Closed-loop Control of Spacecraft Formations with Applications on SPHERES

by

Matthew M. Jeffrey

Bachelor of Science
Johns Hopkins University, 2006

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

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author .....
Department of Aeronautics and Astronautics
May 23, 2008

Certified by .....
Jonathan P. How
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Closed-loop Control of Spacecraft Formations with Applications on SPHERES

by

Matthew M. Jeffrey

Submitted to the Department of Aeronautics and Astronautics
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Formation flying satellites offer potentially greater science returns and operational capabilities than attainable with a monolithic spacecraft. Successful control of a formation of spacecraft can be divided into two separate stages. The first stage creates a plan that meets a set of mission objectives, and the second stage implements the plan. Plans are specified as a sequence of ΔV commands executed at specific times during an orbit. This thesis presents an online method for generating fleet-wide plans, using convex optimization techniques, that satisfy multiple objectives. The approach allows for minimum and balanced fuel usage, can position spacecraft in arbitrary configurations, and favors low-maintenance orbits that do not drift apart. Additionally, the architecture is applicable not only to formation-keeping maneuvers, but also to formation reconfigurations. Various simulations demonstrate the importance of accurately implementing plans for formation flying as well as autonomous rendezvous and docking missions. Specifically, the relationships between process error, overall fuel use, and position error are studied. Theory is put into practice with the development of a new low-level, closed-loop thrust controller for the Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES). The controller processes measurements from accelerometers and gyroscopes to monitor thruster performance in real-time. Experiments conducted on the International Space Station (ISS) validate the controller and establish a foundation for future enhancements to the underlying algorithm. Finally, data from a series of high-fidelity formation flying simulations is presented that confirms the analysis done elsewhere in the thesis. The multi-objective planner is used in a closed-loop control system that guides a formation of five spacecraft through a hypothetical mission involving both reconfigurations and formation-keeping. Data from the simulations allows a straightforward, side by side comparison of the effects and relative importance of sensor error versus implementation error.

Thesis Supervisor: Jonathan P. How
Title: Professor of Aeronautics and Astronautics

Acknowledgments

First I would like to thank my advisor, Prof. Jonathan How, for his guidance and keen insight, and for being patient whenever I was stuck on a problem or didn't know what step to take next. His ability to always ask the important questions was essential to completing this work. Thank you also to Kathryn Fischer for her administrative assistance, and for somehow providing a sense of order to everyone's hectic schedules. Thanks to Dr. Alvar Saenz-Otero and the entire SPHERES team (Amer, Brent, Chris, Christy, Jake, John and Swati) for letting an ACLer come onboard and feel welcome. Working with the satellites and participating in the ISS test sessions were highlights of my stay at MIT. A special thank you to Dr. Simon Nolet for his help with modifying the SPHERES estimator to work with my code. Thanks to my fellow labmates in the Aerospace Controls Lab for their creativity and intellectual energy, and thank you in particular to my officemate Georges Aoudé for his encouragement and good humor during the more difficult times. A significant portion of this thesis extends research by Dr. Louis Breger — without his time and unending willingness to help, it would not have been possible. Thank you, Louis. Thanks also to Adam Wahab for being a great friend and providing some much needed entertainment throughout the past two years, and of course to Erin, for putting up with the long work hours and occasional bout of grumpiness!

Finally, the deepest thanks to my parents and sister Kerri for their love and support. Any small success I have enjoyed, I owe to their example that hard work, persistence and dedication are rewarded.

This work was supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship, administered by the American Society for Engineering Education (ASEE). Research also funded in part by the Mitsubishi Electric Corporation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, formation flying satellites have emerged as one of the most promising technologies in space flight. A fleet of smaller, cheaper, less sophisticated satellites working together can replace a single, monolithic satellite while at the same time increasing flexibility and redundancy. Such a fleet is ideal for a variety of science purposes, including distributed sensor systems, simultaneous observations, and interferometry for Earth-observing or stellar telescopes. Moreover, there is the option of adding or replacing individual spacecraft during the mission to enhance the capabilities of the fleet or even create new ones. Although multi-spacecraft interferometers were suggested as early as the late 1970s [1], only the advances in estimation and control of the last several years have brought about the capability to realize such formations. As a result, there has been a surge in missions (both flown and proposed) utilizing formations of spacecraft.

NASA's Earth-Observing 1 successfully flew in formation with Landsat 7 in 2001, repeating the latter's ground track at a separation of about 450 km. The experiment was a demonstration of some of the basic technologies required for future formation flying missions [2]. In 2006, ST5, a formation of three micro-satellites and part of NASA's New Millennium Program, successfully provided simultaneous, multi-point measurements of the Earth's magnetic field [3]. The A-Train constellation, an international collaboration between NASA and CNES, currently consists of five satellites with plans to add two more. The satellites fly in an along-track separation within sev-

eral minutes of each other and study clouds and precipitation [4]. Each spacecraft in the A-Train has its own instrumentation, but by combining the diverse measurements, scientists are learning new things about cloud formations.

While these missions have been impressive successes, space-based interferometry promises even greater science returns. Agencies worldwide are currently planning missions like Darwin [5], the Terrestrial Planet Finder (TPF) [6], the Laser Interferometer Space Antenna (LISA) [7], the X-ray Evolving Universe Spectroscopy (XEUS) [8], and the Micro-Arcsecond X-ray Imaging Mission (MAXIM) [9], to harness this potential. LISA will search for low-frequency gravitational radiation characteristic of black holes, while Darwin and TPF scan the stars for Earth-like planets that might support life. These missions utilize two, three or even eight spacecraft and present challenges more difficult than those that have thus far been met on orbit. For that reason, a separate crop of formation flying demonstration missions seeks to test and validate the algorithms required for maintaining strict inter-spacecraft geometry requirements. PROBA 3, by ESA [10] and PRISMA [11], from the Swedish Space Corporation, are a pair of two satellite missions that will test formation flying sensing and control technologies.

