estimated values of the two disturbance moments. This is not an unreasonable assumption, given that typical spacecraft microprocessors are capable of generating state estimates at rates of 4 Hz and higher. Based on the proof presented in [32], it is known that $(Q, \omega_e) = (I, 0)$ is an asymptotically stable equilibrium of the closed loop system consisting of the dynamic and kinematic equations of motion (1.1) and (1.4) and the control law (3.1), with a domain of convergence that is almost global on the state space of attitude motion.

In this research we restrict both K and L to being diagonal positive definite matrices. While the original stability proof requires that K be diagonal [32], no such restriction exists for L. However, it is possible to use a diagonal version of L without loss of generality by a change of body coordiante frames. For example, let R_1 be a rotation matrix from the body frame B to a body frame B_1. Then for a general torque vector applied in each body frame we have the relationship $\tau_1 = R_1\tau$ and $R_1 L\omega_e = R_1 L R_1^T(R_1\omega_e) = L_0(R_1\omega_1)$, where $L_0 = R_1 L R_1^T$ is diagonal. Reduction of the gain matrix L to a diagonal 3×3 matrix results in a total of six gain terms that need to be optimized for the detumble maneuver under consideration. This is by no means a trivial space of candidate designs, however it is somewhat less complex than would be the case for a general positive definite choice of L.

3.4 GA Implementation

Prior to implementing the GA for the gain tuning problem considered here it is necessary to develop a chromosome structure that represents the candidate controller solutions. Referring back to the control law of Equation (3.1), there are six gain terms to be tuned, three each in the gain matrices K and L. Based on preliminary testing of the controller in simulation, the GA will test gain values between 0.0001 and 0.1. The magnitude of these gains might seem relatively low, but it was found that gains in this range gave rise to controller solutions less likely to violate the actuator saturation death penalty. If six bits are used to encode each gain term (resulting in $2^6 = 64$ increments), a resolution of 0.00016 can be achieved by evenly dividing the gain interval of 0.0001 - 0.1 into 64 steps. It is interesting to note that this is a similar gain resolution as was used in [14], where a 2×4 LQR gain matrix was tuned using a GA. However, in that research, the gain terms were allowed to vary between -1 and 1, while Equation (3.1) requires non-negative gains. Using six bits per gain term results in a chromosome that is 36 bits long, where each bit represents a part of the binary representation of a gain value. There are 64 possible gain values for the six different gain terms, resulting in more than 68 *billion* (64^6) design solutions. A visualization of a representative chromosome for the current problem is given in Figure 3.2.

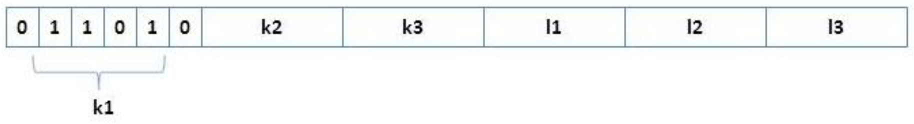

Figure 3.2 – Visualization of the chromosome structure used for the Hawai'iSat-1 gain tuning problem.

3.4.1 Fitness Function Design

The fitness function that will be used to assess the performance of each controller design is similar to the general equation presented in Chapter 2 of this dissertation, and was first implemented in [42]. This function rewards controller designs that result in the minimum possible combination of settling time (T_s) and the amount of control effort expended (C_e), such that

$$f = -(T_s + \beta C_e) \tag{3.2}$$

The desire to balance minimal settling time against minimal control effort is a benchmark problem in spacecraft engineering [47], however for multi-axis rotation problems the standard definition of settling time cannot be applied. In this work we define settling time in terms of the principal axis of rotational error ζ, which can be extracted directly from the attitude error rotation matrix Q using Rodrigues' formula [5]. A system is considered to have settled when the norm of the attitude error vector ζ falls below a certain reference level, e.g.

$$\|\zeta\| < \|\zeta_{ref}\| \tag{3.3}$$

where ζ_{ref} is determined from an attitude error reference matrix Q_{ref}, defined as the composition of three principal rotations about each axis in an amount of 0.1°:

$$Q_{ref} = R_1(0.1^\circ) \cdot R_2(0.1^\circ) \cdot R_3(0.1^\circ) \tag{3.4}$$

Thus, a system is said to have settled when the error in the attitude of the spacecraft falls within a cube of 0.1°. Note that this is equivalent to $\|\zeta_{ref}\| = 0.003$. The total control effort expended during a maneuver is calculated as the $\mathcal{L}_1$ norm of the total torque produced by the actuators, with C_e defined as

$$C_e = \sum_i \|\tau_i\| \cdot \Delta t_i \tag{3.5}$$

The i subscript in (3.5) denotes the ith time step of the simulation and the overall value of C$_e$ is given in units of Nms. A non-zero weight factor β is included in Equation (3.2) to allow for tuning of the relative importance of the two performance metrics, as well as to roughly account for unit-based differences in the magnitude of these metrics. Thus, if a spacecraft designer was more concerned with minimizing the amount of control effort expended, it would be possible to increase β to penalize designs which require a higher amount of control effort. Regardless of the value of β, the most desireable controller design will be that which results in the largest (least negative) fitness value. This results from the smallest possible combination of T$_s$ and C$_e$. Genetic algorithm search results will be presented initially for $\beta = 1$ and for a value of β that has been selected to roughly balance the magnitude of the two performance metrics.

As mentioned previously, the fitness function of (3.2) is combined with a death penalty acting on the maximum control torque provided by the control actuators. This death penalty verifies that the controller solution never requires more than the maximum available control torque that the actuators can produce. For this research the Hawai'iSat spacecraft model was simplified slightly by only considering the magnetic torque rods as available actuators. The selected torque rods are capable of producing a magnetic dipole moment of 25 Amp m^2, which for a circular LEO orbit of 600 km is equivalent to approximately 1.2×10^{-3} Nm of torque about each axis. For each member of the design population the death penalty first checks to make sure that no value greater than this maximum torque is ever required during the MATLAB simulation, and if true allows that specific design to be evaluated by the fitness function. In this way, the genetic algorithm is capable of not only ensuring that the actuator saturation limit is not violated, but also selecting a controller which will yield ideal system peformance by means of the fitness function.

3.5 Numerical Simulation

3.5.1 Simulation Description

For the purposes of this case study the spacecraft is initially rotated 1° about an arbitrary principal axis of rotation and has zero angular velocity in the B frame. This initial state is analagous to that which may be achieved after a spacecraft has completed the "detumble" phase of operations, which was described in the Introduction. In this research the principal axis of the initial rotational error is defined as

$$\hat{\lambda} = \frac{1}{\sqrt{3}} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}^T \tag{3.6}$$

which necessitates rotation about all three body-fixed axes. It is the job of the controller to drive the spacecraft from this initial error state to the desired state. As described earlier, the desired angular velocity of the spacecraft is $\omega_d = \begin{bmatrix} 0 & \Omega_0 & 0 \end{bmatrix}^T$ (rad/s), which results in a time-varying desired rotation matrix R_d. Since a constant angular velocity about the $\hat{b}_2$ axis is equivalent to a time-varying principal rotation about this axis, R_d can be defined as [13]:

$$R_{d_i} = \begin{bmatrix} \cos(\theta_i) & 0 & \sin(\theta_i) \\ 0 & 1 & 0 \\ -\sin(\theta_i) & 0 & \cos(\theta_i) \end{bmatrix} \tag{3.7}$$

where $\theta_i = \Omega_0 \cdot t_i$, the angle of rotation at the *ith* time step. When this desired state has been achieved the spacecraft will be holding its alignment with the LVLH frame as that frame orbits the Earth.

Numerical simulation of the spacecraft system is accomplished using a previously-reported Lie group variational integrator (LGVI) [20]. LGVI-based routines have been shown to have superior numerical properties compared to traditional Runge-Kutta methods, in that the orthogonality of the rotation matrices is preserved throughout the simulation [16]. This integrator, based upon a discretization of the Lagrange-D'Alembert principle, has previously been used for spacecraft rotational motion simulations in [32,34]. The discrete-time equations that govern the rotational motion of the spacecraft in this integration scheme are

$$F_k\mathcal{J} - \mathcal{J}F_k^T = h(J\omega_k)^\times \tag{3.8}$$

$$R_{k+1} = R_kF_k \tag{3.9}$$

$$\tau_k = 0.5(\tau(R_k, \omega_k) + \tau(R_{k+1}, \omega_k)) \tag{3.10}$$

$$J\omega_{k+1} = F_k^TJ\omega_k + h(M_g(R_{k+1}) + M_d(\omega_k) + \tau_k) \tag{3.11}$$

In reviewing Equations (3.8) - (3.11) it might become apparent how they represent a discrete-time analogue of the equations of motion presented in the Introduction. Specifically, Equation (3.10) is the counterpart to (1.1) (where h is the time step of the discrete integrator), (3.9) is the counterpart of (1.4), and (3.8) calculates the discrete variation between time steps, denoted F$_k$. Note that in (3.8) the value $\mathcal{J}$ is a specialized version of the inertia matrix J such that $\mathcal{J} = \frac{1}{2}\text{trace}(J)I - J$. The LGVI scheme is implemented in MATLAB and each candidate controller design is simulated for 500 seconds (spacecraft time).

Recall that two different strategies are used to determine the best possible combination of gains for the control task described above. The first method is to apply the genetic algorithm to the gain tuning problem using the fitness function (3.2) and death penalty as a guide for desired performance. The second method is the informed trial-and-error approach, in which a control systems engineer tunes the elements of the gain matrices based on the system behavior observed in simulation. An engineer undertaking a trial-and-error approach to gain tuning necessarily has access to the same information provided by the fitness function, and fitness function scores will also be reported for the trial-and-error design method.

In the trial-and-error tuning approach a control systems engineer is given three hours to tune the gain variables by hand. Three hours were allotted for this method because that was approximately the amount of time required for the GA to execute a standard trial on an AMD Athlon 5600 desktop PC. For the trial-and-error method, both gain matrices are initialized to $0.05 \cdot I_{3x3}$ (essentially the middle of the admissable gain range) and then numerous simulation iterations are undertaken. After each simulation run, gain variables are either increased or decreased depending on the observed time-response of the spacecraft. The proportional gain matrix K is adjusted first, and then the derivative gain matrix L is adjusted as appropriate. The nonlinear controller under study is analogous to a

PD-type controller, but the overall behavior is certainly not intuitive. The approach of adjusting the K matrix first and then the L matrix is an attempt to tune the proportional response prior to the derivative response, but other strategies may be equally valid. In contrast to this directed gain tuning approach, the GA will randomly change any gain within the controller structure as dictated by the crossover and mutation probability rates. While some poor-performing solutions will be created, this approach generally leads to better-performing solutions as the design population evolves.

3.5.2 Simulation Results

Four different genetic algorithm trials were initially undertaken, the results for which are presented in Tables 3.1 and 3.2. For these preliminary tests the weight factor within the fitness function was held constant at $\beta = 1$. The variables of interest are the design population size (Pop), the number of generations for which the algorithm was run (Num), and the crossover probability (Xover). The values of these variables, given in Table 3.1, follow the basic suggestions provided in [9]. For all of these simulations the mutation rate was held constant at 0.001%, also replicating the work of [9]. The outputs that are recorded for these GA trials are the maximum fitness value achieved (f_{max}), the settling time (T_s) in seconds, the control effort (C_e) in Nms, and the gain values that resulted in these performance attributes (given in Table 3.2).

Table 3.1 – GA performance for four different trials, with the fitness function weight factor set to $\beta = 1$.

Run	**Pop**	**Num**	**Xover**	f_{max}	T_s **(s)**	C_e **(Nms)**
1	40	20	80%	-205.5	205.4	0.12
2	40	40	70%	-187.2	187.1	0.12
3	40	50	70%	-179.5	179.4	0.1
4	50	40	70%	-176.2	176.1	0.1

As can be seen in Table 3.1 the best-performing controller solution was achieved using a population of 50 designs evolved over 40 generations. The maximum fitness score is f = -176.2, corresponding to T_s = 176.1 s and C_e = 0.1 Nms. The gain values that resulted in this performance are $K = \text{diag}\begin{pmatrix} 0.0604 & 0.0017 & 0.0302 \end{pmatrix}$ and $L = \text{diag}\begin{pmatrix} 0.0968 & 0.0968 & 0.0984 \end{pmatrix}$. A component-wise time history of the spacecraft angular velocity and the commanded control torque for this

Table 3.2 – Gain values corresponding to the four GA trials undertaken with $\beta = 1$.

Run	**k$_1$**	**k$_2$**	**k$_3$**	**l$_1$**	**l$_2$**	**l$_3$**
1	0.0445	0.0397	0.0207	0.0715	0.0905	0.0905
2	0.0619	0.0255	0.0175	0.0937	0.0921	0.0715
3	0.0651	0.0112	0.0207	0.0905	0.0984	0.0937
4	0.0604	0.0017	0.0302	0.0968	0.0968	0.0984

best-performing controller design are presented in Figure. 3.3.

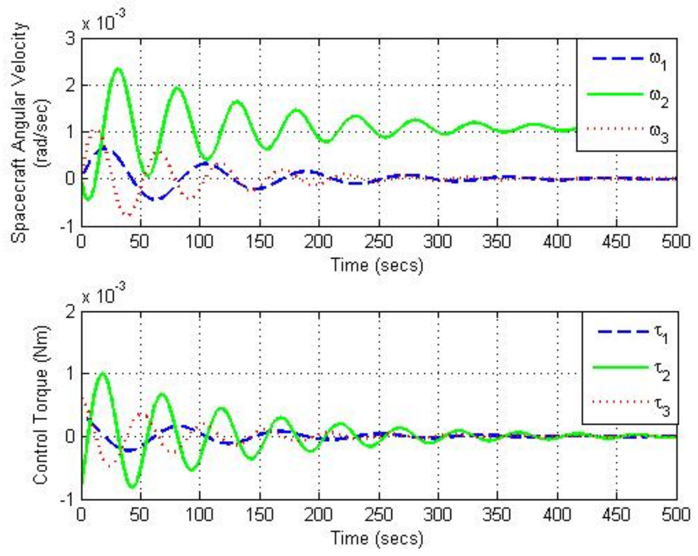

Figure 3.3 – Angular velocity (top plot) and control torque (bottom plot) as a function of time for the best controller design found using the GA with $\beta = 1$.

The efficacy of the GA for gain tuning is compared to that of the informed trial-and-error approach with the fitness function weight factor set to $\beta = 1$. Using trial-and-error search, the best-performing gain values found were K = diag$\begin{pmatrix} 0.0175 & 0.02 & 0.025 \end{pmatrix}$ and L = $0.1 \cdot I_{3\times3}$. For this gain combination a fitness score of f = -186.96 was achieved, resulting from a settling time of T$_s$ = 186.9 s and a control effort of C$_e$ = 0.0626 Nms. It is interesting to note that the hand-tuned gains resulted in the system taking 10 seconds longer to settle, but only required approximately 60% of the control effort needed by the GA-derived solution. The extent to which settling time and control effort must be balanced is up to the engineer concerned with a specific application, but it is worth noting that from the standpoint of overall fitness score the GA did outpeform the hand-tuning method.

Referring back to the best-performing design found using the GA, there is a large difference in the numerical value of settling time and the amount of control effort expended. The difference in magnitude between T_s and C_e (176.1 vs. 0.1) indicate that it might be worthwhile to increase the weight factor β in order to force the fitness function to more evenly balance between the contribution from the two performance metrics during the evolution of the design population. For example, using $\beta = 1000$, a typical value of βC_e (as calculated within the fitness function) might be on the order of 100 Nms as opposed to 0.1 Nms. If the fitness function is trying to minimize the combination of settling time and control effort this should have the effect of causing the GA to favor designs with a longer settling time that expend less control effort. To test this idea, the same four GA trials were re-run with $\beta = 1000$, the results of which are reported in Tables 3.3 and 3.4. Note that in Table 3.3 the value of β is included in the value of βC_e to reflect the balance that the GA is being asked to make via the fitness function. As such, one must divide by 1000 to see the value of C_e associated with a specific design solution.

Table 3.3 – GA performance for four different trials when the fitness function weight factor has been increased to $\beta = 1000$.

Run	**Pop**	**Num**	**Xover**	$\mathbf{f}_{max}$	$\mathbf{T}_s$ **(s)**	$\mathbf{C}_e$ **(Nms)**
1	40	20	80%	-270.38	237.5	32.88
2	40	40	70%	-403.8	268.9	135.1
3	40	50	70%	-289.7	245.9	43.8
4	50	40	70%	-270.36	230.8	39.56

Table 3.4 – Gain values corresponding to the four GA trials undertaken with $\beta = 1000$.

Run	$\mathbf{k}_1$	$\mathbf{k}_2$	$\mathbf{k}_3$	$\mathbf{l}_1$	$\mathbf{l}_2$	$\mathbf{l}_3$
1	0.00049	0.0112	0.0096	0.0952	0.0937	0.0826
2	0.0017	0.035	0.0604	0.0794	0.0715	0.0889
3	0.0001	0.0033	0.0223	0.0952	0.081	0.0921
4	0.0096	0.0175	0.0064	0.0984	0.0889	0.0905

As can be seen in Table 3.3, the design solution achieved from a population of 50 designs evolved over 40 generations is just barely the best overall performer. For this trial, in which f = -270.36, the value of settling time is now only about 6 times greater than that of control effort, as opposed to T_s being nearly 2000 times larger than C_e in the original GA trials. Thus, the weight factor β is having the intended effect, in that the fitness function is now favoring design solutions which give a more even balance of the two performance metrics. It is important to note that it is not appropriate to compare the values of f_{max} for $\beta = 1$ and $\beta = 1000$. By adjusting the weight factor β the basic structure of the fitness function has now changed, and the GA is trying to optimize something different in each test case. Instead, one should compare values of T_s and C_e across the two trials, as these do not change regardless of the structure of the fitness function. The time response of the best-performing design solution with $\beta = 1000$ can be seen in Figure 3.4.

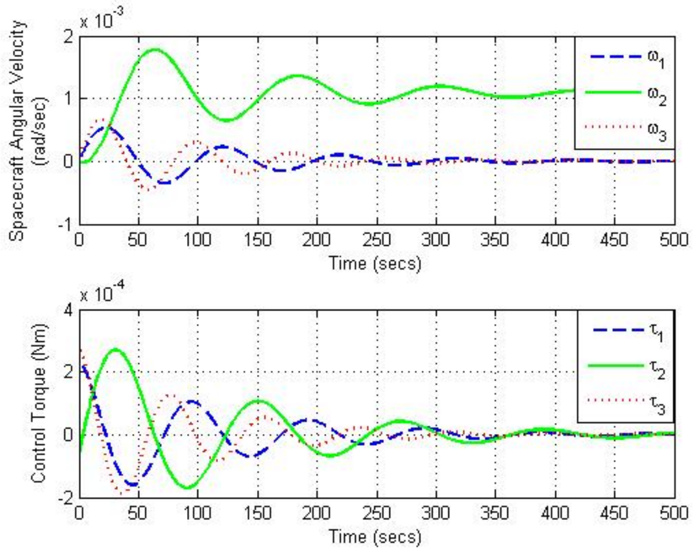

Figure 3.4 – Angular velocity (top plot) and control torque (bottom plot) as a function of time for the best controller design found using the GA with $\beta = 1000$.

The design space under study in this research is obviously quite complex. There are six gains to be optimized, all of which can take on a large range of values. While it is essentially infeasible to exhaustively test every admissable design solution in this problem, it is possible to test cross-sections of the design space to better understand the relationship between certain gain terms and the resulting fitness value. The approach taken was to set $\beta = 1$ and hold all gain values constant except for two, and then to vary these two gains across all admissable values. In the first test, all gain values were held to their optimal values as dictated by the best-peforming GA design (with $\beta = 1$) except the first two entries of the proportional gain matrix K (k_1 and k_2). This results in a controller with gains $K = \text{diag}\begin{pmatrix} k_1 & k_2 & 0.0302 \end{pmatrix}$ and $L = \text{diag}\begin{pmatrix} 0.0968 & 0.0968 & 0.0984 \end{pmatrix}$, where k_1 and k_2 are varied between 0.0001 and 0.1 using all 64 possible values. By calculating the fitness value for each possible gain combination, it is possible to generate a surface in 3D space (seen in Figure 3.5) that shows how the fitness value varies with changing values of k_1 and k_2. The highest fitness score found in this cross-sectional optimization test is f = -165.53, corresponding to T_s = 165.4 s and C_e = 0.13 Nms. This performance was achieved using $k_1 = 0.0449$ and $k_2 = 0.0641$. Looking at Figure 3.5, there is one point ($k_1 = k_2 = 0.0001$) in Fig. 3.5 which has an extremely low fitness value. Recall that one requirement introduced in the stability proof of [32] was that all values of the gain matrix K of in Equation (3.1) be non-equal. Given that this gain combination violates that requirement, the extremely poor fitness value may not be surprising. Logic was written into the genetic algorithm to discard randomly generated values of K that violated the stability requirements of (3.1), thus avoiding any such outliers.

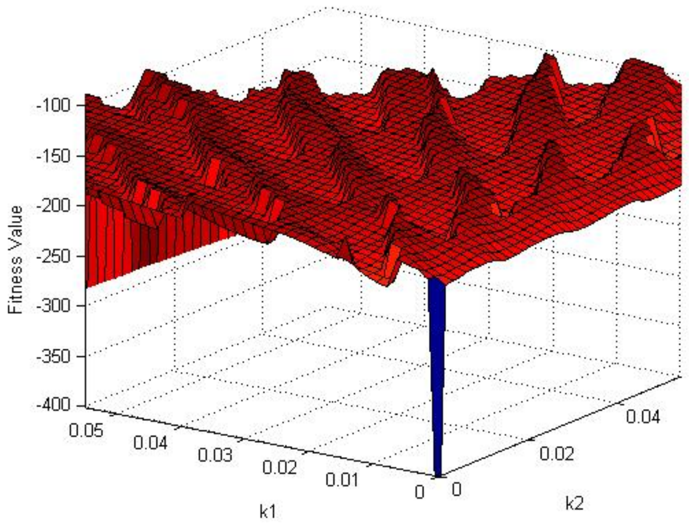

Figure 3.5 – Fitness value as a function of gain combination when all gains are held constant except k_1 and k_2 and these gains are uniformly varied between 0.0001 and 0.1.

A similar test to determine the relationship between the fitness value and certain gain terms was performed for the derivative gain matrix L, in which the gain matrices were set to K = diag$\begin{pmatrix} 0.0604 & 0.0017 & 0.0302 \end{pmatrix}$ and L = diag$\begin{pmatrix} l_1 & l_2 & 0.0984 \end{pmatrix}$ for all possible values of l_1 and l_2. Seen in Figure 3.6, there are no discontinuities in this fitness surface because the only requirement imposed on the gain matrix L is that it be positive definite. Thus, all gain combinations within the gain range being tested are valid solutions for the controller design problem, in that $l_1 \neq 0$ and $l_2 \neq 0$. For this test the maximum fitness value achieved is f = -175.5, corresponding to a settling time of $T_s = 175.4$ s and $C_e = 0.1$ Nms. This performance was accomplished with the two derivative-type gains set to their maximum value, such that $l_1 = l_2 = 0.1$.

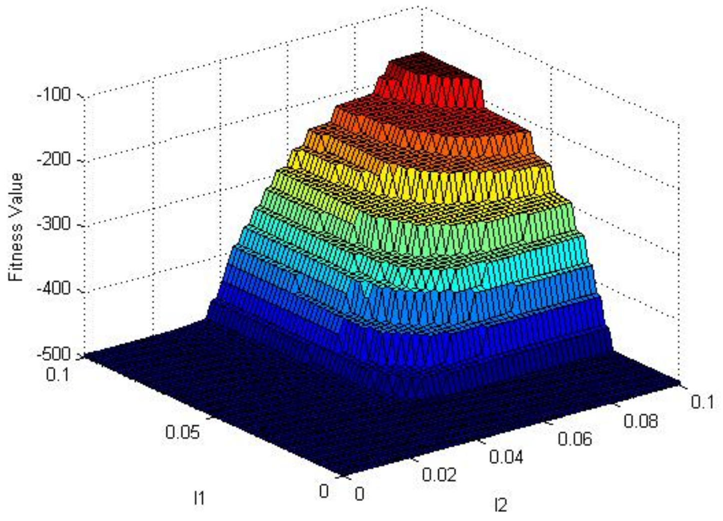

Figure 3.6 – Fitness value as a function of gain combination when all gains are held constant except l_1 and l_2 and these gains are uniformly varied between 0.0001 and 0.1.

3.6 GA Performance Assessment

The results presented in the preceding section indicate that the GA-based gain tuning approach yields comparable system performance as compared to a trial-and-error strategy. For the study in which $\beta = 1$ the GA was able to find a design with a better maximum fitness score, and was also able to search a broader swath of the overall design space. As a point of reference, during the roughly three hour execution time of the GA 2000 design solutions were tested, while during the informed trial-and-error approach approximately 120 designs were tested. Furthermore, during the three hours in which the GA was executing no human involvement was necessary, which means that in a real-world project that engineer could be performing other tasks during the GA design process. It is noteworthy that the hand-tuned solution did yield a design which expended less control effort. The need to minimize either settling time or control effort expended is obviously a choice of the control systems engineer, and depends on the application. For the Hawai'iSat-1 mission it was necessary to slew the spacecraft around to point the hyperspectral imager to the required location in a relatively short amount of time, and thus settling time was typically prioritized higher than control effort expended. Depending on the specific application in question, a reverse set of priorities may in fact hold.

The relative importance of settling time and the amount of control effort expended also inform the selection of the weight factor β. Turning this weight factor up to 1000 from the original value of 1 created a more even balance between settling time and control effort (approximately 6:1), which might be desireable for a certain application. For Hawai'iSat the controller designs resulting from using $\beta = 1$ were appropriate for the specific needs of the tracking maneuver, but the same might not be true for other maneuvers or different spacecraft. The weight factor within the fitness function is a powerfull tool for balancing multiple performance metrics, but the specifics of that balance must be the responsibility of the design engineer.

3.7 Sensitivity Analysis

An important consideration for any controller design scheme is that of sensitivity of the solutions to changes in system parameters. One system model was used to test all of the controller design solutions within the GA population, and it is certain that some system parameters will differ between the spacecraft simulation and the real world hardware. To better understand what impact parameter variations have on the GA-derived controllers two different tests were undertaken. In the first test, the entries of the diagonal inertia tensor J were varied $\pm 2.5\%$, $\pm 5\%$, $\pm 7.5\%$, $\pm 10\%$, and $\pm 12.5\%$. The spaceraft simulation was then re-run for the tracking maneuver using the gains found via the GA for each of the perturbed values of J. A plot of the maneuver settling time and amount of control effort expended for the different changes in J can be seen in Figure 3.7.

As can be seen in Figure 3.7, both T_s and C_e tend to get better for decreasing values of the inertia tensor J while they get worse for increasing values of J. This follows intuition, in that the same amount of control torque is being applied to a system that now has less mass, and thus the spacecraft can be driven more quickly to the desired final state without any "cost" to the amount of control effort expended. It is however interesting to see the large jump in the value of T_s when the inertia perturbation is increased from 10% to 12.5%. Apparently, when this much of a mass increase is seen the selected gain combination is no longer effective for the tracking maneuver.

A similar sensitivity study was undertaken for variations to the drag coefficient of the spacecraft

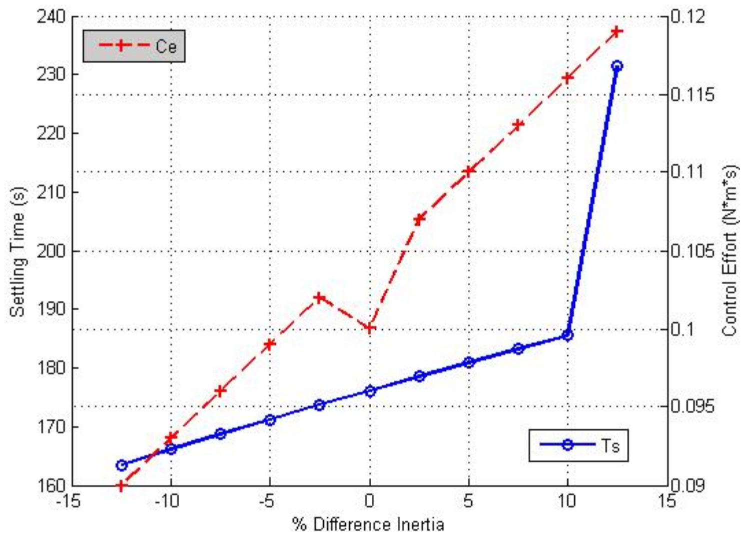

Figure 3.7 – Settling time and amount of control effort expended as a function of percentage difference in the inertia tensor values.

C_d. Referring back to Equation (1.2), the drag coefficient has a strong impact on the value of the aerodynamic disturbance moment M_{ad}, but this is a parameter that is very hard to estimate for LEO spacecraft. Following the suggestion of [45] a value of $C_d = 2$ was used in all of the original spacecraft simulations in this research. To understand what impact variation in C_d has on controller performance, the drag coefficient was varied from 0.8 to 2.6 in increments of 0.2, and the spacecraft simulation was re-run. Once again, the best-performing controller gains from the GA study with $\beta = 1$ were used. Interestingly, varying the drag coefficient C_d had an essentially undetectable impact on the performance metrics T_s and C_e. This implies that the drag moment M_{ad} potentially has a lower impact on the overall performance of the spacecraft than anticipated. In terms of the elements of the fitness function, the aerodynamic drag moment is more "detectable" to the GA through the C_e metric, in that increasing C_d will increase the drag moment (and thus the required control torque) with an essentially linear relationship. In contrast, changes to C_d will more subtly impact T_s, given that this disturbance moment is on the order of 10^{-5} Nm, which leads to a lower overall change in the observed fitness score.

One final aspect of parameter sensitivity that is of interest to a control systems designer is that of sensitivity to changes in the gain values themselves. Referring back to Figures 3.5 and 3.6, it is important to note first of all that there are no gain combinations that result in the system becoming unstable. On the plot of k_1 and k_2 there is one outlier point, but this data point violates the original stability proof outlined in [32] and thus can be discarded. Instead, one can use Figures 3.5 and 3.6 to study how changing gain values will affect the overall fitness of a controller solution. For example, looking at Figure 3.6 one sees that there are plateaus of constant fitness value whereby changes in l_1 or l_2 do not impact the performance of the spacecraft system. This implies a certain measure of robustness to variations in gain values for the system under study.

In contrast, the plot of variations in k_1 and k_2 show ridges of approximately constant fitness value. Thus, varying k_1 or k_2 might result in a decrease in the spacecraft system performance, but none of these ridges are extremely steep. Furthermore, this gain sensitivity is not a feature of the GA-based approach, it is a function of the design problem itself. A human control systems engineer would also encounter sensitivity to changes in the elements of K after selecting the best-performing gain values for implementation. This sensitivity to gain changes arises from the physical system and the controller under study, not the gain tuning methodology. It is very important to remember that the surfaces seen in Figures 3.5 and 3.6 are only one cross-section of a much more complex multi-dimensional design space, but studying these plots does provide some insight into the sensitivity of the controller gains.

Chapter 4

The Location-Scheduled Control Methodology

The results in the preceeding chapter demonstrate that a genetic algorithm can be an effective design tool for the challenging problem of optimizing controller gain parameters. Using these results as a foundation, in this chapter the GA-based design approach will be extended to the broader problem of location-scheduled control, whereby controller gains are optimized *in parallel* with other ADCS design parameters. Once again, the performance of the genetic algorithm will be compared to that of an informed trial-and-error approach, however in this chapter these results will also be compared against the globally optimal solution. By considering a relatively small design space, it is possible to exhaustively test every candidate design in order to determine the true optimum. The GA, trial-and-error, and exhaustive search all will be evaluated for both the amount of time required to carry out a search and for the performance of the selected design.

4.1 Nanosatellites for Scientific Research

A key motivating factor behind the location-scheduled control concept is the growing field of spacecraft that are capable of being reconfigured either in between missions or on-orbit. Modular spacecraft which can be easily reconfigured in between missions have the potential to drastically reduce the development lifecycle, while spacecraft that can reconfigure on-orbit can undertake a wider array of mission operations. Both of these reconfiguration approaches are being considered for spacecraft that adhere to the CubeSat standard [21], due in part to the natural discretization of the spacecraft into 1U modules. This chapter applies the location-scheduled control concept to the dual problem of tuning gain parameters and the physical location of ADCS components, with the objective of enabling efficient reconfiguration of a 6U nanosatellite in between missions.

The 6U form factor was selected for study due to its potential for performing a wide array of science missions. As described in the Introduction, the recent history of nanosatellites largely consists of spacecraft that were either passively stabilized or which simply demonstrated a specific technology concept. While such missions are valuable for advancing the state of the art of nanosatellites, it is of interest to NASA and other organizations to perform detailed scientific research in these small, comparatively inexpensive spacecraft. In contrast to the typical 3U nanosatellite, a 6U spacecraft could likely allocate as much as 4U of volume for a science payload. This volume could be used for a miniature Earth-imaging payload, complex astrobiology experiments, or even astrophysics payloads. These complicated science missions are made possible in part by the basic increase in available volume, and in part by recent advances in miniaturized attitude determination and control hardware.

The reference case that will be considered in this chapter is a 6U nanosatellite in which 4U of volume have been allocated for an ‘L’-shaped science payload. As can be seen in Figure 4.1, it is assumed that 1U of volume is reserved for the spacecraft bus and 1U of volume is reserved for an ADCS module (the darker- and lighter-shaded cubes in the figure). Advances in plug-and-play technologies [15] make it possible for a spacecraft designer to create a self-contained ADCS unit that can be used for various missions without redesign. The plug-and-play controller will be placed in one of two available locations, based on the optimization criteria included within the fitness function. The ability to rapidly redesign the overall spacecraft layout by repositioning the ADCS module is critical to reducing spacecraft development time, and is a key motivator for location-scheduled control.