Autonomous rendezvous and docking is another research area that generates a lot of interest. The Russians have been docking autonomously with their Progress and Soyuz spacecraft since Mir [12], and in 1998, the Japanese ETS-VII mission performed an autonomous docking between two unmanned spacecraft [13]. Most recently, ESA's Jules Verne [14], the first of five planned automated transfer vehicles (ATVs), successfully docked with the International Space Station (ISS) after performing a series of checkout and demo maneuvers. This type of vehicle, with autonomous rendezvous and docking capabilities, plays an important role in NASA's post-shuttle plans for the ISS (*e.g.,* the Commercial Orbital Transportation Services, or COTS, program [15]). Despite these successes, it would be premature to call AR&D routine, as recent missions have shown that serious failures can occur. In 2005, DART [16] experienced navigational difficulties that caused it to fly into directly into its target satellite, MUBLCOM, despite believing it was 130 m away [17]! Two years later, in

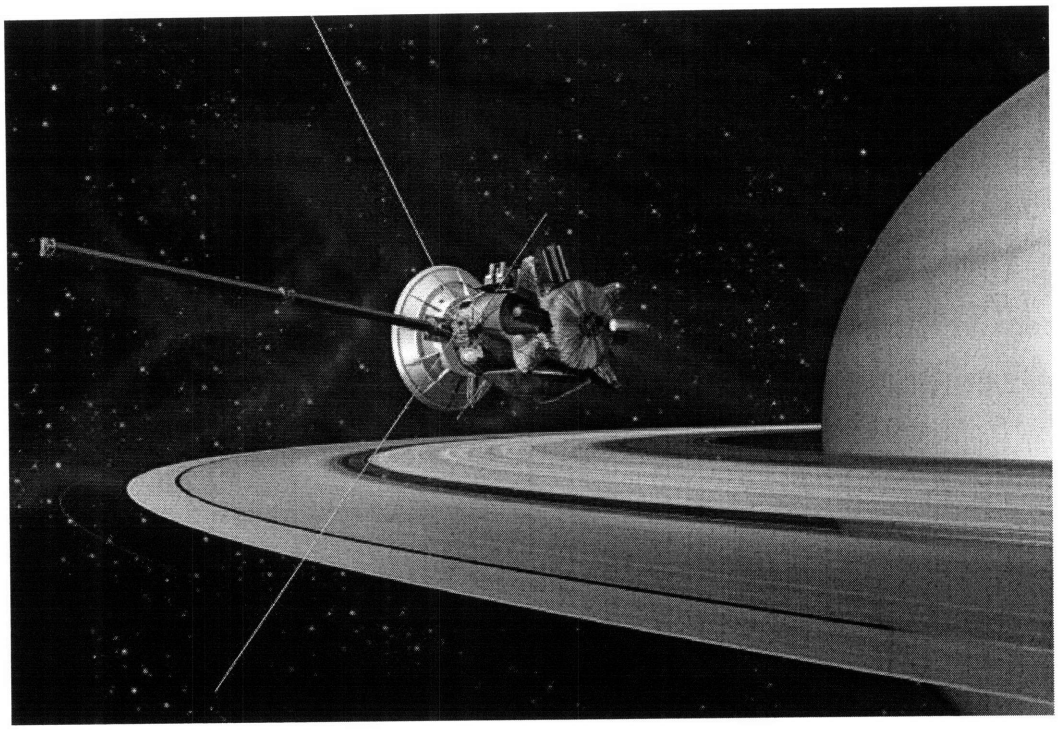


Fig. 1-1: Artist's impression of the Cassini SOI (Courtesy NASA/JPL-Caltech).

2007, the Orbital Express mission successfully demonstrated many of the technologies first attempted with DART [18].

Formation flying and rendezvous and docking missions share a common need for accurate sensing and implementation of ΔV commands. This thesis studies the effects of inaccurate ΔV implementation on rendezvous scenarios as well as formation flying missions. Specifically, it explores the use of accelerometers to improve the ΔV implementation.

1.1 Previous Work

Numerous authors have proposed control algorithms for formation flying missions [1]. Many of these are based on the Hill-Clohessy-Wiltshire (HCW) equations [19, 20] for circular orbits, or Lawden's time-varying equations [21] for elliptical orbits. These descriptions are convenient for computing the relative motion of satellites; the HCW equations are linear time-invariant, and Lawden's equations are linear time-varying. They have also been modified to include the dominant effects of the Earth's oblate-

ness [22]. Finally, both of these dynamics models have been used in PD [23], LMI [24], LQR, nonlinear [25], impulsive [26, 27], and model predictive control (MPC) [28] control designs.

Unfortunately, these approaches use Cartesian coordinates in an LVLH frame that limits their accuracy to formations with short baselines, typically less than 1 km. Breger and How showed in [29] that by using Gauss's variational equations (GVEs), where the dynamics are expressed in relative orbital elements, formations with much longer baselines, such as MMS [30], can be successfully controlled using MPC. The longer baselines are possible because the GVEs describe satellite motion in a curvilinear frame, where long linear distances are described by small perturbations in the orbital elements [31]. Breger's MPC uses linear programs (LPs) [32] and a state transition matrix incorporating J_2 effects developed by Gim [33]. Elsewhere in the literature, GVEs have been used in other control schemes, such as continuous-thrust [34] and impulsive controllers [26, 35, 36].

This thesis builds on the previous research on MPC control of formations [23, 28, 29, 37–41]. Linear programs are convenient because they are fast and reliable, and can incorporate realistic constraints (actuator bounds, thrusting restrictions, position error boxes) that are difficult to model with other control techniques [28]. The idea was proposed in [23] as a possible control approach for the ORION formation flying demonstration mission that was being planned at that time [42]. The concepts were significantly expanded in [28] and [39], where a number of realistic constraints were modeled inside the LP framework. With any formation, a primary goal will be to prevent the spacecraft from drifting apart by initializing them to *invariant* orbits. References [43, 44] present approaches for achieving J_2 invariance based on solving necessary conditions for partial invariance of the mean orbital elements. These methods minimize secular drift in the mean orbital elements, but contain unspecified degrees of freedom. Work by Breger and How [29, 41] introduced an optimization-based approach for finding invariant orbits that can minimize more general definitions of drift (*e.g.,* Cartesian separation, osculating states) over arbitrary time frames while minimizing fuel and considering performance objectives. This approach yields results

that can be used online to optimize and re-optimize initial conditions for Earth-orbiting formation flying missions that require drift to be minimized, but that also have particular geometry requirements on separation distance or shape.

Sensor noise has long been known as a driver of system performance for spacecraft formations [37, 45]. Tillerson investigated the impact of sensing noise for formations in circular orbits in [39]. His work was based on, at that time, a state-of-the-art CDGPS navigation filter developed by Busse [46] that provided relative position measurements with an accuracy of about 1 cm and relative velocity measurements within 0.5 mm/s. Advances in filter design and technology place current estimates of attainable accuracy at 1.5 mm for relative position, and 5 μm/s for relative velocity [47]. The drastic reduction in sensing uncertainty has allowed other sources of error to limit performance, specifically the thruster system [36]. The implementation of ΔV commands is an essential function of any spacecraft, whether it is part of a formation or not. Inertial measurement units (IMUs), fairly standard on modern spacecraft, usually consist of gyroscopes and accelerometers and assist in this function. The ATV that services the ISS uses three two-axis accelerometers and four two-axis gyrometers [48]. The Cassini spacecraft used accelerometer measurements to perform the mission-critical Saturn Orbit Insertion [49, 50]. Likewise, the Deep Impact mission to "rendezvous" with the comet Tempel I relied on accelerometer measurements to autonomously correct its final approach trajectory [51]. The trend will continue into the foreseeable future, with one example being the control system for the upcoming Terrestrial Planet Finder. TPF will use accelerometers and gyros to accurately implement ΔV commands [52].