4.2 Location-Scheduled Control

The objective of location-scheduled control (LSC) is to reduce the amount of time required to develop the ADCS of science missions being undertaken by reconfigurable spacecraft by creating a

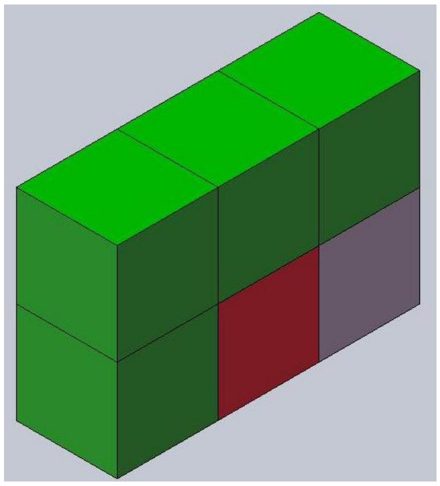

Figure 4.1 – The conceptual 6U spacecraft in which 4U of volume are allocated for the science payload in a 'L' shape.

family of ADCS designs *a priori*. This collection of designs is generated by reusing a predetermined combination of sensors and actuators that are selected to support as wide a range of missions as possible while still adhering to given size, mass, and power budgets. If this hardware combination can be modularized to exist as a discrete unit, then it becomes possible to relocate the unit within the overall spacecraft volume with relative ease. As mentioned in the previous section, advances in spacecraft plug-and-play technologies [15] and standardization of hardware interface components [12] make it a reasonable assumption that the ADCS can be packaged as a discrete unit. When applying LSC, this stand-alone ADCS unit is placed in numerous locations within the spacecraft and gain values are tuned in simulation. This results in a *location*-based schedule of gains that can be referenced as each new science mission becomes ready for implementation. A given allocation of volume for the science payload results in a certain spacecraft layout, and based on the known pointing requirements for that payload an engineer can look up the optimal ADCS configuration from the location-based gain schedule.

In traditional ADCS design the science requirements of a specific mission levy pointing requirements onto the ADCS, which informs sensor and actuator selection. Once the ADCS hardware has been chosen, an appropriate control algorithm is selected and tested. In contrast, the location-scheduled approach to ADCS design can be thought of as a two phase process (depicted in Figure 4.2). In the first phase, a general purpose ADCS unit is designed by selecting a suite of sensors and actuators that would be appropriate for a class of science missions. A number of generic science payload layouts are then considered, and the controller gains/ADCS location are tuned based on the unique dynamics of each layout and anticipated performance requirements. In the second phase of the process, one of the previously tested spacecraft configurations is selected based on the needs of the current science mission. The performance of this configuration is verified, and any necessary gain tuning is undertaken. As long as the catalogue of spacecraft configurations satisfies the needs of the science payloads, the developer of the reconfigurable spacecraft need not return to the first phase of the design process.

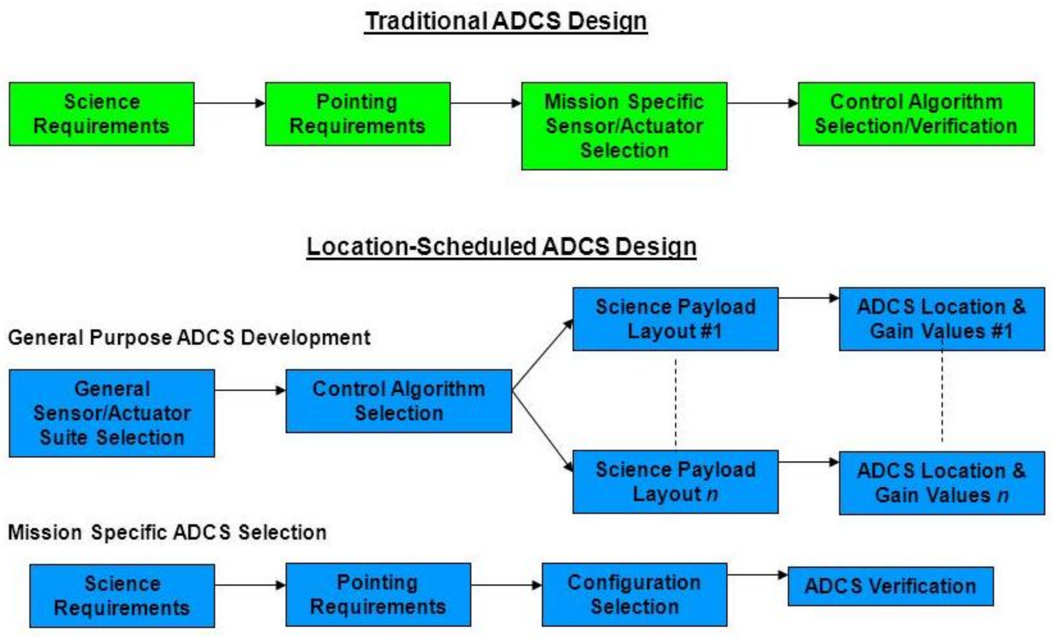
Traditional ADCS Design
Science Requirements
Pointing Requirements
Mission Specific Sensor/Actuator Selection
Control Algorithm Selection/Verification
Location-Scheduled ADCS Design
General Purpose ADCS Development
General Sensor/Actuator Suite Selection
Control Algorithm Selection
Science Payload Layout #1
ADCS Location & Gain Values #1
Science Payload Layout n
ADCS Location & Gain Values n
Mission Specific ADCS Selection
Science Requirements
Pointing Requirements
Configuration Selection
ADCS Verification

Figure 4.2 – A comparison between traditional ADCS design and location-based ADCS design.

In order to use the same ADCS unit to control multiple missions, it is necessary to determine the best possible combination of control unit location and control law gain values for each of the possible science payloads given that they have different inertia properties and likely different pointing requirements. This can be done in simulation prior to a mission being undertaken, meaning that during the actual development period the ADCS designer need only look up the baseline design for the given layout and pointing requirements. Some real-world tuning of gains will inevitably be required due to system and environmental uncertainty, but the overall amount of time required for ADCS development will be shortened substantially.

No matter what combination of actuators and sensors are chosen for the stand-alone ADCS unit there will be certain missions for which the given ADCS design simply cannot supply the necessary control authority. One advantage to developing a catalogue of ADCS designs in simulation is that it also provides an envelope of capability, allowing the ADCS designer to see what range of missions a certain control unit design can support. Additionally, the basic LSC concept can be extended to multiple ADCS units, and a spacecraft developer could create multiple gain schedules for each level of control authority enabled by a certain hardware combination. This also provides flexibility in terms of mass, power, and cost budgets, as the appropriate control unit (and its gain schedule) can be chosen based on the needs of the mission.

4.3 Detumble Maneuver Optimization

4.3.1 Fitness Function Selection

The location-based schedule of gains is generated for a particular nanosatellite form factor by means of a GA that is similar to the one presented in Chapters 2 and 3. It is necessary to augment the chromosome structure to include digits representing the possible spacecraft configurations, but the same concepts of crossover, mutation, and natural selection by means of a fitness function are the same for the LSC design problem. In fact, the same peformance metrics presented in Chapter 3, settling time (T_s) and control effort expended (C_e), are also used for the fitness functions considered here. To better understand the utility of the LSC approach two different fitness function structures will be considered in this chapter. The first function is only concerned with control effort expended, such that

$$f = -C_e \tag{4.1}$$

Thus, a design that uses the least amount of effort to drive the spacecraft to the desired state will have the least-negative fitness score and be more likely to be selected for reproduction. This function is combined with a death penalty that operates on the settling time of the rotational maneuver. In Chapter 3 it was necessary to define T_s in terms of the amount of time required for the tracking error to drop below a certain level, and a similar approach is taken here. Since the detumble maneuver drives this spacecraft to a known orientation and zero angular velocity after initially starting from the maximum value of the rotational error $\|\zeta\|$, it is sufficient to say that the system has settled when $\|\zeta\|$ has decreased to 2% of its maximum value during the simulation run. Once again this is not a traditional definition of settling time, but it is appropriate for the general multi-axis rotation considered in this chapter. The death penalty that is paired with Equation (4.1) discards any design that has not settled to the 2% metric by the end of the simulation run.

The second fitness function does not make use of the death penalty, however settling time has now been incorporated directly into the function. The equation for this new function is

$$f = -(T_s + \beta C_e) \tag{4.2}$$

which is the same function used in Chapter 3 (with T_s defined as above), and once again β is initially held equal to one. It is important to note that Equation (4.2) seeks designs which best balance two competing peformance metrics, whereas (4.1) only minimizes one metric subject to a hard bound on another metric (the death penalty). These are fundamentally different optimization goals, which is part of the motivation for testing both of them in this first implementation of LSC.

4.3.2 Control Law Selection

For this application example the LSC design methodology will seek to optimize a detumble maneuver. As described in the Introduction, a detumble maneuver is characterized by the spacecraft having high initial angular velocity and an arbitrary rotation, and it is the job of the controller to drive the spacecraft from this arbitrary state to a specified "rest" state. Numerous strategies exist for accomplishing the detumble operation, but for the 6U spacecraft application one driving design factor is is the ability to accommodate saturation of the control actuators.

The requirement that all attitude determination and control hardware be modularized to fit within 1U of volume places stringent mass, power, and volume constraints on the attitude actuators, which will in turn lead to constraints on the available control torque. In this chapter it is assumed the spacecraft is being controlled via three mutually orthogonal reaction wheels, and that these wheels cannot supply a torque greater than τ_{max}. One approach to controlling the rotational motion of a spacecraft subject to actuator saturation limits is proposed by Robinett, et al., in [28]. In this work the authors present a two-step development for their control law which first considers rigid body rotations without saturation restrictions, and then again with such restrictions imposed. In the course of synthesizing this controller the authors make use of the parameterized state space $\mathbb{R}^6$ described in the Introduction, and the stability proof for this control law is referenced to that parameterization, as opposed to the actual state space $SO(3) \times \mathbb{R}^3$. In the absence of actuator saturation, the feedback control torque is defined as

$$\tau_{us} = J(Q\dot{\omega}_d - \omega^\times Q\omega_d) + \omega^\times J\omega - k\bar{\sigma} - P\omega_e \tag{4.3}$$

where Equation 4.3 makes use of the attitude error matrix Q, the angular velocity error ω_e, and the skew-symmetric operator $(\cdot)^\times$, all of which were defined in the Introduction. Additionally, the vector $\bar{\sigma}$ is the modified Rodrigues parameters (MRP) vector describing the error in the spacecraft attitude. The MRP vector is defined in terms of the principle angle of rotational error ϕ and the principle axis of roational error ζ, both of which were used in the preceding chapter and are defined in [5]:

$$\bar{\sigma} = \tan(\frac{\phi}{4})\zeta \tag{4.4}$$

The gain k in Equation (4.3) is a scalar gain operating on the error in spacecraft attitude, while P is a 3×3 positive definite gain matrix operating on the error in spacecraft angular velocity. Thus, the feedback controller in (4.3) can be thought of as a PD-type control law (similar to that used in the preceding chapter). The feedback control law in the absence of saturation restrictions (4.3) is extended to consideration saturation effects as

$$\tau_i = \begin{cases} \tau_{us_i} & \text{for } |\tau_{us_i}| \leq \tau_{max_i} \\ \tau_{max_i} \cdot \text{sgn}(\tau_{us_i}) & \text{for } |\tau_{us_i}| > \tau_{max_i} \end{cases} \tag{4.5}$$

Equation (4.5) is defined for each axis of the spacecraft ($i = 1 - 3$), and each reaction wheel can in fact have a different value for the saturation torque τ_{max} if necessary. For this research the control saturation limit will be held constant at $\tau_{max} = 4 \times 10^{-3}$ Nm about all three axes. This corresponds to the saturation limit of the Sinclair Interplanetary RW-0.03 reaction wheel, a commonly used actuator in nanosatellite applications that has flight heritage on a recent CubeSat mission [17]. While it is outside the scope of this dissertation to address this issue directly, it should be noted that there exist certain physical limitations to the trajectories that one can asymptotically track when the controller of (4.3) is subject to known torque bounds. In essence, only certain trajectories–those for which the attitude is varying somewhat slowly with respect to time–can be asymptotically tracked using (4.5). The results presented in subsequent sections of this chapter show that for the system in question the control law defined by Equation (4.5) can drive the spacecraft to the desired state, but it should not be assumed that this same result would hold for other problems.

The stability proof presented in [28] is based upon Lyapunov's direct method [41] on the space $\mathbb{R}^6$, which is used to parameterize rigid body attitude motion. However, it is important to remember that the equilibrium point $(Q,\omega_e) = (I_{3\times3},0)$ is not rendered Lyapunov stable by this control law. In practice MRPs behave in much the same way as quaternions, and the controller presented in (4.5) will also be subject to the unwinding phenomenon. This makes the selected control law somewhat less desireable than that which was presented in the preceeding chapter, and in practical spacecraft applications it may not be teneble to implement this controller. That being said, the control law in question was selected for this first simulation-based test of location-scheduled control since it accommodates for actuator saturation in a fairly straight-foward manner. That simplicity is desirable when considering a first attempt at modularizing the spacecraft ADCS to be contained within 1U of volume.

4.3.3 Problem Formulation

Referring back to Figure 4.1, there are two possible locations for the ADCS unit, corresponding to the bottom-middle and bottom-rear cube positions in the 6U layout. Placing the ADCS module in the bottom-middle cube location will be referred to as Configuration #1, while placing the ADCS in the bottom-rear cube location will be designated Configuration #2. The task then is to find the optimal combination of control unit location and control law gain values (as defined by the fitness functions) for the detumble maneuver. For this example the spacecraft has an initial angular velocity of $\omega = \begin{bmatrix} 0.0523 & 0.0523 & 0.0523 \end{bmatrix}^T$ rad/s in the body-fixed frame, which corresponds to an angular velocity of 3°/s about each body axis. This is a typical tip-off rate antcipated for the NASA Ames Nanosatellite Deployer, which is currently scheduled for first flight in 2013. In addition to this angular velocity, the spacecraft is initiallly rotated 20° about the same principal axis $\hat{\lambda}$ that was used in Chapter 3. The task for the controller is to drive the spacecraft from this high angular velocity and arbitrary rotation to the final desired state of $R_d = I_{3\times3}$ (e.g., the spacecraft body frame B aligned with the LVLH frame) and zero angular velocity.

The last step before the GA can be executed is to define the chromosome structure for the LSC design problem. Given that two locations for the ADCS unit are to be tested, a location within the spacecraft can be represented with a single binary digit. A value of 1 corresponds to Configuration #1, while a value of 0 corresponds to Configuration #2. Referring back to Equation (4.5), the control law under study has one scalar gain and one 3×3 matrix gain. If a diagonal matrix is selected for P then a total of four gain terms must be optimized. In this work the scalar gain k is restricted to taking on integer values on the interval [1,15] while each element of the matrix P is restricted to values in the interval [0.1,1.5] in increments of 0.1. These ranges of gain value can both be represented using four binary digits, with the decimal interpretation for each gain in the matrix P being divided by 10 to bring it into the range of [0.1,1.5]. Using one binary digit to represent control cube location and four digits each to represent the gain values, a total of 17 bits are required to define one candidate design. An example chromosome is depicted in Figure 4.3, where the first bit defines control cube location, the next four bits define the value of k, the next four bits define the first entry of the matrix P, and so on.

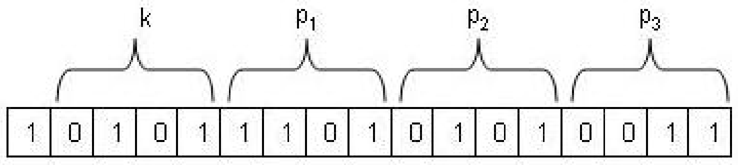

Figure 4.3 – Visualization of the chromosome structure used for the 6U controller optimization problem.

For the gain ranges defined as above, there are a total of 154 possible combinations of gain values. When multiplied by the two different control cube locations, the full space of candidate designs has 101,250 members. This design space would increase in size if a higher resolution (or a greater range) was desired for the gains. Given the relatively small global design space, an exhaustive search will be carried out along with a GA-based search and an informed trial-and-error search to evaluate the performance of the GA as compared to these alternative design methods.

4.4 Simulation Results

The simulation results presented herein were obtained using a MATLAB simulation running for 30 seconds (spacecraft time). The rotational motion of the spacecraft was simulated using the same LGVI scheme described in Chapter 3. Every candidate design was given the initial conditions detailed in the previous section and then had its fitness evaluated using each of the two fitness functions described above. Results were obtained using the informed trial-and-error method, an exhaustive search of all possible design solutions, and the genetic algorithm. Direct testing of all 101,250 solutions required 6.5 hours on the same desktop computer used in Chapter 3, while the average execution time for a run of the GA was approximately 45 minutes. Following the methodology presented in the preceding chapter, a control systems engineer was given 45 minutes to find the best possible design solution using the MATLAB simulation. The trial-and-error search was carried out by initializing the controller gains to k =1 and P = diag $\begin{pmatrix} 0.5 & 0.5 & 0.5 \end{pmatrix}$ and then changing the gain values based on the observed time response. Depending on the performance of a given design solution, the proportional gain k was tuned either up or down and the simulation was re-run. Changes were also made to the derivative gain matrix P, and the location of the control cube was sporadically changed to encourage a more diverse search of the possible solutions. The informed trial-and-error process was repeated until changes in the gain values no longer yielded an improved time response of the nanosatellite system.

The first approach used by all three design strategies to optimize the detumble maneuver was the combined fitness function/death penalty structure. The fitness function of Equation (4.1) was paired with a death penalty that eliminated any design that had not settled to 2% of the maximum rotational error ζ by the end of the 30 second simulation. Using this combination of performance metrics, the globally optimal design solution (found via exhaustive search) resulted in a fitness value of f = -0.0237. This corresponds to 0.0237 Nms of control effort being expended and a settling time of 29.8 seconds. This performance was achieved by placing the control cube in Configuration #1, and using gain values of k = 7 and P = diag $\begin{pmatrix} 0.8 & 0.3 & 0.5 \end{pmatrix}$. The component-wise time history of both the spacecraft angular velocity ω and the commanded control torque τ are depicted in Figure 4.4.