1.2 Thesis Overview

Chapter 2 extends the GVE-based MPC in [29] from a single spacecraft to a fleet-wide optimization, and a series of new constraints and performance objectives are introduced that expand the controller's capabilities. The initial state of the formation is separated from the desired state so that the MPC can be used for reconfiguration

problems in addition to formation-keeping problems. Additionally, rather than specifying all spacecraft positions with respect to a reference orbit, a new way to specify relative positions, based on the leader-follower strategy, is derived. The issue of balancing fuel use is also addressed, and a series of constraints designed to encourage this desirable behavior are presented. Example plans for spacecraft formations are generated and analyzed to demonstrate the performance of the controller.

Maneuver plans designed by the MPC in Chapter 2 are output as a list of ΔV commands that must be executed at specific times. Chapter 3 tackles the problem of actually implementing the ΔV commands that make up the plans. Two approaches are considered: an open-loop system where thrust durations are calculated from a thruster model, and a closed-loop system that monitors the performance of the burn in real-time and uses feedback to determine when to terminate it. The dangers of the open-loop strategy are discussed, and the closed-loop system using accelerometers is proposed as a way to increase burn accuracy. After developing the theory for the closed-loop system, the two approaches are compared side by side through a series of simulations. The performance degradation that results when plans are poorly implemented is quantified for a rendezvous scenario (in LEO with an orbit similar to that of the ISS). Two methods are identified for improving the performance for such a mission; more accurate plan implementation, or replanning. The performance tradeoffs for each are explored with an emphasis on the advantages of better plan implementation. After the rendezvous scenario, the fleet-wide planner introduced in Chapter 2 is revisited to confirm the importance of accurately following its plans.

The ideas of Chapter 3 are put into practice through the development of a new low-level controller for the SPHERES satellites in Chapter 4. This chapter walks through the design of the control system and documents how various problems encountered during the process were solved. Two different algorithms for measuring thruster performance in real-time during a burn are proposed. The first utilizes low-pass digital filters to process accelerometer measurements and integrate them directly. Test results from experiments on a 2D air table as well as on orbit testing from the ISS then form the basis for an improved algorithm using least-squares estimation

techniques.

Formation planning and accurate ΔV implementation concepts are unified in Chapter 5 through a series of high-fidelity, realistic formation flying simulations. Based on performance requirements that are representative of actual missions, the planner of Chapter 2 is integrated with a commercial orbit propagator to control a fleet of spacecraft over dozens of orbits. Plans for reconfigurations and formation-keeping are both generated and executed, and data on the impact of poor ΔV implementation is analyzed in terms of fuel use and position error. This analysis is repeated for perfect implementation with varying levels of sensor error, and comparisons are made between the two types of error. The simulations underscore the benefits of improving the accuracy of ΔV implementation.

Finally, Chapter 6 closes the thesis with a recap of the important findings, and concludes with suggestions for future work.

Chapter 2

Multi-Objective Planning for Spacecraft Formations

Spacecraft formation flying is expected to be an enabling technology for many future missions [53]. Successful control planning for a formation must consider a range of competing performance objectives and find acceptable compromises that meet mission requirements. Through collaboration and cooperation, the spacecraft in a formation act as a single system that offers better performance than is attainable with a monolithic spacecraft. Performance enhancements include superior science observations with space-based interferometry, cheaper launch costs spread out over smaller, lighter vehicles, or more versatility with a fractionated spacecraft design. The planning tasks for a formation can be divided into two classes [39]: *formation-keeping* and *formation reconfiguration.* In formation-keeping, the spacecraft are already in their desired configuration, and positions and velocities are simply maintained by applying control to counteract disturbances (which manifest as drift). In formation reconfiguration, the desired positions and velocities change, and trajectories must be generated to relocate the spacecraft accordingly. Both classes are important and will be addressed here.

Formation flying missions frequently have several different operating configurations. For example, the Magnetospheric Multiscale Mission (MMS) will consist of four spacecraft in a tetrahedral formation studying plasma dynamics in the Earth's

magnetosphere [30, 54], but the separation distance between spacecraft will vary from 10 km to 1000 km depending on the phenomena currently being observed. When reconfiguring, it is critically important both to conserve fuel during the maneuver and to maneuver to a state that will, over time, conserve fuel. The selection of states with these properties is called an initial condition (IC) problem, the specifications of which will depend upon the unique requirements of a particular mission. However, in any spacecraft formation, a primary goal will be to prevent the vehicles from drifting apart, since that will typically end the mission. If the spacecraft in a formation tend to drift apart, then periodic maintenance maneuvers will be required during formation-keeping to restore the formation. Initial conditions are called *invariant* if they eliminate drift, thereby allowing spacecraft to maintain their relative orbits without expending fuel. Invariant orbits simplify the formation-keeping process. Two expected sources of relative drift for an Earth-orbiting formation are mismatched semimajor axes [55] and J_2 disturbances. Finally, for space-based interferometers, such as TPF [6], MAXIM [9], or LISA [7], loss of control of one spacecraft cripples the science output for the entire formation. Since a spacecraft would become uncontrollable if its fuel supply were exhausted, fuel use should be balanced across the formation.

This chapter presents an online method that meets the challenges outlined above; it approaches planning from a fleet-wide perspective and optimizes fuel usage, fuel balancing, formation geometry, and drift. Online methods are desirable because they help minimize the delay between the detection of disturbances and their correction, enabling more precise formation flying. The method is an extension of the model predictive controller in [29], with more flexible performance objectives, but its roots can be traced even further back to ideas in [23, 28, 40]. Separation of the desired geometry from the starting geometry enables reconfiguration maneuvers in addition to formation-keeping. Furthermore, the optimization problem is expanded to include all spacecraft in the formation, permitting the introduction of relative geometry constraints — a prerequisite for implementing the popular leader-follower approach. The planner optimizes formation-keeping and reconfiguration maneuvers not just for a sin-

gle spacecraft at a time, but for the entire formation simultaneously, enabling true formation flying via extensive cooperation [38].