It is important to notice in Figure 4.4 that despite the fact that the spacecraft has completed

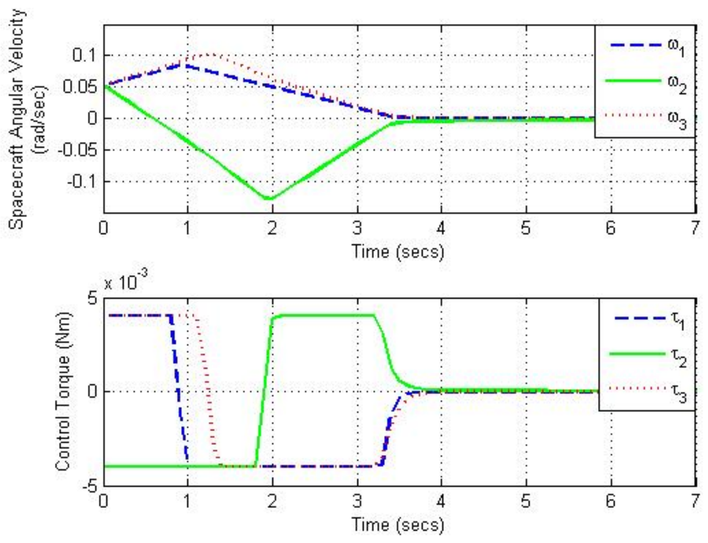

Figure 4.4 – Component-wise time response of the spacecraft angular velocity (top plot) and commanded control torque (bottom plot) for the globally optimal solution found using f = -C_e.

the majority of its reorientation after approximately 3.5 seconds, there is still some residual angular velocity that must be eliminated in order to achieve 2% of the maximum value of $\|\zeta\|$. This is accomplished over a long period of time, leading to a settling time of 29.8 seconds. This very long settling time still results in the globally optimal solution as per the definition of the fitness function, but it is important to compare that performance against the other design strategies under consideration. A comparison between the globally optimal solution found via exhaustive search, the best-performing solution found using the GA, and the best-performing solution found using trial-and-error search is given in Table 4.1. As seen in this table, the genetic algorithm far outperforms trial-and-error search in terms of control effort expended. Recall that control effort is the only term being operated on in the fitness function, and thus if one wanted to also impose a stricter settling time requirement, it would be necessary to implement that change in the death penalty.

Table 4.1 – Comparison between best-peforming designs found using exaustive search, the genetic algorithm, and trial-and-error tuning with f = -C_e

Method	**Config. #**	**f_{max}**	**T_s (s)**	C_e (Nms)	k	p_1	p_2	p_3
Exhaustive	1	-0.0237	29.8	0.0237	7	0.8	0.3	0.5
GA	2	-0.0244	19.9	0.0244	9	1	0.3	0.7
Trial-and-Error	1	-0.0334	29.8	0.0334	8	0.4	0.4	0.4

To further illustrate the differences between the best-performing solutions found for each design strategy, the norm of the spacecraft angular velocity ω and the principal axis of rotational error ζ are plotted for all three controller designs in Figure 4.5. Note that in this plot exhaustive search is abbreviated as 'ES', genetic algorithm is abbreviated as 'GA', and trial-and-error search is abbreviated as 'TE'. While the exhaustive search and genetic algorithm responses are quite similar to one another, they both far outperform the trial-and-error design, which requires much more motion from the spacecraft in order to drive it to the desired state.

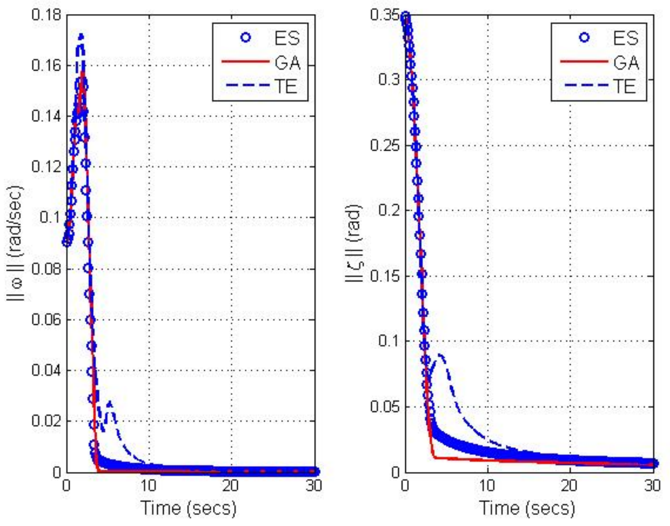

Figure 4.5 – Comparison between performance results for all three design methods for f = -C$_e$.

For the fitness function $f = -(T_s + \beta C_e)$ (used without a death penalty) the globally optimal design solution found via exhaustive search yielded a fitness value of f = -3.3247. This fitness value results from a settling time of 3.3 seconds and 0.0247 Nms of control effort being expended. For this solution the ADCS was placed in Configuration #2, and the gains were set to k = 0 and P = diag $\begin{pmatrix} 1.1 & 0.3 & 0.7 \end{pmatrix}$. It is interesting to see how drastically the settling time falls when compared with the solutions achieved using f = -C$_e$, due largely to the direct inclusion of settling time into the fitness function. The time-response of the globally optimal solution is plotted in Figure 4.6. It is worth observing that in this case the controller once again follows a bang/bang response, however now the spacecraft is clearly driven to its desired state in less than 3.5 seconds.

The performance of the genetic algorithm and trial-and-error search methods as compared to exhaustive search of all design solutions for the second fitness function is presented in Table 4.2.

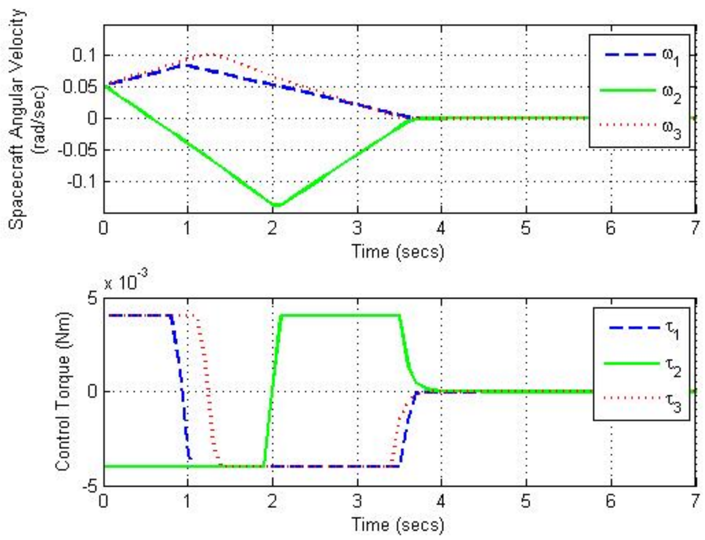

Figure 4.6 – Component-wise time response of the spacecraft angular velocity (top plot) and commanded control torque (bottom plot) for the globally optimal solution found using $f = -(T_s + \beta C_e)$.

Again, the genetic algorithm outperforms the informed trial-and-error strategy, finding a design solution that is actually very similar to that of the globally optimal solution. A comparison between the time responses of these different design solutions is shown in Figure 4.7, and again the performance of the trial-and-error solution is markedly worse than that of the GA and exhaustive search approaches.

Table 4.2 – Comparison between best-peforming designs found using exaustive search, the genetic algorithm, and trial-and-error tuning with $f = -(T_s + \beta C_e)$

Method	**Config. #**	**f**$_{max}$	T$_s$ (s)	C$_e$ (Nms)	k	p$_1$	p$_2$	p$_3$
Exhaustive	2	-3.3247	3.3	0.0247	10	1.1	0.3	0.7
GA	2	-3.325	3.3	0.025	14	1.5	0.4	1
Trial-and-Error	2	-9.3526	12	0.0526	12	0.3	0.3	0.3

4.5 Initial Assessment of the LSC Methodology

The results presented in the preceding section demonstrate that once again the genetic algorithm outperforms an informed trial-and-error approach, and now the problem has been extended to the broader challenge of location-scheduled control. Studying the time response and other characteris-

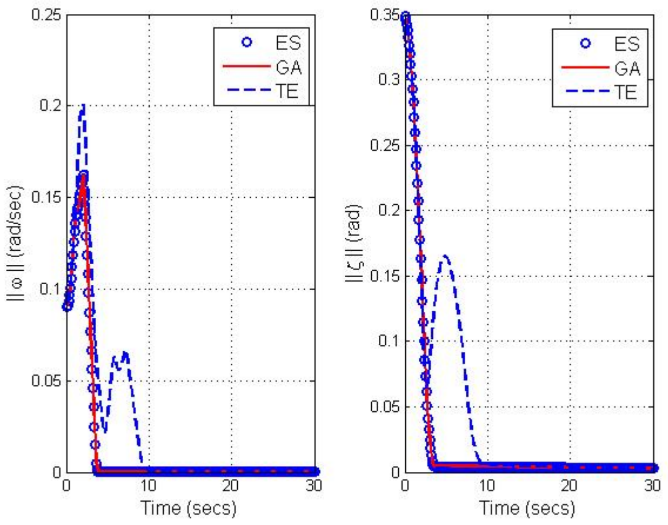

Figure 4.7 – Comparison between performance results for all three design methods for $f = -(T_s + \beta C_e)$.

tics of the globally optimal solution found via exhaustive search is helpful for the purposes of this first application of LSC, however it is important to remember that in very few situations will the globally optimal result actually be known. Recall that in the last chapter a total of six gain terms needed to be tuned, and if the same lower resolution (e.g. 15 candidate values) from this chapter was applied to that problem it would still result in 22 *million* solutions that must be tested. Such a large design space would essentially preclude design via exhaustive search. Thus, it is important to assess the validity of LSC not against what a globally optimal solution might produce, but rather what can be achieved by a control systems engineer tuning the system by hand. In that context, the GA performed quite well. For both fitness function structures the GA was able to quickly and efficiently find design solutions that outperformed trial-and-error. Similar to the results generated in the last chapter, a general advantage of using the genetic algorithm is that the initial design population is generated and then evolved randomly, allowing for a search of a broader swath of the design space.

While the results presented in this chapter are concerned with the application of LSC to the design of nanosatellites, it is important to remember that this concept can be extended to any spacecraft whose major components can be reconfigured. The recent advances in creating modular nanosatellites make them an appropriate candidate for this research, but the genetic algorithm structure is entirely independent of the size or complexity of the spacecraft under study, and can be implemented with essentially any control law. These preliminary design results demonstrate that a time savings can be achieved by applying a genetic algorithm to the dual optimization of controller gains and hardware location while still delivering a solution that peforms at a level close to that of the globally optimal solution. That is not to say that the GA will always yield results that are comparable to the global optimum, but this optimum will often not be known. That is the case in the problem presented in the next chapter, in which LSC will be applied to a hardware testbed for reconfigurable spacecraft.

4.6 Sensitivity Analysis

An interesting characteristic observed in the simulation results was the sensitivity of the different optimal solutions in terms of settling time. Small changes in control energy resulted in large changes in the settling time of a given solution. This extreme sensitivity of settling time most likely derives from the physics of the nanosatellite system itself, whereby very small changes in applied torque will result in large changes in the angular acceleration of the system. Such sensitivity to a traditional performance metric makes the creation of multiple fitness functions very appealing, as they could allow the ADCS designer to try a number of different combinations of performance metrics to dial-in the best performance for the system in question.

Once again, the preliminary GA results obtained using $\beta = 1$ in Equation (4.2) demonstrate that for many applications it may be necessary to adjust this weight factor. The original intent of including β was to allow the ADCS designer to tune the relative importance of the two performance metrics when evaluating a candidate design. However, close inspection of Table 4.2 shows that the values of settling time are approximately 100 times greater than the values of control energy, which means the "unweighted" results actually are skewed in favor of settling time. Recall that this is a similar result as was seen in Chapter 3. To understand how responsive the system was to changes in β, additional exhaustive search trials were performed using the MATLAB simulation. Surprisingly, the globally optimal solution for the detumble problem remained unchanged up until the weight factor was $\beta = 1670$, and then a new controller design became the global optimum and remained unchanged for still higher values of β. This blunt change from one design to another over such a drastic span of weight factors corroborates the previously observed sensitivity to settling time for this system. The intended use of the weight factor was to allow the control designer to make fine adjustments to the balance between settling time and control energy, but that was not possible for this system. Looking at the results presented in the previous section, a better strategy for nanosatellite control system design may be to use a fitness function in combination with a death penalty, because the designer can choose a very specific numerical value of any performance metric for use in the death penalty.

Chapter 5

Extension to Reconfigurable Spacecraft Including Hardware Testbed Validation

As described in the Introduction, on-orbit spacecraft reconfiguration is increasingly considered an effective means to meet the demand for inexpensive, versatile space platforms that can optimize their configuration to carry out a variety of tasks. There are many approaches to this concept, including mechanically connecting reconfigurable modular components [15], or repositioning members of a constellation of satellites into different large-scale formations [26]. One relatively new concept in spacecraft reconfiguration involves formations of close-proximity, non-contacting satellites which can maneuver into different configurations with respect to one another. Such a system architecture is applicable to fractionation concepts where different subsystems are carried in discrete spacecraft that work together [4]. This architecture can also be applied to vibration-isolated non-contacting platforms where the spacecraft bus and an instrument can be reconfigured to simultaneously optimize, for example, both power generation and science throughput. When spacecraft are separated by distances much less than their own length scale, control architectures must accommodate subtle and changing system geometries on-orbit.

This chapter applies the basic LSC framework to simulation and experimental results for the closed-loop control of a formation of nanosatellites operating together at separation distances on the order of centimeters or closer using a developing technology known as flux-pinned interfaces (FPIs) for spacecraft. FPIs are an enabling technology for close-proximity non-contacting spacecraft reconfiguration. They influence the relative dynamics between the different modules in a spacecraft formation using a phenomenon in superconducting physics known as magnetic flux pinning. Although FPIs provide benefits in a variety of close-proximity spacecraft maneuvers, they can be particularly useful in manipulating the kinematics of multi-module formations. Because FPIs provide both stable spring-like behavior between modules and an alterable interface which can be controlled to drive a reconfiguration maneuver, they serve as an enabling technology for close-proximity reconfigurable formations.

In order to realize this architecture on space systems, however, the system must be controlled at two levels: module-level interface maintenance/reconfiguration maneuvers and formation-level collective maneuvers. The problem of controlling the relative reconfiguration of an FPI-linked formation has been explored in previous work [48], but the challenge of controlling the formations collective maneuvers has remained an open problem. The formation-level controller must be designed to accommodate different system configurations and the associated change in system parameters, such as the inertia properties of the formation or the actuators position within the formation. For example, the formation may perform a series of check-out maneuvers in the configuration it had when it was ejected from the launch vehicle, but may later undergo rotations in a configuration optimized for data collection.

In this chapter LSC will be applied to a two-module formation of CubeSat scale nanosatellites connected by an FPI which is commanded to perform a formation-level rest-to-rest slew. While both flux-pinning and the LSC approach can be applied to complex systems involving multiple modules, a two-module formation was selected for this initial work to simplify the hardware development and to reduce the number of variables under investigation. In the maneuver under consideration, the formation starts from rest, rotates a specified angle about a given axis, and then comes to rest again. LSC is used to select both a system configuration and a set of controller gain values for the maneuver offline using the GA. Note that for the purposes of this chapter the term “location” refers to the physical placement of discrete system modules, whereas the preceeding chapter was concerned with the placement of actuator elements.

5.1 Flux-Pinned Interfaces for Reconfigurable Spacecraft

5.1.1 The Physics of Magnetic Flux Pinning

Magnetic flux pinning is a physical phenomenon found in type-II superconductors that are cooled in the presence of magnetic fields of sufficient strength [7, 36]. Once the superconductor crosses below its critical temperature, super-current vortices are created in the superconductor volume such that the magnetic flux lines become pinned into the superconductors structure. For the Yttrium Barium Copper Oxide (YBCO) used in this experiment, the critical temperature is 88 Kelvin. This process, known as field cooling, produces a superconductor with an imprinted equilibrium magnetic flux distribution; while the superconductor remains below its critical temperature it will seek to maintain this distribution. As a result, a flux-pinned superconductor will provide the nonlinear restoring forces and torques necessary to keep a magnet in an equilibrium, which can be set by the magnets initial position and orientation when the superconductor crosses below its critical temperature. In a 1-g environment, not only can the magnet be levitated above a superconductor at separation distances up to centimeters, but it can also be held in that same equilibrium suspended below or next to the superconductor, as shown in Figure 5.1.

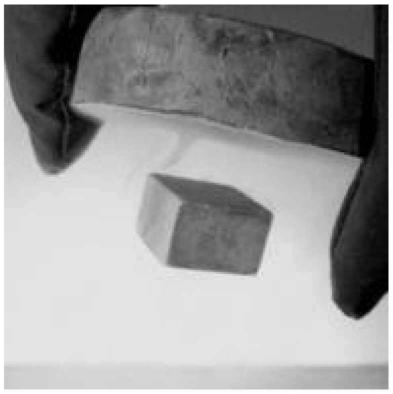

Figure 5.1 – The flux-pinned interface effect demonstrated by a YBCO superconductor and a rare earth permanent magnet.

The equilibrium created by flux pinning physics is passively stable and requires no power to maintain beyond that which is required to keep the superconductor below its critical temperature. Previous research has established that for a single-crystal YBCO disk 5.8 cm in diameter and 1.8 cm thick, flux pinning effects can be seen at separation distances up to 10 cm [37]. Changing the magnetic field strength of the magnet can alter the equilibrium position and orientation as well as the stiffness of the connection, while warming the superconductor above its critical temperature erases it. An axisymmetric magnetic field produces a non-stiff degree of freedom such that a cylindrical magnet can spin freely about its longitudinal axis, whereas a completely asymmetric magnetic field provides stiffness in all six degrees of freedom.

5.1.2 Applications of FPIs to Spacecraft

An FPI consists of a superconductor array and magnet array mounted to different spacecraft modules such that flux pinning influences the dynamics between them. The physics and the scale of flux pinning make it particularly well-suited to close-proximity spacecraft applications. For example, in a virtual space structure, spacecraft can be locked together with an FPI connection and later released if the magnetic field is turned off or otherwise canceled, safely enabling applications such as repairing or refueling modularized space systems. FPIs can also be uniquely applied to a reconfigurable spacecraft architecture in which they serve as both non-contacting structural members and kinematic mechanisms [39]. When used at close relative distances in a spacecraft formation, the FPI-connected modules in the formation operate as if they are connected by a damped, non-contacting, nonlinear spring. This property enables FPI formations to maneuver as a single flexible body during a formation-level reorientation or other collective maneuver. Past research [38] has demonstrated that the configuration of a formation connected by an FPI can be changed by manipulating the magnetic field in the interface. In this chapter it will be assumed that such a reconfiguration is possible, and LSC will only focus on optimizing the motion of the overall formation during a slewing maneuver.

5.1.3 Experiment Design with FPIs

The FPI design for this experiment is based on previous successful FPI reconfiguration experiments [48]. As such, it uses a strong cylindrical permanent magnet with its axis of symmetry perpendicular to the operating surface to provide the stiffness during the reconfiguration maneuver, with two other locking magnets to keep the system in the desired configuration. For each experimental run, the system is locked into one of the two possible configurations. Configuration #1 is a roughly linear arrangement designed to simulate a launch configuration. Configuration #2 is an “L-like” shape that a CubeSat formation may use for sensing or other operations. Both of these configurations are depicted in Figure 5.2.

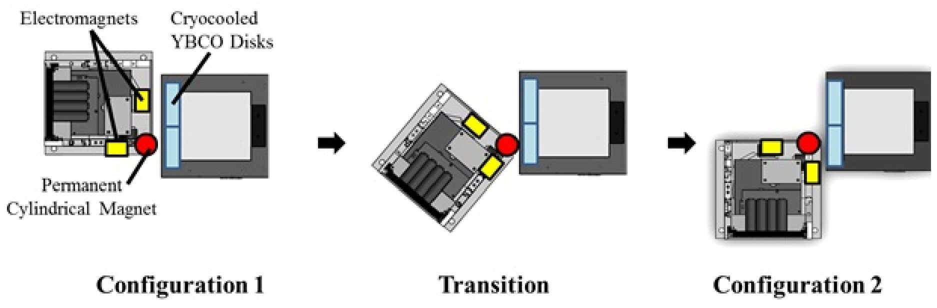

Figure 5.2 – The two candidate configurations that the FluxCraft formation can take on, each of which are tested in simulation and hardware.

For both configurations a spring-like stiffness is provided by the permanent hinge magnet and by the locking magnet closest to the superconductor. The hinge magnet for this experiment is an N52 Neodymium cylinder that is 2.54 cm high and 2.54 cm in diameter. Three 2.54 cm diameter and 0.64 cm thick N52 Neodyminum permanent toroid magnets were stacked together to form the locking magnets. These magnets interact with two YBCO disks 5.8 cm in diameter and 1.8 cm thick that are cooled in a bath of liquid nitrogen (LN_2). For each experimental run, the spacecraft formation (consisting of the two CubeSat mockups connected by their FPI in a fixed configuration) sets its initial orientation as a baseline, and then uses a PID controller to perform a 30° (0.524 rad) slew, as seen in Figure 5.3. This figure shows a slew in Configuration #1, but the same maneuver can be performed in Configuration #2. The formation retains its initial configuration because of the stiffness in the FPI connecting the spacecraft, which serves as a passive non-contacting interface between the modules. The slewing maneuvers are performed by reaction wheels governed by a PID controller, which are discussed in detail in subsequent sections.

5.2 The Reconfigurable Spacecraft Testbed

5.2.1 Hardware Description

The experimental results for this chapter are obtained using Cornell University's FloatCube testbed, which was developed as a CubeSat-scale, multibody, reduced-friction environment for hardware validation and laboratory experiments. This testbed consists of custom-made platforms equipped with

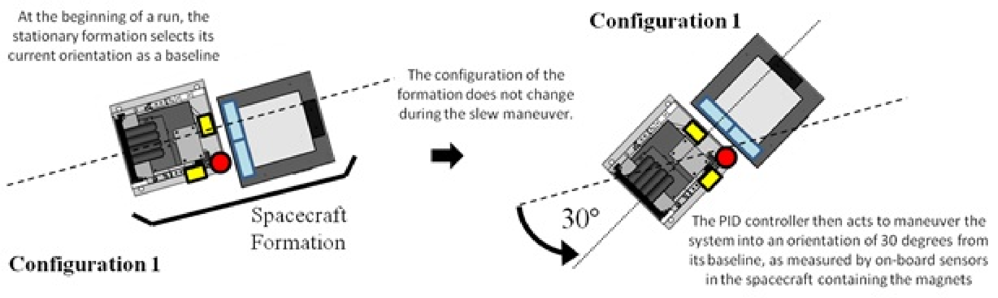
At the beginning of a run, the stationary formation selects its current orientation as a baseline
Configuration 1
The configuration of the formation does not change during the slew maneuver.
Spacecraft Formation
Configuration 1
30°
The PID controller then acts to maneuver the system into an orientation of 30 degrees from its baseline, as measured by on-board sensors in the spacecraft containing the magnets

Figure 5.3 – The rest-to-rest reorientation to be optimized using the LSC methodology.

pressurized CO_2 cartridges supported by air bearing feet that can lift payloads up to approximately 6 kg. Each platform can provide approximately 10-12 minutes of reduced-friction floating. These platforms are operated over a four-foot square piece of glass to provide a level and smooth surface for experimental operations. The FloatCube testbed can accommodate up to four separate modules at a time, as shown in Figure 5.4.

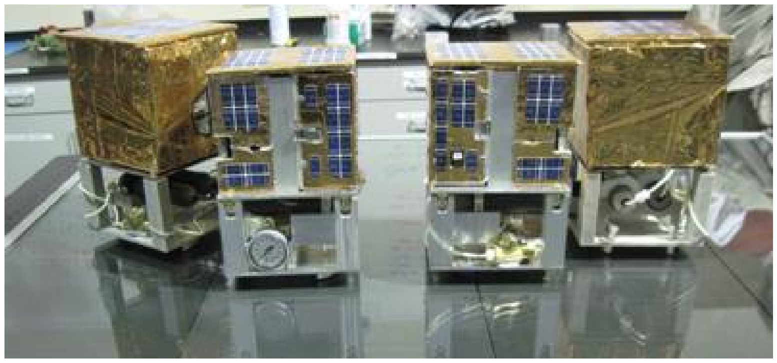

Figure 5.4 – Four examples of assembled FluxCraft/FloatCube Modules on the glass table

In order to collect information on the position and orientation of the modules on the table, the FloatCube testbed has an overhead camera-based sensing system that records position and orientation data at up to 30 frames per second by identifying targets affixed to the corners of the modules. For the configuration used in this experiment, the static resolution of position accuracy is 3.1 mm and angular accuracy is 1.2°, although rapid motion of the testbed can cause reduced accuracy or loss of data. The slewing maneuver in this experiment (shown in Figure 5.3) was designed to avoid these limitations. The data from the camera system are not integrated into the feedback control loop for the experiments described in this work; they are used only to assess the performance of the system in post-processing data analysis.

5.2.2 CubeSat Module Payloads

The FloatCube payloads used for the experiments in this work are small satellite mockups called FluxCraft that were built to approximate the CubeSat standard. These satellite modules contain laboratory equivalents of many of the same subsystems found on actual CubeSats. However, instead of using flight-optimized hardware, the components in the module are standard off-the-shelf equipment to enable low-cost laboratory experimentation. Thus, each FluxCraft is slightly larger than a CubeSat, with a measurement of 12.5 cm on each side of the cube and a mass ranging from 1-4 kg depending on the module. They mount on top of the FloatCube platforms (as shown in Figure 5.4). The full assembly of the active FluxCraft (the module with the on-board sensing and actuation capability) and its FloatCube platform in this experiment has a mass of 4.856 kg.

Each standard FluxCraft is equipped with an on-board sensing and actuating package, a communication system, and a microcontroller. In this work, the sensor system in the active FluxCraft uses an ADIS16362 Inertial Measurement Unit (IMU) that can collect information in six degrees of freedom, with a typical angular random walk of 2 $^{\circ}/\sqrt{hr}$ and 0.5°/s of angular rate precision. Data is collected at approximately 820 Hz, but this data is down-sampled to 6.4 Hz when transmitted to the controller on the PC in order to minimize the time delays associated with packaging and transmitting the data across the wireless connection. Nevertheless, the measured one-way time delay in the system is 380 ms, almost all of which is attributable to BlueTooth protocols in handling the incoming and outgoing data. Given the time scale of the experiment runs (30 seconds total were allotted in simulation and the system tended to settle in approximately 10 seconds) this is a significant time delay. The fact that the system is capable of reorienting itself in closed-loop despite this delay reflects a certain measure of robustness in the system. Such a large delay has the potential to cause marginally stable or even unstable behavior because of the discontinuity between the commanded torques and the actual state of the system; however, as will be shown in the next section, the formation is still capable of successfully performing the reorientation maneuver.

The actuators on board, shown in Figure 5.5, consist of two Maxon90 brushless motors acting as reaction wheels, both of which are aligned so that they provide torque in the axis perpendicular to the operating surface. Each motor has a maximum spin speed of 300 rad/s (2900 rpm) and a rotor inertia of 3.06×10^{-4} kg m^2. Torque or speed commands are sent from the main PC via Bluetooth

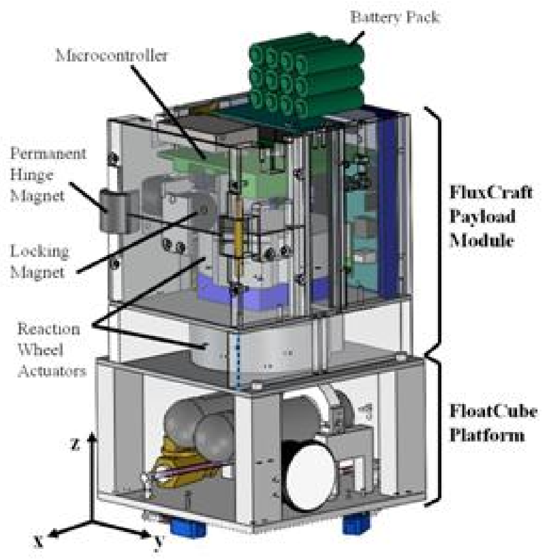

Figure 5.5 – The active cube in the Flux-Craft array which contains the reaction wheels and IMU

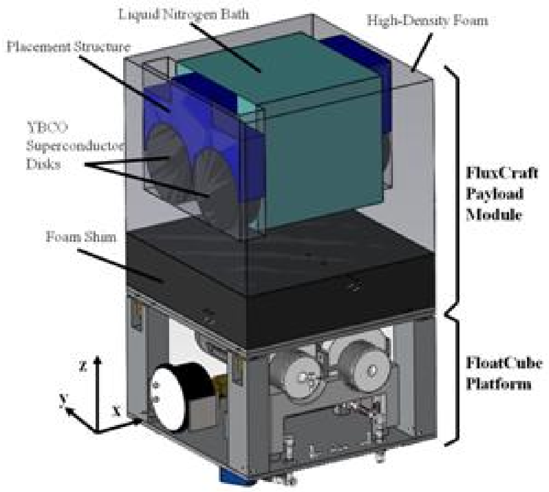

Figure 5.6 – The passive cube which contains the superconductors in the liquid nitrogen bath

to the Arduino BT microcontroller onboard the FluxCraft, with the aforementioned one-way time delay of 380 ms. Torque commands to the actuators are parsed such that the total commanded torque is shared equally between the motors on a given FluxCraft. The main PC connects to all of the FluxCraft during a particular experiment and synchronizes the timestamps for the recorded data. The controller is hosted on the main PC, where it can be easily altered for different experiment types.

A second, simpler module is also used to cool the superconductor disks in a bath of liquid nitrogen. Because the electronics and structural materials used in the standard FluxCraft have a number of undesirable behaviors at cryogenic temperatures, this module was designed to be inert and passive, carrying only the superconductor disks, a structure to hold them in place, and the LN_2. The structure of this module is constructed out of high density foam sealed to prevent leaks. This passive FluxCraft mounts on top of the FloatCube platforms in the same manner as its active counterpart, as seen in Figure 5.6, and the final assembly mass is 3.035 kg. Although this FluxCraft is a lower-fidelity model of a CubeSat system containing a cryocooled superconductor, designing a complete CubeSat-scale cryogenic thermal system is not in the scope of this experiment.

5.2.3 Experiment Procedure

Each experiment typically lasts an hour and includes 15-20 different runs. It is preceded by a standard setup procedure designed to ensure that experimental variations between runs are limited. Before each run, the FloatCube platforms are fitted with full CO_2 cartridges, and the air-bearing feet and glass surface are cleaned to ensure a low-friction environment. The batteries in the active FluxCraft are replaced with a freshly charged pack to ensure that the motors are provided a consistent voltage. The FloatCubes are then positioned on the glass, the valves turned off to prevent motion, and the FluxCraft payloads fitted to the platforms. The two modules are arranged in their test configuration for field cooling. In order to field cool the modules, the LN_2 is added to the passive FluxCraft containing superconductors with the magnets from the active FluxCraft in place (for consistency, the cylindrical magnet was touching the face of the passive cube at a tangent point, near the center of the superconductor). This cooling procedure takes 5-7 minutes in order to ensure the superconductors are below their critical temperature. Once the flux-pinning effect is verified by inspection of the formation, the PC software connects to the active FluxCraft in order to begin collecting IMU data. At the same time, the camera vision system is calibrated and set to track the points on the top of the FluxCraft. Once both data sources are confirmed to be functional, the data collection phase of the experiment begins.

Each experimental run consists of three phases. The first phase, described above, ends with the flux-pinned interface established and the modules stationary, awaiting a command. Once the modules are confirmed stationary, the second phase of the experiment involves the test operator sending the desired PC command to the FluxCraft. Each experimental run is allowed to continue until it achieves a performance to within a threshold of the desired value, or the test operator determines that an off-nominal condition is causing errors in the run. Examples of experimental variations include: a) batteries falling below their minimum allowable charge, b) pressure in the CO_2 system not providing sufficient lifting force to provide low-friction motion, c) LN_2 levels in the passive cube falling below the acceptable threshold, or d) glass surface conditions requiring alteration (for example, small ice flakes from frozen condensation on the passive cube will occasionally fall onto the glass and interfere with the floating of the modules, necessitating a re-cleaning of the operating surface). When any of these situations are determined to be affecting the results, the experiment is stopped and the system reset accordingly. However, these conditions are consistently monitored by test personnel and preventative measures taken to ensure any potential problems are addressed before an experimental run is conducted. The final phase of an experiment involves commanding the actuators to turn off and resetting the equipment for the next run. These phases are repeated until the end of the experiment, when the data sets are collected and prepared for analysis.

5.3 Application of Location-Scheduled Control

The maneuvers commanded during each run of the experiment–and the optimal combinations of system configuration and control law gains–are governed by the LSC methodology. Specifically, LSC selects one of two system configurations and tunes the three gain terms of a PID controller in parallel in order to optimize the response of the hardware testbed for the rest-to-rest slew maneuver. For this example problem, the "location" is concerned with the physical arrangement of discrete spacecraft modules. The need to reconfigure such discrete spacecraft within a formation (change their location) will alter the inertia properties of the overall system, impacting the time response for a specific maneuver. As in Chapter 4, LSC generates a schedule of gains that optimize controller gain values based upon the unique system configurations available. The location of the reconfigurable modules directly influences the gain values selected, giving rise to a location-based gain schedule.

In order to apply LSC to the hardware testbed the GA must once again be tailored to the needs of the system under study. The first step in that process is to define the structure of the chromosome representing each design. This string of digits must encompass both the configuration the formation assumes and the gain values selected for the control algorithm. For simplicity, this research only considers two distinct configurations for the collective, as shown in Figure 5.2. Since only two configurations are to be tested, only one binary digit is required to capture this design element, with a 0 indicating the system is in Configuration #1 and a 1 indicating it is in Configuration #2. The remaining digits are allocated for the gains of the control law governing the slewing of the formation. In this work a PID control law with three gain terms, k_P, k_I, and k_D, is used to implement the rest-to-rest 30° baseline slew maneuver. Early MATLAB simulations indicated that for all three gain terms values falling between 0.001 and 5 tended to yield stable, desirable system response. Eight binary digits were allocated for each gain term, providing a gain resolution of 0.0196 and 256 different possible values that each gain can take on. This leads to more than 16 million different design combinations. Once again it is not feasible to apply exhaustive search in order to optimize the system configuration and controller gains, so informed trial-and-error tuning will again be used as a point of comparison for the GA. A representation of the string of digits for a candidate design is given in Figure 5.7.