2.1 Formation Optimization

Reference [41] derived a single spacecraft IC optimization based on Gauss' Variational Equations (the GVEs derived in [56]) modified to include the effects of J_2. The GVEs describe the motion of an orbit (expressed in absolute osculating elements) subject to perturbations. A reference orbit is taken as

$$\mathbf{e}_{ref} = \begin{bmatrix} a & e & i & \Omega & \omega & M \end{bmatrix} \tag{2.1}$$

where a is the semimajor axis, e is the eccentricity, i is the inclination, Ω is the right ascension of the ascending node, ω is the argument of perigee, and M is the mean anomaly. The relative position of a spacecraft (with absolute state $\mathbf{e}$) is defined with respect to the reference orbit as

$$\delta\mathbf{e} = \mathbf{e} - \mathbf{e}_{ref} \tag{2.2}$$

Using the GVEs and (2.2), Breger [41] derives the optimization

$$C^* = \min_U \{Q_d \|W_d(\bar{\Phi}^*_{D1} - I)(\bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) + \hat{\Gamma}U)\| + Q_x\|W_x\hat{\Gamma}U\| + Q_u\|U\|\} \tag{2.3}$$

where C^* is the optimal cost, U is a column vector of control inputs, $\delta\mathbf{e}$ is a differential column vector of osculating orbital elements, $\hat{\Gamma}$ is a discrete convolution matrix, and the weighting matrices W_d and W_x specify the form of the penalties for drift and formation geometry, respectively. The scalar weights Q_d, Q_x, and Q_u determine the penalties on drifting, formation geometry, and fuel use. The state transition matrix, $\bar{\Phi}^*_{Dn}$, propagates the differential state vector from time t_n to time t_{n+1}. Additionally, t_0 is the time at the start of the optimization, t_1 is the time at the end of the initialization maneuver, and $[t_1, t_2]$ is the period over which drift is minimized (usually

one orbit). All norms in these optimizations are one-norms. The control sequence $U(\tau),\ \tau \in [t_0, t_1]$ is of the form

$$U = \begin{bmatrix} \mathbf{u}_0^T & \mathbf{u}_1^T & \cdots & \mathbf{u}_n^T \end{bmatrix}^T \tag{2.4}$$

where n is the number of steps in the plan. Moreover, each entry $\mathbf{u}_k$ is a three-element vector

$$\mathbf{u}_k = \begin{bmatrix} u_{k_x} & u_{k_y} & u_{k_z} \end{bmatrix}^T \tag{2.5}$$

where the subscripts x, y, and z denote the radial, in-track, and cross-track directions in the local vertical/local horizontal (LVLH) frame. Since control is not allowed during the drift period, a good solution to the optimization problem must balance the desire to achieve low-drift initial conditions at t_1 with the desire to preserve formation geometry and conserve fuel. The individual elements of the cost function (2.3) are written as C_d (drift), C_x (geometry), and C_u (fuel).

2.1.1 Separating the Initial and Desired States

The geometry cost in (2.3) is given by

$$C_x = Q_x \| W_x \hat{\Gamma} U \| \tag{2.6}$$

Inspection of (2.6) shows that the cost will increase with the input effect of the control, $\hat{\Gamma}U$. In other words, if the applied control causes the spacecraft to move to a new position, it will incur a positive cost. The only way C_x is zero is if the overall effect of U is to not move the spacecraft at all. Therefore, the optimization presented in Ref. [41] tacitly assumes that the initial location of the spacecraft is the desired location. This limits its usefulness to the formation-keeping case, where the formation is already initialized and the main concern is to find a minimum-drift orbit; it does not allow for a formation to be reconfigured.

This limitation can be overcome by modifying the optimization in (2.3) to separate the initial state, $\delta\mathbf{e}(t_0)$, from the desired state. The vector $\delta\mathbf{e}_d(t_1)$ is introduced as

the desired state at the end of the initialization maneuver, and the difference between the desired state and the actual state at t_1 is penalized

$$C_x = \|W_x(\delta\mathbf{e}_d(t_1) - \delta\mathbf{e}(t_1))\| \tag{2.7}$$

Substituting $\delta\mathbf{e}(t_1) = \bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) + \hat{\Gamma}U$ gives a new geometry cost

$$C_x = \|W_x(\delta\mathbf{e}_d(t_1) - \bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) - \hat{\Gamma}U)\| \tag{2.8}$$

making the altered optimization

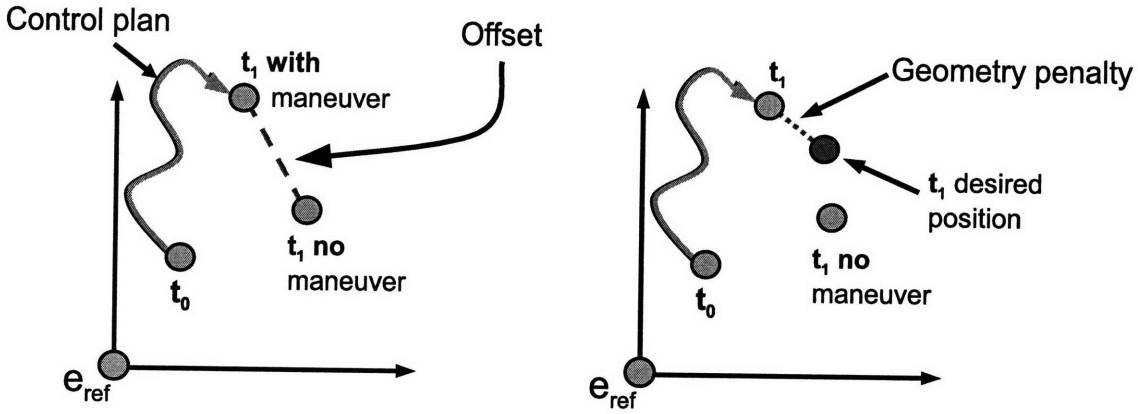
$$\begin{aligned} C^* = \min_U \ \{&Q_d\|W_d(\bar{\Phi}^*_{D1} - I)(\bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) + \hat{\Gamma}U)\| + \\ &Q_x\|W_x(\delta\mathbf{e}_d(t_1) - \bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) - \hat{\Gamma}U)\| + Q_u\|U\|\} \end{aligned} \tag{2.9}$$

The independence of the starting position, $\delta\mathbf{e}(t_0)$, from the desired position, $\delta\mathbf{e}_d(t_1)$, provides the capability to design trajectories that deploy the spacecraft to some new orbit, while still minimizing drift and fuel use. This expands the capability of the controller to reconfiguration problems. Functionality has only been added, not lost; if the desired position at t_1 is equal to the open-loop propagation

$$\delta\mathbf{e}_d(t_1) = \bar{\Phi}^*_{D0}\delta\mathbf{e}(t_0) \tag{2.10}$$

then the cost (2.9) is equivalent to (2.3). The new formulation is also useful for implementing the optimization as a closed-loop controller, because in practice, the spacecraft will never exactly be in their desired position, due to thruster and sensor error.

Figure 2-1(a) illustrates the previous geometry cost associated with (2.3) and is adapted from [57]. The initial position of the spacecraft at t_0 is marked by a circle, as well as the open-loop (no maneuver) position at t_1 and the actual position at t_1 (which depends on the control plan). The dashed line indicates the quantity that is penalized. Henceforth, it is called the "offset" cost since it is proportional to the



(a) Offset Cost

(b) Geometry Cost

Figure 2-1: Comparison of offset cost and geometry cost.

magnitude of the offset caused by the control input. The offset cost is not used from this point on, and should not be confused with the new "geometry" cost, described by the modifications in (2.9) and reflected in Figure 2-1(b). The key difference is the new desired state at t_1, and the cost is now proportional to the spacecraft's distance from this desired state. The open-loop propagation to t_1 no longer plays a role in the cost assessment, and is irrelevant.

2.1.2 Extension to the Multi-Vehicle Case

Previously, using the general strategy presented here for formation flight consisted of combining separate solutions to the single spacecraft optimization in (2.3) [41]. While this has the advantage of keeping the problem size small (fewer decision variables and constraints), it is unlikely that implementing independent solutions for each spacecraft will result in an optimal overall plan. For example, sometimes it may be advantageous for one spacecraft to sacrifice individual optimality for global optimality (*e.g.*, one spacecraft uses more fuel so that all of the others can use less). To extend the formulation to the multi-vehicle case, introduce the formation state $\delta\tilde{\mathbf{e}}$, a desired

formation state $\delta\tilde{\mathbf{e}}_d$, and a formation plan $\tilde{U}$ as

$$\delta\tilde{\mathbf{e}} = \begin{bmatrix} \delta\mathbf{e}_1^T & \delta\mathbf{e}_2^T & \cdots & \delta\mathbf{e}_N^T \end{bmatrix}^T, \quad \delta\tilde{\mathbf{e}}_d = \begin{bmatrix} \delta\mathbf{e}_{d_1}^T & \delta\mathbf{e}_{d_2}^T & \cdots & \delta\mathbf{e}_{d_N}^T \end{bmatrix}^T,$$
$$\tilde{U} = \begin{bmatrix} U_1^T & U_2^T & \cdots & U_N^T \end{bmatrix}^T \tag{2.11}$$

where N is the number of spacecraft in the formation, $\delta\mathbf{e}_i$ is the state of the i^{th} spacecraft in the formation, $\delta\mathbf{e}_{d_i}$ is the desired state of the i^{th} spacecraft, and U_i is a vector of control inputs for the i^{th} spacecraft. Similarly, define the formation-wide propagation matrix from t_i to t_{i+1} as $\tilde{\Phi}_i$ and a formation-wide discrete convolution matrix $\tilde{\Gamma}$ as

$$\tilde{\Phi}_i = \begin{bmatrix} \bar{\Phi}_{Di}^* & 0 & \cdots & 0 \\ 0 & \bar{\Phi}_{Di}^* & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \bar{\Phi}_{Di}^* \end{bmatrix} \quad \text{and} \quad \tilde{\Gamma} = \begin{bmatrix} \hat{\Gamma} & 0 & \cdots & 0 \\ 0 & \hat{\Gamma} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \hat{\Gamma} \end{bmatrix} \tag{2.12}$$

Finally, the multi-spacecraft weighting matrices are

$$\tilde{W}_d = \begin{bmatrix} W_d & 0 & \cdots & 0 \\ 0 & W_d & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & W_d \end{bmatrix} \quad \text{and} \quad \tilde{W}_x = \begin{bmatrix} W_x & 0 & \cdots & 0 \\ 0 & W_x & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & W_x \end{bmatrix} \tag{2.13}$$

Combining (2.11)–(2.12) yields the state of the entire formation at time t_1 given the state at t_0

$$\delta\tilde{\mathbf{e}}(t_1) = \tilde{\Phi}_0\delta\tilde{\mathbf{e}}(t_0) + \tilde{\Gamma}\tilde{U} \tag{2.14}$$

The propagation in (2.14) and the formation weighting matrices in (2.13) can be combined to form a full formation initialization optimization based on (2.9)

$$\tilde{C}^* = \min_{\tilde{U}} \{Q_d\|\tilde{W}_d(\tilde{\Phi}_1 - I)(\tilde{\Phi}_0\delta\tilde{\mathbf{e}}(t_0) + \hat{\Gamma}\tilde{U})\| + \tag{2.15}$$
$$Q_x\|\tilde{W}_x(\delta\tilde{\mathbf{e}}_d(t_1) - \tilde{\Phi}_0\delta\tilde{\mathbf{e}}(t_0) - \tilde{\Gamma}\tilde{U})\| + Q_u\|\tilde{U}\|\}$$

Solving (2.15) yields the optimal formation-wide initialization cost, $\tilde{C}^*$, and produces a set of optimized control inputs for each spacecraft in the formation. These control inputs, in turn, specify the trajectories each spacecraft should follow to achieve the initial conditions. In contrast to the analytic methods for finding invariant orbits that are available in the literature, this new approach to finding formation ICs can be used to minimize drift globally across an entire formation of spacecraft.

2.1.3 Relative Position Cost

The primary reason for converting to a formation-wide optimization is to provide a method that can handle coupling between the spacecraft. Coupling occurs in the traditional leader-follower architecture, where one spacecraft is designated the "leader", and the others in the formation are its "followers". Position and velocity constraints for the followers are specified relative to the leader. If the leader spacecraft executes a maneuver, then the other spacecraft in the formation are affected; hence, there is coupling. Simple examples of a leader-follower are EO-1 [2], which followed one minute behind the ground track of Landsat-7, and XEUS [8], where the X-ray science instrument is split between two spacecraft. Moreover, the geometry requirements for space-based interferometers such as TPF [58] and Darwin [5] are typically specified in the relative frame. The cost function (2.15) only penalizes geometry deviations from a static reference orbit; penalizing relative geometry requires the trajectories to be optimized jointly because a variation in the trajectory of one spacecraft will require an adjustment in the trajectory of the other. For leader-follower missions and interferometers, it might be advantageous for the formation to stray some distance away from the reference orbit, as long as strict relative separations are maintained. Allowing this flexibility can result in reconfigurations that save fuel or reduce drift. Relative position constraints address this issue.