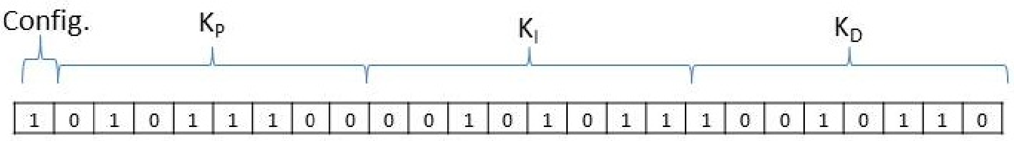

Figure 5.7 – The chromosome structure used to tune formation configuration and PID controller gains.

The second major aspect of tailoring the GA for the reconfigurable testbed is to create an appropriate fitness function. This chapter will make use of the same fitness function introduced in Chapter 4, such that

$$f = -(T_s + \beta C_e) \tag{5.1}$$

however since this chapter is concerned with a testbed operating in discrete time a slightly different definition for control effort expended is required, whereby

$$C_e = \frac{1}{N} \sum \|\tau_i\| \Delta t_i \tag{5.2}$$

where the i subscript denotes the ith time step of either the simulation or the hardware implementation and N is the total number of time steps for a particular simulation or hardware run (not to be mistaken with the inertial reference frame N). The Simulink model of the FluxCraft system makes use of a variable time-step Runge-Kutta integration scheme, and a typical simulation run generates approximately 120 data points. In contrast, a typical hardware run only generates between 15 and 30 torque commands, due to a combination of the system time delay and the hardware response time. The format of Equation (5.2) is designed to accommodate for this difference in time vectors by dividing by the length of that vector. Note that for this single-axis rotation maneuver it is sufficient to use a traditional definiton of settling time, in which a system is deemed to have settled when the spacecraft has reached 2% of the desired final value. However, as will be described in subsequent sections of this chapter, it is necessary to slightly modify this basic definition for the results obtained from the hardware testbed.

Prior to applying the fitness function, each design must pass through a death penalty applied to the angular velocity of the system control actuators. As described in the previous section, the primary hardware actuators that control the formation's slew is a set of brushless DC motors that act as momentum exchange devices. The maximum angular velocity that each wheel can achieve is 303 rad/s; therefore, the death penalty discards controller designs that result in the reaction wheels exceeding this value. Since the death penalty has the effect of limiting the torque commands coming from the controller it was decided that β should be tuned to provide an even balance between settling time and control effort expended. Early testing revealed that the design metrics T_s and C_e were approximately an order of magnitude different from one another, so the parameter β was set equal to 10 to yield fitness scores with roughly even contributions from the two metrics.

The last step before implementing the GA is to tune the variables of the algorithm itself. As in the preceeding chapters, the salient variables are the design population size (Pop), the number of generations over which the GA is executed (Num), the crossover probability (Xover), and the mutation probability (Mut). Using the results from the previous chapters as a guide, a number of test runs of the GA were studied to determine what combination of variables resulted in the highest average fitness value. Simulations revealed that the best combination of variables for this particular application is Pop = 40, Num = 20, Xover = 70%, Mut = 0.001%.

For this hardware implementation a control systems engineer was given 2 hours to tune the PID gain terms by hand in an effort to generate the best possible system response. The human engineer used the same fitness function from Equation (5.1) to guide the gain tuning process, and was given approximately twice the amount of time required for the GA to search the design space. This is more time than has been previously allotted for trial-and-error tuning, but due to the complexity of the hardware system it was felt that more time might be necessary. In addition to the trial-and-error solution, a set of gains was deliberately selected to demonstrate a poor system response as a counterpoint to the optimized system behavior. In particular, these non-optimal gains show that the systems' performance does improve with gains selected to optimize its behavior, and that the system is not inherently so well-behaved that any combination of gains will produce reasonable system performance. A summary of the best-performing PID gain values and system configurations selected for the reorientation maneuver using the three different methods is presented in Table 5.1.

Table 5.1 – Experimental PID gains and configurations.

Gain Sets	$\mathbf{k}_P$	$\mathbf{k}_I$	$\mathbf{k}_D$	**Config. #**
GA-Derived	0.0435	0.022	0.0265	2
Trial-and-Error	0.045	0.003	0.035	1
Non-Optimal	0.05	0.022	0.05	1

5.4 Model Development and Controller Implementation

5.4.1 System Model

The FluxCraft formation is capable of translating in two directions on the glass table as well as rotating about an axis perpendicular to the table plane, but in this work only single-axis rotation is considered. Four major system elements must be modeled in order to create a simulation of the formation: the FluxCraft formation properties, the disturbances acting on the system, the system actuators, and the ground computer to testbed communication. Ultimately the FPI formation is a time-delayed, nonlinear system whose motion varies somewhat from run to run. Thus, it is important to note that even with a highly accurate system model, the empirical results inevitably differ from simulations due to experimental variations that cannot be accurately captured in the simulated system model.

The GA requires a MATLAB/Simulink simulation of the testbed in order to select a system configuration and tune controller gains. For this first experimental implementation of LSC the system was modeled as two rigid cubes with non-homogenous mass distribution rigidly fixed to one another in a given configuration. Neglecting the spring-like behavior of the FPI vastly simplifies the equation of motion for the formation, and thus simplifies the evaluation of the fitness of the candidate solutions in the genetic algorithm. Even with such a simplifying assumption, it will be shown in the next section that the gains and system configuration selected still result in stable closed-loop control of the hardware.

The most notable environmental disturbance torque acting on this system is Coulomb friction between the air-feet and the glass surface. Although designed to be as low as possible, this constant torque significantly affects the system performance. To understand this disturbance, a series of open-loop friction characterization tests were performed using the FluxCraft/FloatCube assembly with the motors onboard. The variations in the magnitude of the observed friction torque as a function of angular rate of the assembly were small enough to justify a constant-torque friction model. The equation of motion for a planar rigid body with a control torque and a constant friction disturbance is

$$\ddot{\theta} = \frac{1}{J_{zz}}(\tau_c - \tau_f) \tag{5.3}$$

where $\ddot{\theta}$ is the angular acceleration of the FluxCraft formation about the body-fixed z-axis (depicted in Figure 5.5), J_{zz} is the moment of inertia of the entire formation about the z-axis, τ_c is the control torque applied by the reaction wheels, and τ_f is the Coulomb friction disturbance torque. The value of J_{zz} differs for each of the two system configurations, which in turn changes the time response of the system. This fact provides a central motivating factor for using LSC, in that better performance can be achieved when system configuration and controller gains are tuned in parallel. The numerical value of J_{zz} for each of the two system configurations is derived from SolidWorks CAD models. While such models faithfully represent inertia properties when the inputs are correct, uncertainties in the fabrication and alignment of components may contribute to discrepancies between the performance of the simulation and that of the testbed.

The last step in modeling the system is developing an appropriate dynamics model for the two Maxon90 brushless DC motors that serve as reaction wheels in the formation. These actuators are aligned so that they provide torque in the axis perpendicular to the operating surface, and are subject to limits in angular acceleration and angular rate, which are defined by manufacturer specifications. The jerk limit was determined experimentally using open-loop tests. The Simulink model for the reaction wheels accepts torque commands from the PID controller and applies a saturation and rate-limited torque to the modeled FluxCraft array. The motors themselves are rate-governed and cannot be controlled with a commanded torque directly. Thus, a number of characterization tests were performed in order to estimate the mapping between a commanded speed and the output torque, and the resulting transformation is applied to the torque command prior to it being sent to the FluxCraft actuators.

Table 5.2 provides a summary of the numerical values of the parameters used to model the formation considered in this work. These values were incorporated into the final version of the Simulink model, depicted in Figure 5.8. This system model was used by the GA to generate the gain/configuration schedule for the rest-to-rest slew under study. Note that all simulation runs were executed for 30 seconds of simulation time, regardless of the state of the simulated system at the end of the run. In simulation the GA was able to select a combination of controller gains and hardware configuration that settled to the desired state in less than 30 seconds, so this became the baseline against which other candidate configurations were compared.

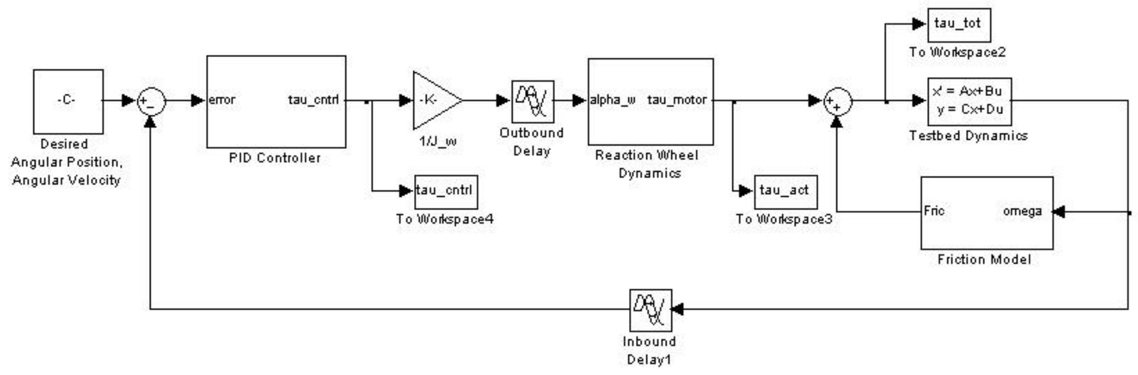

Figure 5.8 – The Simulink block diagram used for simulated control of the reconfiguragable testbed.

Table 5.2 – Reconfigurable testbed simulation parameters.

Parameter	**Value**	**Units**
θ_d	0.524	rad
$\dot{\theta}_d$	0	rad
J_{zz} Config. #1	0.08	kg m^2
J_{zz} Config. #2	0.0923	kg m^2
Colulomb Friction Torque	0.0188	Nm
Reaction Wheel Velocity Saturation Limit	303.7	rad/s
Reaction Wheel Acceleration Saturation Limit	0.115	rad/s^2

5.4.2 Data Processing and Controller Implementation

Real-time closed-loop control consists of a PID controller operating on a desktop PC, with state data and torque commands transmitted via BlueTooth. The controller relies on rate sensing data which, in the hardware implementation, is collected by the IMU sensor on board the active FluxCraft. In order to minimize the time delays in the system, the FluxCraft stores a set of five collected samples before transmitting the data to the PC (where the controller is implemented). While the time delay in the system is large by the standards of many digital controllers, it is small enough as compared to the bandwidth of the system to not impact the feasibility of the experiment, which is the central focus of this research. Upon receiving this packet from the FluxCraft, the PC then implements a series of data processing techniques to obtain the error states necessary for the control law. The resulting commanded torque is then translated into the motor rate commands and transmitted back to the FluxCraft. An overview of this procedure is shown in Figure 5.9, and is replicated for the simulated system (including inbound and outbound time delays of 380 ms) in Figure 5.8.

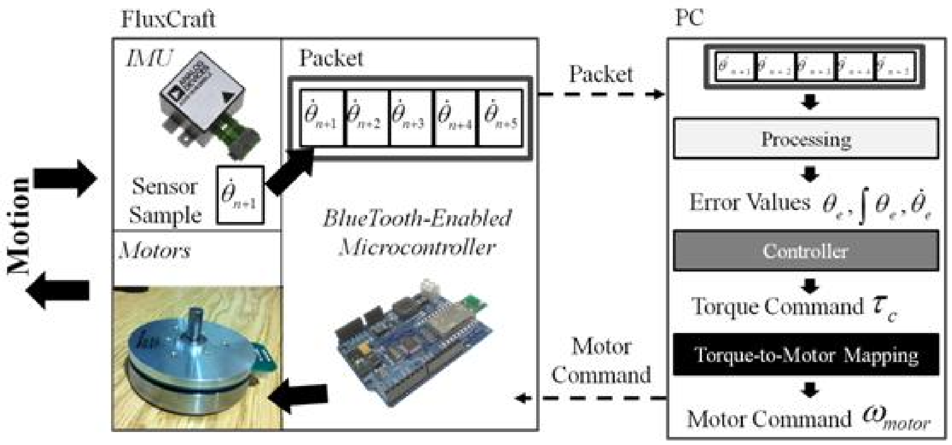
FluxCraft
PC
IMU
Packet
$\dot\theta_{n+1}$ $\dot\theta_{n+2}$ $\dot\theta_{n+3}$ $\dot\theta_{n+4}$ $\dot\theta_{n+5}$
Packet
Sensor Sample $\dot\theta_{n+1}$
Motion
Motors
BlueTooth-Enabled Microcontroller
Processing
Error Values $\theta_e, \int\theta_e, \dot\theta_e$
Controller
Torque Command τ_c
Torque-to-Motor Mapping
Motor Command
Motor Command ω_{motor}

Figure 5.9 – Flow of data from the IMU data collection to the command of the motor's angular rate.

When the packet of data arrives at the PC, the first processing step is to subtract out the IMU bias values, which are found in a calibration step completed prior to each experiment. The unbiased data is then processed to determine the proportional, integral, and derivative error signals. In this set of experiments, the desired formation angle (as depicted in Figure 5.3) is 30°, and the desired derivative and integral of the angular error is zero.

The derivative of the angular error is the most straightforward to find, since the derivative of the angular error is the error in angular rate. The IMU directly reads angular rate, and so the weighted average of the five values in the packet is taken as the current value (where the weights from Booles Rule with step size of 0.25 are used). It is worth noting that this approach effectively applies a low-pass filter on the incoming IMU data. The PC software then calculates the estimated change in the angle corresponding to the angular rates recorded in the packet. At each of the five sample time steps, Boole's rule is used to integrate the five previous angular rate values (using values from previous packets as necessary or zeros if it is the first packet). The output of this process is a set of values representing the change in the angle that occurred between each data sample. The summation of these values over time produces an angle estimate for the module. This angle can be directly compared to the desired angle (30°) to produce the proportional error term. The integral error term is found in much the same way–however, in this case Booles rule is applied to the previous angular positions (as opposed to rates) and summed. These processing steps are depicted in Figure 5.10.

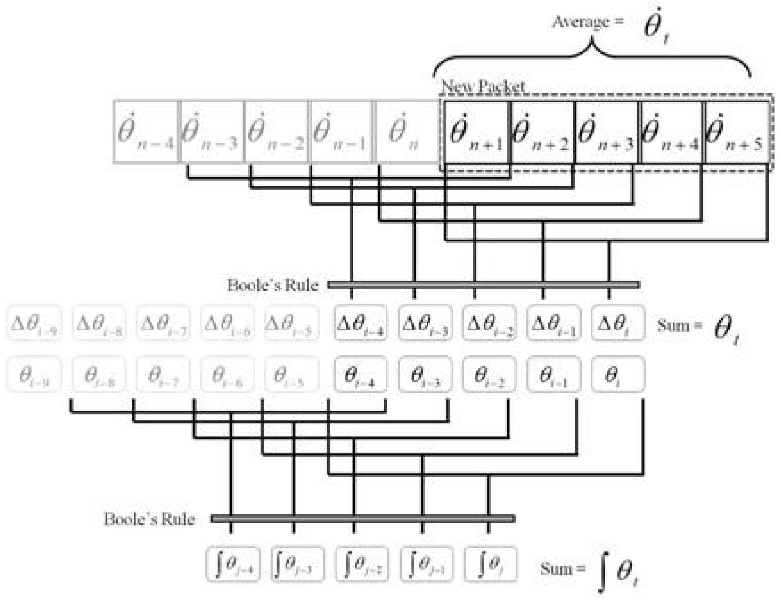
Average = $\dot{\theta}_t$
New Packet
$\dot{\theta}_{n-4}$ $\dot{\theta}_{n-3}$ $\dot{\theta}_{n-2}$ $\dot{\theta}_{n-1}$ $\dot{\theta}_{n}$ $\dot{\theta}_{n+1}$ $\dot{\theta}_{n+2}$ $\dot{\theta}_{n+3}$ $\dot{\theta}_{n+4}$ $\dot{\theta}_{n+5}$
Boole's Rule
$\Delta\theta_{t-9}$ $\Delta\theta_{t-8}$ $\Delta\theta_{t-7}$ $\Delta\theta_{t-6}$ $\Delta\theta_{t-5}$ $\Delta\theta_{t-4}$ $\Delta\theta_{t-3}$ $\Delta\theta_{t-2}$ $\Delta\theta_{t-1}$ $\Delta\theta_{t}$
Sum = θ_t
θ_{t-9} θ_{t-8} θ_{t-7} θ_{t-6} θ_{t-5} θ_{t-4} θ_{t-3} θ_{t-2} θ_{t-1} θ_{t}
Boole's Rule
$\int\theta_{t-4}$ $\int\theta_{t-3}$ $\int\theta_{t-2}$ $\int\theta_{t-1}$ $\int\theta_{t}$
Sum = $\int\theta_t$

Figure 5.10 – Process for determining current angular position, rate, and integral from a data packet of five samples.

Once the angular position, rate, and integral and the appropriate errors are found, the PID controller can be implemented with the following expression for control torque:

$$\tau_c = k_P\theta_e + k_I\int\theta_e + k_D\dot{\theta}_e \tag{5.4}$$

where θ_e is the error in the angular position of the array, $\int\theta_e$ is the integral of that error, and $\dot{\theta}_e$ is the derivative of the error. One advantage to the control law structure of Equation (5.4) is that it does not require differentiating measured values from the IMU, which would amplify noise. Given the processed data and the system gains, the control law then calculates the commanded torque.