Figure 2-2 illustrates the relative position cost. The new idea is that the origin of the coordinate system is fixed to a leader spacecraft, not a reference orbit. Any spacecraft in the formation can be selected as the leader, and its differential state at

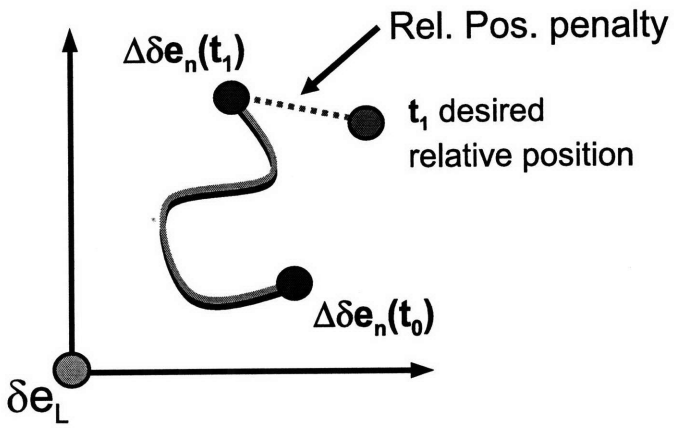


Figure 2-2: Relative position cost.

t_0 is $\delta\mathbf{e}_\ell(t_0)$. The relative position of the i^{th} spacecraft is then given by

$$\Delta\delta\mathbf{e}_i(t) = \delta\mathbf{e}_i(t) - \delta\mathbf{e}_\ell(t) \tag{2.16}$$

In this notation, the Δ signifies a relative position. Using (2.16) and the single-spacecraft equivalent of (2.14) yields the desired relative position at t_1

$$\Delta\delta\mathbf{e}_{d_i}(t_1) = \bar{\Phi}^*_{D0}(\delta\mathbf{e}_i(t_0) - \delta\mathbf{e}_\ell(t_0)) + \Gamma(U_i - U_\ell) \tag{2.17}$$

where U_ℓ is the control plan of the leader. To penalize the deviation from the desired relative position at t_1, $\Delta\delta\tilde{\mathbf{e}}_d(t_1)$ and $\Delta\delta\tilde{\mathbf{e}}(t_1)$ are constructed in the same manner as (2.11) and a new term, $\tilde{C}_r$, is appended to the cost function:

$$\tilde{C}_r = Q_r \left\| W_r \left(\Delta\delta\tilde{\mathbf{e}}_d(t_1) - \Delta\delta\tilde{\mathbf{e}}(t_1)\right)\right\| \tag{2.18}$$

In this cost, Q_r is a scalar weight specifying the penalty on relative position error, and W_r is a weighting matrix specifying the form of this error. Until now, geometry costs have been assessed in differential osculating orbital elements (2.2). As shown in [41], choosing W_r as the transformation matrix from differential osculating orbital elements to the LVLH frame allows the geometry to be penalized in a Cartesian sense (rectilinear distances as opposed to the curvilinear distances for osculating elements). The

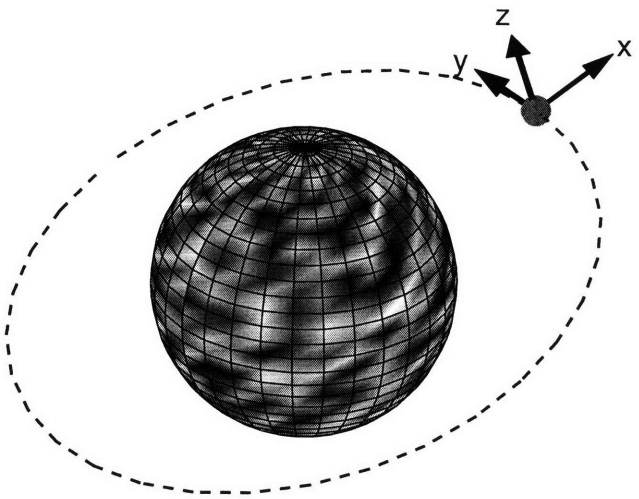


Figure 2-3: Hill's Frame (Local Vertical/Local Horizontal).

LVLH frame is more intuitive for visualizing formations and assessing performance because distances are specified in meters rather than orbital elements. Figure 2-3 depicts the LVLH frame for an orbiting satellite. The transformation matrix $M(\mathbf{e}(t))$ is defined in Ref. [59] such that

$$M\mathbf{x} = \delta\mathbf{e}_{\text{osc}} \quad \rightarrow \quad \mathbf{x} = M^{-1}\delta\mathbf{e}_{\text{osc}} \tag{2.19}$$

and it rotates

$$\mathbf{x} = [\ x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z}\ ]^T \tag{2.20}$$

from the LVLH frame centered on the reference orbit to differential osculating elements.

Although the form of $\tilde{C}_r$ is similar to $\tilde{C}_x$ in (2.15), the matrices in (2.12) must be slightly modified to incorporate the new coupling effects. Subtract $\bar{\Phi}^*_{Di}$ from the column of the formation propagation matrix corresponding to the leader, and subtract $\hat{\Gamma}$ from that same column in the discrete convolution matrix for the full formation. Without loss of generality, let $i = 1$ be the formation leader. The new relative

formation matrices, denoted by the subscript r, then become

$$\tilde{\Phi}_{r_i} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -\bar{\Phi}^*_{Di} & \bar{\Phi}^*_{Di} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ -\bar{\Phi}^*_{Di} & \cdots & 0 & \bar{\Phi}^*_{Di} \end{bmatrix} \quad \text{and} \quad \tilde{\Gamma}_r = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -\hat{\Gamma} & \hat{\Gamma} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ -\hat{\Gamma} & \cdots & 0 & \hat{\Gamma} \end{bmatrix} \tag{2.21}$$

The cancelations in the first row results in a row of zeros that automatically discards the position constraint of the leader relative to itself. In fact, the row can be omitted entirely; the net result is an additional $N-1$ constraints. The new cost, $\tilde{C}_r$, is adjoined to the global optimization, yielding the cost function

$$\begin{aligned} \tilde{C}^* = \min_{\tilde{U}} \{ & Q_d \|\tilde{W}_d(\tilde{\Phi}_1 - I)(\tilde{\Phi}_0 \delta\tilde{\mathbf{e}}(t_0) + \hat{\Gamma}\tilde{U})\| + \\ & Q_x \|\tilde{W}_x(\delta\tilde{\mathbf{e}}_d(t_1) - \tilde{\Phi}_0 \delta\tilde{\mathbf{e}}(t_0) - \tilde{\Gamma}\tilde{U})\| + \\ & Q_r \|\tilde{W}_r(\Delta\delta\tilde{\mathbf{e}}_d(t_1) - \tilde{\Phi}_{r_0} \delta\tilde{\mathbf{e}}(t_0) - \tilde{\Gamma}_r\tilde{U})\| + Q_u\|\tilde{U}\| \} \end{aligned} \tag{2.22}$$

2.2 Fuel Balancing

When a satellite in a formation runs out of fuel, there are several possible responses. If the formation is fully redundant, then perhaps there is not any discernable effect on operations and no action is required. Otherwise, the formation might continue to function in a reduced capacity. Another option is to replace the satellite at some cost. In the worst case, the satellite that runs out of fuel is irreplaceable or mission critical, and the mission ends. Regardless, it is beneficial to consider each satellite's remaining fuel supply when planning maneuvers and trajectories. For many missions it is desirable that each satellite in the formation expends fuel at roughly the same rate to avoid the aforementioned scenarios. For example, suppose a formation consists of two interdependent satellites (Sat A and Sat B) with an initial in-track separation of 1 km. They are tasked with a mission that requires them to reconfigure to 100 m relative separation. If both satellites begin the maneuver with the same amount of

fuel, the best solution is for each to move 450 m towards the other, and meet in the middle. However, if Sat A has more fuel than Sat B, then Sat A should travel further than Sat B. This will help balance the fuel between the satellites and prolong the mission life. Sat A sacrifices *local* optimality in favor of *global* optimality; it uses more of its own fuel, but improves the overall fuel management of the formation.

In practice, questions of fuel balancing are likely to be much more complex, but the ideas remain the same. If the formation is specified with respect to a reference orbit, then care must be taken that the reference orbit accurately describes the mean motion of the fleet. Because it is a theoretical construct, the reference orbit is not subjected to the disturbances experienced by the actual spacecraft. If these disturbances are inaccurately modeled when propagating the reference orbit, then fuel will be wasted trying to track a poor description of the formation's motion [38]. The traditional leader-follower approach eliminates this shortcoming by using the leader spacecraft as the reference point. However, by definition, the leader spacecraft has no state error and will use less fuel than the followers. Beard considered the problem of rotating a formation of spacecraft in free space [60], noting that if the formation rotates about a spacecraft, then that spacecraft will consume much less fuel than the others. Therefore, the starting fuel distribution should determine the optimal point of rotation. Later, Rahmani, Mesbahi, and Hadaegh considered the problem of balancing energy consumption for formations of two and three spacecraft [61]. Their approach was tested on problems in free space as well as for circular orbits (using Hill's equations), but the control solutions involve continuous thrust, which may be undesirable. Vadali, Vaddi and Alfriend also derived a continuous controller for maintaining a formation in a circular orbit, but using Hill's equations modified to include J2 [62]. Each of these approaches are valid only for specific classes of orbits.

Unfortunately, the minimization of fuel for one spacecraft and the balancing of fuel use across a formation are in general, competing objectives. The *virtual structure* attempts to solve this problem by treating the formation as a single entity, allowing it to evolve in a given direction as a whole [63]. Tillerson, Breger, and How derive the *virtual center* in [38], a reference point from which the desired positions of the space-

craft are specified. It is computed from the relative states and weighted according to fuel remaining. This biases the virtual center towards spacecraft with less fuel, and yields improved fuel balancing during formation-keeping for sparse aperture formations. The virtual center is computed strictly from the current formation geometry and does not depend on the dynamics of the orbit.

The following sections propose several ways to introduce fuel balancing constraints into the planner, presented in order of increasing generality. Used in conjunction with relative position constraints, it shares some similarities to the virtual center approach for formation-keeping. However, the need for an explicit calculation of the virtual center is eliminated; the optimal point will be found by the optimization itself. Moreover, since the fuel balancing constraints are implemented along with all of the other constraints (minimum fuel, position, and drift), the dynamics of the system are embedded (contrary to the virtual center approach). Overall, this allows fuel balancing to also be used for formation reconfigurations, and it is not limited to circular orbits.

2.2.1 Equal Fuel Use

One method for equalizing fuel use across the fleet is to penalize the deviation of each spacecraft's fuel use from the average. Fuel consumed is equivalent to the one-norm of the control inputs for each spacecraft. For a plan with n steps, the fuel use of the i^{th} spacecraft is given by:

$$\sum_{j=1}^{n} \|\mathbf{u}_{i,j}\| \tag{2.23}$$

where $\mathbf{u}$ is defined as in (2.5) and the second subscript, j, corresponds to a specific input step in the plan. For a formation of N spacecraft, the average fuel use is then

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n} \|\mathbf{u}_{i,j}\| \tag{2.24}$$

The equal fuel use fuel-balancing cost associated with the k^{th} spacecraft is the norm of the difference between its fuel use (2.23) and that of the average (2.24)

$$C_{k,f_{use}} = \left| \sum_{j=1}^{n} \|\mathbf{u}_{k,j}\| - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n} \|\mathbf{u}_{i,j}\| \right| \tag{2.25}$$

which can be simplified by defining $v_i = \|\mathbf{u}_i\|$ and $V_i = \|U_i\|$ so that

$$\tilde{V} = \begin{bmatrix} \|U_1\| & \|U_2\| & \dots & \|U_N\| \end{bmatrix}^T \tag{2.26}$$

With these substitutions, (2.25) is equivalent to

$$C_{k,f_{use}} = \left| V_k - \frac{1}{N} \sum_{i=1}^{N} V_i \right| \tag{2.27}$$

The total cost for unbalanced fuel use can be written as a matrix equation in a manner similar to the previously developed relative position cost. If the weighting matrix W_f is

$$W_f = \begin{bmatrix} (1-\frac{1}{N}) & -\frac{1}{N} & \dots & -\frac{1}{N} \\ -\frac{1}{N} & (1-\frac{1}{N}) & -\frac{1}{N} & \vdots \\ \vdots & -\frac{1}{N} & \ddots & -\frac{1}{N} \\ -\frac{1}{N} & \dots & -\frac{1}{N} & (1-\frac{1}{N}) \end{bmatrix} \tag{2.28}$$

then (2.26) and (2.28) lead to the final equal fuel use cost

$$\tilde{C}_{f_{use}} = \left\| W_f \tilde{V} \right\| \tag{2.29}$$

Although (2.29) contains the norm of a function of the norm of $\tilde{U}$, it is still convex and is therefore a valid linear programming constraint. In practice, it is not very difficult to add this constraint to (2.22), as $\tilde{V}$ is already available as part of the minimum fuel cost.

2.2.2 Equal Fuel Remaining

For situations when the spacecraft in the formation all start with the same amount of fuel, it might make sense to consider the approach in the previous section. However, it is more likely that each spacecraft in the formation has a different amount of fuel remaining at the start of the maneuver. In that case, it is preferable for the spacecraft starting with more fuel to use more than those starting with less fuel. This can be handled simply by keeping track of the fuel remaining for each spacecraft. The fuel remaining in the formation prior to planning is written as a vector

$$\tilde{F}^- = \begin{bmatrix} f_1^- & f_2^- & \dots & f_N^- \end{bmatrix}^T \tag{2.30}$$

Here, f_i^- is the fuel remaining for the i^{th} spacecraft before the maneuver, and the units are the same as for $\tilde{U}$. The formation is penalized for having spacecraft with fuel remaining after the maneuver that deviates from the average. The cost for a single spacecraft is then

$$C_{i,f_{rem}} = \left| f_i^- - V_i - \frac{1}{N} \sum_{k=1}^{N} \left(f_k^- - V_k \right) \right| \tag{2.31}$$

Similar to (2.29), the total formation cost can be written in matrix form as

$$\tilde{C}_{f_{rem}} = \left\| W_f (\tilde{F}^- - \tilde{V}) \right\| \tag{2.32}$$

When the linear program is formed, a series of constraints are added to guarantee that the fuel used by each spacecraft does not exceed the fuel available.

$$f_i^- - V_i \geq 0 \qquad \forall i \tag{2.33}$$

The cost in (2.32) is actually a generalization of (2.29), and rewriting it yields

$$\tilde{C}_{f_{rem}} = \left\| W_f \tilde{F}^- - W_f \tilde{V} \right\| \tag{2.34}$$

If the initial fuel of each spacecraft is the same ($f_1^- = f_2^- = ... = f_N^-$), then from the structure of W_f

$$W_f \tilde{F}^- = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T \tag{2.35}$$

and (2.34) reduces to (2.29). A weakness of this formulation is that even a small variation in remaining fuel will be penalized, and over the short term, it is unrealistic to expect the fuel usage of each spacecraft in the formation to be identical. There will be slightly different disturbances acting on each, and the initial state errors will also vary, requiring different levels of control. Balanced fuel use is really better described as a long term objective; some local variation is acceptable, but trends should be avoided. The next approach better captures this behavior.