5.5 Reorientation Maneuver Optimization Results

The simulated and empirical data both offer insight into LSC and FPI performance, but the amount of data gathered during the experiments can be difficult to parse when seen in raw form. Thus, the results and discussion are presented together in two main focus areas emphasizing the objectives of the research. This first section will compare the simulated and experimental performance of the system using the LSC methodology, and the second section will compare the systems performance using gains obtained from different selection techniques.

5.5.1 Collected Data Overview

The testbed collected two sets of position and rate data: one from the on-board IMU and the other from the overhead camera-based vision system. The IMU provides angular velocity information about three axes from the active FluxCraft, and the camera system measures the absolute translational and rotational position of each module in the FluxCraft formation. Each data source is used to determine the states of the formation in post-processing, although the IMU data alone is used in the control law. Figure 5.11 shows an example of the time history of the angular position and angular velocity of the testbed using both data sources. Note that both the IMU and camera system data presented in this figure (and in subsequent figures) plot the motion of the active FluxCraft. The differentiated camera measurements provide noisy angular velocity results, but the integrated IMU data tracks the camera orientation data closely. The performance of the formation–in particular its final orientation angle relative to the testbed and the time it took to settle to within 2% of that final value–was calculated using data such as that presented in Figure 5.11

The torque commands sent from the PC to the FluxCraft were also recorded. This data set provided the basis for determining the control effort expended by the testbed for each maneuver. As described in Section 5.4, the control effort metric is calculated as the discrete integral of the commanded control torques sent from the desktop PC to the testbed, where the time step for this

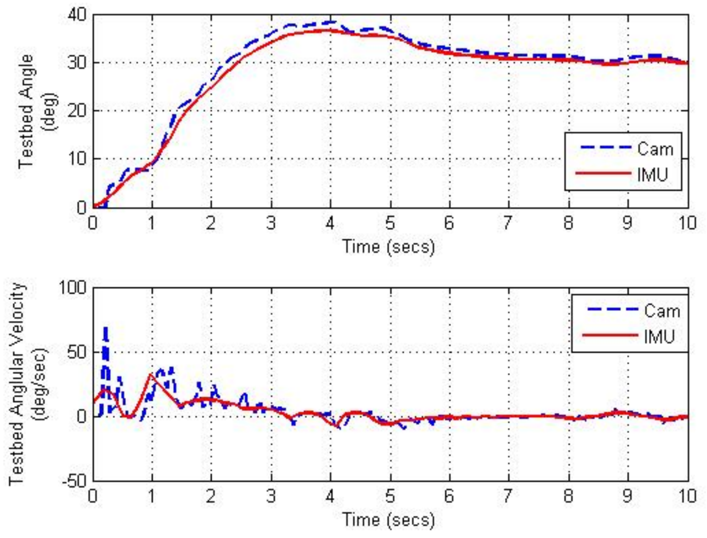

Figure 5.11 – Example time response obtained form the hardware testbed.

integral is driven by the communication delay rate between the PC and the FluxCraft. Using the control effort and the information from the position and rates of the system for a given test run it is possible to calculate a total fitness score (defined in Equation (5.1)) for the testbed.

5.5.2 Comparison Between Simulated and Hardware Results

A comparison between simulated and hardware performance metrics for the design solution obtained using LSC is presented in Tables 5.3 and 5.4. Recall from Table 5.1 that the GA selected a set of gains and chose to operate the testbed in Configuration #2; to understand the utility of this choice the same set of performance results are also presented for Configuration #1 (with the same controller gains as in Configuration #2). The number of individual test runs completed for each hardware test is included to provide a context for the performance information found in the tables. The minimum and maximum values of settling time and the final settling angle (θ_f), as well as the data measurement source for these minima/maxima is presented along with the mean value over all test runs. A data value is marked with an asterisk if the hardware data is derived from the IMU; otherwise, hardware data can be assumed to come from the camera system. While the sample sizes (11 and 7 runs, respectively) are too small to derive any meaningful statistical analysis, presenting minimum, maximum, and mean values does provide a sense of the nature of the testing data. The control effort and fitness scores are both calculated from the corresponding experimental or simulation data. As seen in Tables 5.3 and 5.4, both the simulated and hardware systems perform better (e.g. have a larger fitness score) when using the GA-derived gains in Configuration #2 than in Configuration #1. In simulation the system only has a slightly higher fitness score when in Configuration #2 as compared to Configuration #1, but in the hardware experiments this difference was more pronounced. The mean settling times and the final settling angles of Configuration #2 are also better than Configuration #1, and the minimum and maximum angle values are closer to the desired final value of 30°.

Table 5.3 – System performance for GA-derived gains in simulation and experiment (where * denotes IMU data; otherwise data is derived from the camera system).

Tests			**Settling Time (s)**		**Settling Angle θ_f (°)**	
Test Type	Data Source	# Runs	Range (min/max)	Mean	Range (min/max)	Mean
GA Config. #2	Hardware	11	4.1*/12.9	7.7	27.2/33.9	30.2
GA Config. #2	Sim	1	6.1/6.1	6.1	30/30	30
GA Config. #1	Hardware	7	5.7/11.9	9.3	25.6*/36.8*	30.4
GA Config. #1	Sim	1	6.6/6.6	6.6	30/30	30

Table 5.4 – System performance for GA-derived gains in simulation and experiment (where * denotes IMU data; otherwise data is derived from the camera system).

Tests			**Control Effort (Nms)**		**Fitness Score**	
Test Type	Data Source	# Runs	Range (min/max)	Mean	Range (min/max)	Mean
GA Config. #2	Hardware	11	0.01/0.05	0.04	-5.1/-12.9	-7.7
GA Config. #2	Sim	1	0.01/0.01	0.01	-6.1/-6.1	-6.1
GA Config. #1	Hardware	7	0.01/0.066	0.04	-5.7/-11.9	-9.4
GA Config. #1	Sim	1	0.02/0.02	0.02	-6.6/-6.6	-6.6

In order to make a one-to-one comparison between the different testing conditions, the metric used for settling time was modified slighly to consider the amount of time required for the testbed to settle to within 2% of the final measured angular position, denoted θ_f. As can be seen in Tables 5.3 and 5.4, all of the hardware tests did settle to some final angle, but oftentimes that final angle was not the 30° specified by the closed-loop controller. This steady-state error was likely induced by the static friction between the table and the testbed, a disturbance which is hard to directly eliminate when the sensor data is being packetized. Furthermore, there is a fundamental coarseness with which torque commands can be applied to the testbed, which when coupled with the aforementioned time delays can result in the formation simply settling at a value other than 30° because by the time the final torque command is applied the system has already achieved the desired final orientation.

While one might expect the testbed system to not have any steady-state error, the motor saturation effects, the system time delay, and the presence of a persistent and roughly-modeled disturbance torque all impact the behavior of the hardware testbed. One positive effect of the discrepancy between the steady-state error of the testbed and that of the simulated system is that it demonstrates a certain measure of robustness. In spite of the unmodeled system elements which result in steady-state error (including the flexible dynamics of the FPI), the hardware system is able to complete the reorientation maneuver using gain values derived from simulation test runs. If the controller design strategy was not robust to such discrepancies in system parameters it is possible that very poor or even unstable behavior would have resulted from the simulation-derived gains. While a formal robustness analysis was not performed, the experiment results demonstrate that acceptable system performance can still be accomplished despite differences between the simulated and hardware testbeds.

The performance data for both system configurations are relatively close, but the aggregate data shows that differences do exist. As seen in Table 5.2, the rotational inertia J_{zz} of Configuration #1 is similar to that of Configuration #2; so the system response between the two configurations should not show drastic differences. However, looking at the data for all hardware runs, the testbed takes on average about two seconds longer to reorient 30° when it is in Configuration #1 as opposed to Configuration #2. This is a difference that could drastically impact spacecraft proximity operations or collaborative architectures such as sparse-aperture optical spacecraft. A strength of the LSC approach is that the system configuration is optimized in parallel with the controller gain values. So, regardless of how large or small a spacecraft may be, the system dynamics are necessarily incorporated into the optimization process. For small spacecraft, system inertias can differ by 0.01 kg m^2, and, as seen in this work, such a small difference will in fact change the time response of the system.

Another discrepancy between the hardware data and the simulation data is that the hardware system appears to require approximately twice as much control effort to perform the reorientation maneuver. Again, the overall trend is still valid in that the system performs better (i.e., uses less control effort) in Configuration #2. This difference is likely due to inaccuracies in predicting the overall inertia of each spacecraft in the formation as well as unmodeled friction within the reaction wheels or static friction effects of the glass table.

5.5.3 Comparison of Gain Tuning Methods

This section compares the performance of the best design solutions obtained using each of the three previously-described gain tuning methods. The simulated time responses of the best-performing solutions for each method (listed in Table 5.1) are plotted in Figure 5.12, and Figure 5.13 shows the hardware response using those same gains. Note that in these figures 'GA' denotes the genetic algorithm solution, 'TE' the informed trial-and-error solution, and 'NO' the poor-performing (non-optimal) solution.

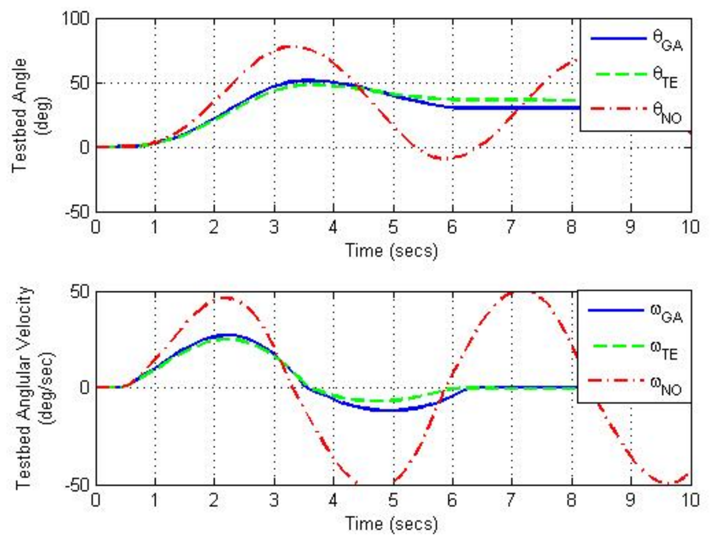

Figure 5.12 – Simulated time response of the best-performing solution found using each of the three design methods.

Numerical data associated with the different gain-tuning methods are presented in Tables 5.5 - 5.8. Tables 5.5 and 5.6 detail the settling information for the different systems and Tables 5.7 and 5.8 give the control effort expended and the final fitness scores for the systems. Once again, minimum, maximum, and mean values are presented to help provide a sense for the behavior of the given experimental set-up. Again, unmodeled system dynamics introduce a certain amount of

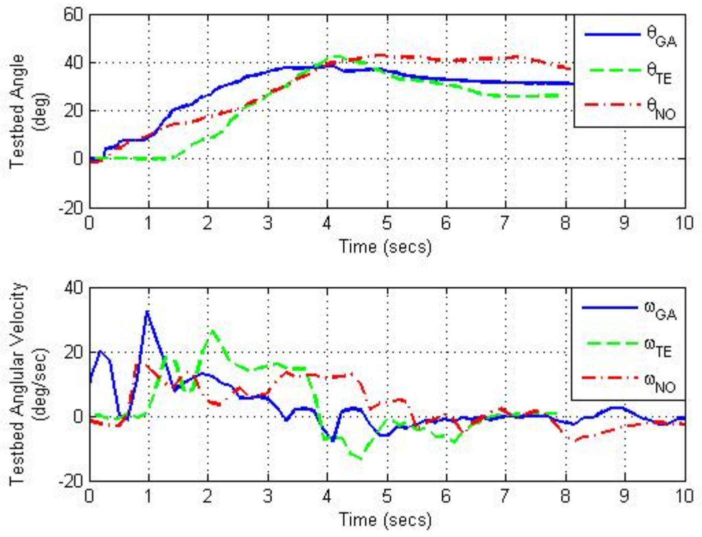

Figure 5.13 – Hardware time response of the best-performing solution found using each of the three design methods.

steady-state error into the hardware runs, so these trials were evaluated by measuring how quickly the system nears its final angular position (θ_f) while tracking the magnitude of this error.

Table 5.5 – Comparison of aggregate gain source settling time data (where * denotes IMU data; otherwise data is derived from the camera system).

Tests			**Settling Time (s)**		
Test Type	Gain Source	# Runs	Hardware Range (min/max)	Hardware Mean	Sim.
GA Config. #2	GA	11	4.1*/12.9	7.7	6.1
GA Config. #2	TE	5	6.1*/21.8	12.0	8.6
GA Config. #1	GA	7	5.7/11.9	9.3	6.6
GA Config. #1	TE	2	4.2*/9.9	7.5	7.2
GA Config. #1	NO	5	10.3*/26.4*	15.1	n/a

These data reveal that–as expected from simulation–the GA-derived combination of configuration and gains is the best overall performer in hardware, as determined by the hardware's average fitness score. In particular, the data (on average) show that this configuration and gain combination had the best fitness score (the lowest absolute value), the least error in the final settling angle, and overall low mean settling times and control effort. The second best performance was found with the trial-and-error gains applied to Configuration #1, followed by the GA gains applied to Configuration

Table 5.6 – Comparison of aggregate gain source final settling angle data (where * denotes IMU data; otherwise data is derived from the camera system).

Tests			**Settling Angle θ_f (°)**		
Test Type	Gain Source	# Runs	Hardware Range (min/max)	Hardware Mean	Sim.
GA Config. #2	GA	11	27.2/33.9	30.2	30
GA Config. #2	TE	5	25.8/29.5	27.5	30
GA Config. #1	GA	7	25.6*/36.8*	30.4	30
GA Config. #1	TE	2	29.6/31.9*	30.5	30
GA Config. #1	NO	5	28.1*/35.4*	32.3	n/a

Table 5.7 – Comparison of aggregate gain source control effort data.

Tests			**Control Effort (Nms)**		
Test Type	Gain Source	# Runs	Hardware Range (min/max)	Hardware Mean	Sim.
GA Config. #2	GA	11	0.01/0.05	0.04	0.01
GA Config. #2	TE	5	0.02/0.15	0.05	0.02
GA Config. #1	GA	7	0.01/0.07	0.04	0.02
GA Config. #1	TE	2	0.03/0.03	0.03	0.01
GA Config. #1	NO	5	0.04/0.06	0.05	0.03

Table 5.8 – Comparison of aggregate gain source fitness function data.

Tests			**Fitness Score**		
Test Type	Gain Source	# Runs	Hardware Range (min/max)	Hardware Mean	Sim.
GA Config. #2	GA	11	-5.1/-12.9	-7.7	-6.1
GA Config. #2	TE	5	-6.8/-21.8	-13.1	-8.6
GA Config. #1	GA	7	-5.7/-11.9	-9.4	-6.6
GA Config. #1	TE	2	-7.6/-9.9	-8.8	-7.2
GA Config. #1	NO	5	-10.4/-16.4	-13.4	n/a

#1. The trial-and-error gains in Configuration #1 have the lowest mean settling time and control effort expended for the hardware runs, but not the lowest overall fitness score. This is due to the wide difference in the number of experiment runs between Configuration #1 and Configuration #2, which impacts the calculated mean value of the performance metrics. It is worth noting that the second-best performer (TE in Configuration #1) only had two successful experiments, suggesting that the data may not be as reliable as for other test types. More experiments would need to be conducted to draw further conclusions about this data.

It is interesting to observe that the performance of the non-optimal gains was much better on the hardware testbed than in simulation. In simulation the non-optimal gains result in marginal stability of the testbed, while in hardware these same gains only result in poorer, albeit stable, performance. This discrepancy must be attributed to unmodeled aspects of the hardware testbed, such as increased friction, different dynamics within the reaction wheels, or effects from the system time delay being different from the single, constant value used in simulation. The effect of these unmodeled system elements is that they actually result in a certain amount of robustness, in that the simulated system was only marginally stable. Neither the time delay nor the friction components were modeled in an exactly perfect manner, and the true hardware system responded better to the application of the control law with non-optimal gains.

5.6 Assessment

The results presented in this chapter show that the performance trends anticipated by simulation of the LSC-generated solutions were verified on the testbed, despite the presence of unmodeled dynamics, such as the flexible motion of the FPIs. Hardware results also show that the gains and configuration selected by the LSC methodology produced the best fitness score, although the small number of experiments made it difficult to draw conclusions about the other high-performing configuration (such as the trial-and-error gains in Configuration #1). The current limitations of the work presented here include unmodeled flexible motion in the simulations, a lack of sensor noise in the system model (or any filtering scheme to accommodate such noise), and the large time delay present in the system. This time delay was the result of multiple sensing systems and off-board control algorithm execution, and the time delay included in the Simulink model was the best possible approximation of those delay sources. The relatively few experiments made a comprehensive error or statistical analysis difficult, and the relatively simple two-spacecraft formation does not fully leverage the utility of the LSC methodology. However, even given these limitations, this work represents an important testbed-based assessment of location-scheduled control feasibility, the first hardware-based evaluation of FPI behavior during a formation-level maneuver, and an experimental demonstration highlighting the utility of a CubeSat-scale reconfigurable formation testbed.

Chapter 6

Conclusion

6.1 The LSC Approach

This dissertation introduced a new approach to the design of the ADCS of nanosatellites known as location-scheduled control. LSC hopes to anticipate the needs of the science payload of a wide array of missions by creating a schedule of design solutions *a priori.* This schedule of designs is valid for both of the reconfiguration approaches presented in this work, rearrangement of modules between missions or reconfiguration of discrete components on-orbit. It is understood that no real-world system will function in the exact manner predicted by simulation, but that limitation is a truth for all controller design methods, not just location-based design. Accepting that a small amount of fine-tuning will be necesssary as the final hardware components come together, there is still a valid contribution to be made by anticipating (as best as possible) the optimal way in which a spacecraft could perform for a given science payload undertaking a specific rotational maneuver.

In its broadest instantiation, the LSC approach could be undertaken for multiple spacecraft layouts (e.g. 6U and 12U), using multiple ADCS cubes (a "low-performance" model and a "high-performance" model), tuned for multiple rotational maneuvers (e.g., detumble and tracking). If all of this work is performed prior to any missions being initiated, then the spacecraft design team has a powerful "toolbox" of solutions that can be drawn upon during the early phases of future missions. Other subsystems, such as the telecommunications system or the command and data hanlding system, already have this broad-base set of tools available to them, in that a specific radio receiver or microprocessor can be selected based on the needs of the mission. However, unlike these other systems, the nature of ADCS hardware largely lends itself to being packaged as a stand-alone module, enabling it to be relocated based on the needs of a particular mission. That potential has existed for a number of years now thanks to companies like Sinclair Interplanetary, Maryland Aerospace, and Isis, and LSC simply aims to harness that potential.

Application of LSC to spacecraft that reconfigure on-orbit is different in that we now define a "location" as a particular configuration of discrete modules, but the same broad potential still exists. As reconfigurable spacecraft technologies become more commonplace it is likely that a collection of standardized modules will arise, and the LSC design strategy can be applied to this collection of modules. For example, a known science payload may be packaged into one module with a known collection of support modules around it, and LSC can be tasked to select the optimal configuration of those modules for any number of rotational maneuvers. It is likely that the main ADCS components will all live in the same unique module, so in many ways the overall design approach really is no different from the basic ADCS design problem presented in Chapter 4. The simulation and hardware results presented in this dissertation demonstrate that the LSC approach shows promise, and it would be helpful to develop more example problems to reinforce the existing results.

6.2 GAs for Design Optimization

The nature of genetic algorithms make them a perfect tool for LSC. The fact that one can combine multiple disparate design features into a single string of binary digits for the purposes of optimization makes tuning gains *and* physical parameters essentially seamless. The stochastic search of the design space as driven by crossover and mutation can be helpful for complex problems, and the user-defined fitness functions allow for relatively precise control of the parameters that are to be optimized. That being said, a genetic algorithm must be tuned for each new application to which it is applied, and due to the heuristic nature of the algorithm there is no set approach to the tuning process. Throughout this dissertation a number of different combinations of design variables (Pop, Num, Mut, and Xover) were tested, but there is no hard proof that these were the best possible combinations. The performance results presented in Chapters 3-5 indicate that the GA tended to outperform both trial-and-error search and exhaustive search, but more work could certainly be done to better confirm this hypothesis. Past research [9] has actually applied a “meta-GA” to the problem of tuning the GA used for parameter optimization, and it would be interesting to apply that same approach to the system presented here.

Traditional optimization strategies often seek a global miminum to a known cost function. One common tool for such optimization problems is the MATLAB ‘fmincon’ function, which applies a gradient search technique to the problem under study [29]. The complex design space demonstrated in Chapters 3 and 5 indicate that gradient search could become stuck on local maxima/minima, and due to the mathematical rigors of miminizing a complex cost function it is also likely that such an optimization approach would simply take a very long time. When assessing the efficacy of the genetic algorithm for LSC it is critical to remember that one cannot apply the intuition of optimization methods for gain tuning only. If one seeks to optimize both gain values and the physical location of control actuators it is not as simple as just placing poles on a root locus diagram or solving an LQR equation. Despite the drawbacks of lacking a rigorous proof of optimality, the ease with which one can obtain well-performing design solutions for complex problems make genetic algorithms an ideal tool for LSC.

6.3 Future Work

Every example presented in this research has neglected the attitude determination part of the overall ADCS design problem. In each case it was assumed that the spacecraft had perfect knowledge of its state, a condition which never exists in real-world systems. This omission was made intentionally in order to better assess the performance of the different design methods under study, because the stochastic nature of “real” systems can make it hard to quantify the behavior of a particular design approach. If, for example, an engineer was trying to optimize a fitness function that included C_e but the actuator itself was producing a noisy output torque signal, the calculated fitness value would vary widely in between trial runs. In order to force a one-to-one comparison between the GA or the other methods all sensor/actuator noise was omitted from the trials.

Given that there now exists a set of baseline results which demonstrate that LSC by means of a GA has a tenedency to outperform exhaustive search or trial-and-error search it would be worthwhile to expand the tests to noisy systems. The FluxCraft testbed described in Chapter 5 is a ready example of this, given that the IMU used to collect state information (seen in Figure 6.1) produces noisy measurements and that noise was more-or-less ignored in the data collection process of Chapter 5. One possible strategy would be to inject noise in simulation into a system that was optimized under a no-noise assumption, which is similar to the sensitivity tests that were performed in some of the chapters of this dissertation. Another possible approach is to run the GA or trial-and-error optimization with a simulation that has system noise. The challenge here would be that the noise properties–and hence the performance that is being optimized–varies from run-to-run, and it is hard to say how well the GA (or a human engineer) could find an optimal solution. Both approaches merit further testing to better understand the efficacy of LSC in the presence of system noise.

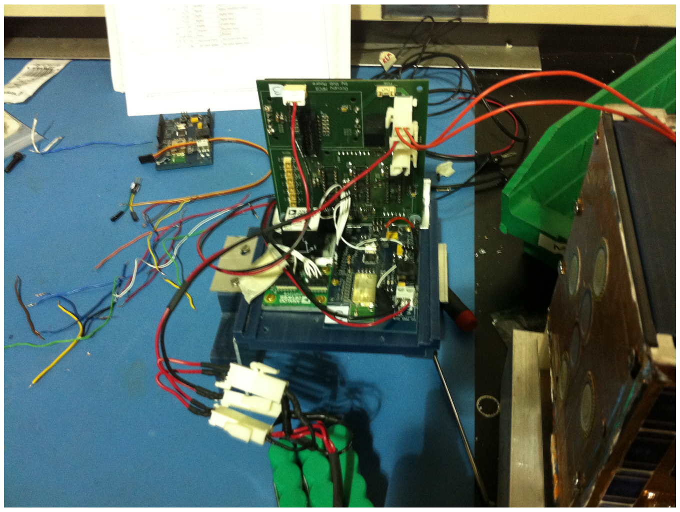
Figure 6.1 – The inside of the active FluxCraft used in Chapter 5, with the Analog Devices IMU visible.

While there is a somewhat limited set of metrics that are used to assess the performance of a control system, in future applications of LSC it could be intstructive to test different performance metrics within the fitness function. Each of the application examples utilized some combination of settling time and the amount of control effort expended; although these are benchmark criteria for controller design the performance of the LSC methodology may well be better understood if a more diverse group of metrics are implemented. Possible candidates include percent overshoot, some description of tracking error, or additional measures for the amount of energy being consumed by the system. The evolution of the design population in the GA is intimately linked to the nature of the fitness function being used, and LSC would benefit from additional application examples that used a wider range of fitness functions.

A clear obstacle impeding the widespread use of nanosatellites for scientific exploration is the limited number of control algorithms and control actuators which have actually been implemented in this form factor. The controller presented in Chapter 4 directly accommodated for actuator saturation limits, however due its use of Modified Rodrigues Parameters it cannot be said to be Lyapunov stable in the state space of rigid body rotations. A valuable area of future research would be to extend controllers which operate on the full state space of $SO(3) \times \mathbb{R}^3$ to accomodate actuator saturation limits, and to expand the range of candidate actuator systems. A number of reaction wheels are available for use in CubeSats, and a few companies are now offering made-to-order magnetic torque coils. However, thruster systems remain too large to be easily incorporated into even a 6U spacecraft. Some promising work is being done at MIT and JPL regarding electrospray propulsion systems, but these remain at a relatively low level of technology readiness. Expanding the range of actuators that can be quickly plugged into CubeSats, along with controllers that can elegantly control them, would be a valuable contribution to the field.

The research presented in this dissertation focused on nanosatellites because they are a timely example of spacecraft that can be readily reconfigured. As described earlier the LSC approach is indifferent to the size of the spacecraft or the nature of the control law being utilized, and it would be interesting to study how the GA (or the other design methods) behaved when the same basic concepts are applied to a different spacecraft model. One nicety of nanosatellites is that they allow the user to assume a perfectly rigid body, an assumption that is not valid on spacecraft with deployable solar panels or other external structures. Flexible body motion greatly changes the dynamic response of the spacecraft, and it would be valuable to determine what impact, if any, this has on the ability of the GA to efficently find LSC design solutions. Similar search strategies as those described for sensor/actuator noise could be employed, in that the averages of multiple trial runs may prove to be the most instructive.

6.4 Concluding Remarks

The original motivation for this research was to help faciliitate a new class of scientific nanosatellites by changing the underlying approach to ADCS design. The unfortunate truth of the aerospace industry is that there are often forces acting on missions which cannot be modeled by an engineer, such as the consistency of funding sources or the whims of politicians. However, if one can build a spacecraft in a short period of time while maximizing the science return of that mission it may, over time, become possible to navigate around such exogenous inputs. This dissertation does not offer a concise proof of stability or optimality, but hopefully it has introduced a new concept which complements the emerging era of small, inexpensive access to space that is upon us.

Bibliography

[1] K. Abdel-Motagaly, B. Duan, and C. Mei. Active control of nonlinear panel flutter under yawed supersonic flow. In *Proc. of AIAA Conf. on Structures, Struct. Dyn., and Materials*, Norfolk, Virginia, April 2003.

[2] D. Acosta and S. Joshi. Adaptive nonlinear dynamic inversion control of an autonomous airship for exploration of Titan. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Hilton Head, South Carolina, August 2007.

[3] D. Bayley, R. Hartfield, J. Burkhalter, and R. Jenkins. Design optimization of a space launch vehicle using a genetic algorithm. *Journal of Spacecraft and Rockets*, 45(4):733–740, 2008.

[4] O. Brown and P. Eremenko. The value proposition for fractionated space architectures. In *Proc. of AIAA Space Conference*, San Jose, California, September 2006.

[5] F. Bullo and A. Lewis. *Geometric Control of Mechanical Systems*, pages 300–320. Springer, 2005.

[6] L. Crespo, M. Matsutani, and M. Annaswamy. Verification and tuning of an adaptive controller for an unmanned air vehicle. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Toronto, Canada, August 2010.

[7] L.C. Davis. Lateral restoring force on a magnet levitated above a superconductor. *Journal of Applied Physics*, 67(5):2631–2636, 1990.

[8] J. Fisher, R. Bhattacharya, and S.R. Vadali. Spacecraft momentum management and attitude control using a receding horizon approach. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Hilton Head, South Carolina, August 2007.

[9] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Sys., Man, and Cyber.*, 16(1):122–128, 1986.

[10] R.L. Haupt and S.E. Haupt. *Practical Genetic Algorithms.* John Wiley and Sons, New Jersey, 2004.

[11] J.M. Holland. *Adaptation in Natural and Artificial Systems*, pages 120–125. University of Michigan Press, 1975.

[12] B. Jackson and K. Epstein. A reconfigurable multifunctional architecture approach for next-generation nanosatellite design. In *Proc. of IEEE Aerospace Conference*, Big Sky, Montana, March 2000.

[13] T. Kane, P. Likins, and D. Levinson. *Spacecraft Dynamics*, pages 120–122. McGraw-Hill, first edition, 1983.

[14] K. Krishnakumar and D.E. Goldberg. Control system optimization using genetic algorithms. *Journal of Guidance, Control, and Dynamics*, 15(3):735–740, 1992.

[15] K. Krishnakumar, J. Kaneshige, R. Waterman, C. Pires, and C. Ippolito. A plug-and-play GNC architecture using FPGA components. In *Proc. of AIAA Infotech@Aerospace*, Arlington, Virginia, September 2005.

[16] T. Lee, N. H. McClamroch, and M. Leok. A Lie group variational integrator for the attitdue dynamics of a rigid body with applications to the 3D pendulum. In *Proc. of IEEE Conf. on Control Applications*, Toronto, Canada, August 2005.

[17] S. Mauthe, F. Pranajaya, and R. Zee. The design and test of a compact propulsion system for CanX nanosatellite formation flying. In *Proc. of AIAA/USU Conference on Small Satellites*, Logan, Utah, August 2005.

[18] M. Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson Education Ltd, London, England, 2002.

[19] N. Nise. *Control Systems Engineering*, pages 37–42. Space Technology Library, 2005.

[20] N. Nordkvist and A. Sanyal. A Lie group variational integrator for rigid body motion in SE(3) with applications to underwater vehicle dynamics. In *Proc. of IEEE Conference on Decision and Control*, Atlanta, Georgia, December 2010.

[21] R. Nugent. The CubeSat: The picosatellite standard for research and education. In *Proc. of AIAA Space Conference and Exhibit*, San Diego, California, September 2008.

[22] S. Ogata. *Modern Control Design*, pages 37–42. Space Technology Library, 2005.

[23] S. Omatu and M. Yoshioka. Self-tuning neuro-PID control and applications. In *Proc. of the IEEE Conf. on Sys., Man, and Cybernetics*, Orlando, Fl, October 1997.

[24] F. Peng, Y.R. Hu, and A. Ng. Testing of a membrane space structure shape control using a genetic algorithm. *Journal of Spacecraft and Rockets*, 43(4):788–793, 2006.

[25] W.M. Qi, W.Y. Cai, Q.L. Ji, and Y.C. Cheng. A design of nonlinear adaptive PID controller based on genetic algorithms. In *Proc. of the 25th Chinese Control Conference*, Harbin, China, August 2006.

[26] A. Rahmani, M. Mesbahi, and F. Hadaegh. Optimal balanced-energy formation flight maneuvers. *Journal of Guidance, Control, and Dynamics*, 29(6):1395–1403, 2006.

[27] D. Riddle, R. Hartfield, J. Burkhalter, and R. Jenkins. Genetic algorithm optimization of liquid-propellant missile systems. *Journal of Spacecraft and Rockets*, 46(1):151–159, 2009.

[28] R. Robinett, G. Parker, H. Schaub, and J. Junkins. Lyapunov optimal saturated control for nonlinear systems. *Journal of Guidance, Control, and Dynamics*, 20(6):1083–1088, 1997.

[29] R. Robinett, G. Wilson, G. Eisler, and J. Hurtado. *Applied Dynamic Programming for Optimization of Dynamical Systems*, pages 37–42. SIAM, 2005.

[30] M. Romano, D. Friedman, and T.J. Shay. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. *Journal of Spacecraft and Rockets*, 44(1):164–173, 2007.

[31] T. Sands. Physics-based automated control of a spacecraft. In *Proc. of AIAA Space Conference and Exhibition*, Pasadena, California, September 2009.

[32] A. Sanyal and N. Chaturvedi. Almost global robust attitude tracking control of spacecraft in gravity. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Honolulu, HI, August 2008.

[33] A. Sanyal and Z. Lee-Ho. Attitude tracking control of a small satellite in low earth orbit. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Chicago, IL, August 2009.

[34] A. Sanyal and N. Nordkvist. A robust estimator for almost global attitude feedback tracking. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Toronto, Canada, August 2010.

[35] A. K. Sanyal and N. Nordkvist. Attitude state estimation with multirate measurements for almost global attitude feedback tracking. *Journal of Guidance, Control, and Dynamics*, 35(3):868–880, 2012.

[36] P. Schonhuber and P.C. Moon. Levitation forces, stiffness, and force-creep in YBCO high-Tc superconducting thin films. *Applied Superconductivity*, 2(7):523–534, 1994.

[37] J. Shoer and M. Peck. Flux-pinned interfaces for the assembly, manipulation, and reconfiguration of modular space systems. *Journal of the Astronautical Sciences*, 57(3):667–688, 2009.

[38] J. Shoer and M. Peck. Simulation of multibody spacecraft reconfiguration through sequential dynamic equilibria. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Toronto, Ontario, August 2010.

[39] J. Shoer, W. Wilson, L. Jones, M. Knobel, and M. Peck. Microgravity demonstrations of flux pinning for station-keeping and reconfiguration of CubeSat-sized spacecraft. *Journal of Spacecraft and Rockets*, 47(6):475–483, 2010.

[40] M. D. Shuster. A survey of attitude representations. *Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[41] J.J. Slotine and W.P. Li. *Applied Nonlinear Control*, pages 37–42. SIAM, 2005.

[42] M. Sorgenfrei and S.S. Joshi. Further results towards a location-scheduled control methodology. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Portland, Oregon, August 2011.

[43] A.C. Stickler and K. Alfriend. Elementary magnetic control system. *Journal of Spacecraft and Rockets*, 13(5):282–287, 1976.

[44] D.A. Vallado and W.D. McClain. *Fundamentals of Astrodynamics and Applications*, pages 37–42. Space Technology Library, 2005.

[45] Wertz and Larson. *Space Mission Analysis and Design*, pages 410–412. D. Riedel Publishing Co., first edition, 1978.

[46] J. Wertz. *Spacecraft Attitude Determination and Control*, pages 200–223. D. Reidel Publishing, 1978.

[47] B. Wie. *Space Vehicle Dynamics and Control*, pages 450–457. AIAA, second edition, 2005.

[48] W. Wilson, J. Shoer, and M. Peck. Demonstration of a magnetic locking flux-pinned revolute joint for use on a CubeSat-standard spacecraft. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*, Chicago, Illinois, August 2009.