2.2.3 Fuel Window

The final method of fuel balancing considered here is based on the concept of error boxes from [28]. In the fuel window approach, the fuel level for each spacecraft is allowed to deviate from the average within some acceptable range (window). As long as the fuel remaining for each spacecraft remains within the window, no cost penalty is assessed. If the window is exceeded, the cost increases. Suppose the size of the window is given by ϵ, then the fuel balance cost associated with the i^{th} spacecraft in the formation is

$$C_{i,f_{win}} = \begin{cases} 0, & \text{if } |f_i^+ - \bar{f}^+| \leq \epsilon; \\ Q_{f_{win}} \left(|f_i^+ - \bar{f}^+| - \epsilon \right), & \text{otherwise.} \end{cases} \tag{2.36}$$

In (2.36), the fuel remaining *after* the maneuver for spacecraft i is written as f_i^+, and the average fuel remaining across the fleet $\bar{f}^+$. Choice of the scalar weight $Q_{f_{win}}$ determines how harshly a fuel imbalance is penalized. If it is large enough, then the planner will keep all fuel levels within the window, and fuel will be expended at roughly the same rate throughout the formation.

It is important to recognize that fuel balancing may not be appropriate for all

maneuvers; its usefulness will be highly mission dependent. Weighting fuel balancing too highly may cause some spacecraft in the formation with less distance to move to "fly in circles" to burn fuel at the same rate as the other spacecraft. This behavior is highly undesirable. Careful selection of an appropriate window size ϵ and weighting factor $Q_{f_{win}}$ can help reduce this risk. Note that if $\epsilon = 0$, then the fuel windowing approach is equivalent to the equal fuel remaining approach.

2.3 Implementation Details

Before a formation plan can be generated from the optimization in (2.22), it must first be converted into a linear program. This step is performed by a MATLAB^{TM}code, called the "planner." Once the LP is formulated, it is solved by an optimization code (choices are discussed shortly). The LP is formulated as

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}\mathbf{y} \leq \mathbf{b} \end{aligned} \tag{2.37}$$

The decision variables are $\mathbf{y}$, and the number of constraints is given by the length of the vector $\mathbf{b}$. The number of constraints and decision variables scales linearly with N, so the formation-wide planning problem can grow quickly with the number of spacecraft, particularly given the form of the different costs in (2.22). Each element of the cost function is actually a magnitude. There are several ways to implement this, but the one used in the planner is described in [32]. For the simplified case of a minimum-fuel problem in a single dimension, the optimization is

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} Q_u|u_i| \\ \text{subject to} \quad & \mathbf{A}\mathbf{u} \leq \mathbf{b} \end{aligned} \tag{2.38}$$

The control input at the i^{th} time step is u_i. The matrix $\mathbf{A}$ and constraint vector $\mathbf{b}$ are functions of the spacecraft dynamics, the initial conditions, and the desired

final conditions. The absolute value in the summation in (2.38) is eliminated by first introducing a new decision variable vector $\mathbf{z}$

$$\mathbf{z} = \begin{bmatrix} \mathbf{u}^T & \mathbf{v}^T \end{bmatrix}^T \tag{2.39}$$

and then recasting the optimization as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} Q_u v_i \\ \text{subject to} \quad & \mathbf{A}\mathbf{u} \leq \mathbf{b} \\ & u_i \leq v_i, \qquad i = 1, 2, \ldots, n, \\ & -u_i \leq v_i, \qquad i = 1, 2, \ldots, n \end{aligned} \tag{2.40}$$

The important point is that the optimum values of the new decision variables v_i are chosen by the LP solver to minimize the objective in (2.40). Since $Q_u > 0$, v_i will be as small as possible. The smallest value to satisfy the constraints on v_i is $v_i = |u_i|$. Overall, for every step in the simple, one-dimensional, minimum-fuel problem, there are two decision variables, u_i and v_i, and two additional constraints, $u_i \leq v_i$ and $-u_i \leq v_i$. For a real formation flight problem in three-dimensions, there are six decision variables and six additional constraints per step.

The structure of the problem lends itself to the use of sparse matrix representations for the matrices in (2.22), as well as in the constraint matrix $\mathbf{A}$ that results. For a formation of seven spacecraft, with a planning horizon of one orbit and a discretization of 1000 time steps per orbit, $\mathbf{A}$ is approximately $40,000 \times 40,000$. A full matrix representation would require a prohibitive 1.6 billion entries. If each value is stored as a 64-bit number it requires almost 12 GB of memory. Fortunately, the number of nonzero entries is roughly 600,000. Therefore, by switching to sparse matrices, the required storage space in memory for the constraint matrix alone can be reduced to less than 0.05% of the full matrix equivalent (less than 5 MB). Solving LPs of this size requires a solver that can handle sparse constraint matrices. There are a number of LP solvers available, both free and commercial, that can be used for this purpose.

Most of these solvers either interface with MATLAB^{TM} directly, or can be interfaced using MEX. In most cases, software is freely available for this purpose.

Linprog The MATLAB^{TM} optimization toolkit includes an LP solver called `linprog` that contains both a large-scale interior point solver and a simplex solver. It was used successfully to solve some of the smaller (relatively speaking) problems, but had more difficulty with formations involving a large number of spacecraft. Still, since it is part of MATLAB^{TM}, it has the bonus of being very simple to interface with the rest of the planner.

GNU Linear Programming Kit (GLPK) The GNU solver [64] is freely available and has a primal-dual interior point method and also a revised simplex method. The solver can additionally be used to solve Mixed Integer Linear Programs (MILPs) using a branch and bound algorithm. It is updated fairly regularly and at the time of this writing was on version 4.28. The latest version is available at any GNU mirror, and it can be interfaced to MATLAB^{TM} by using GLPKMEX [65].

COIN-OR LP Solver (CLP) Of the free codes tested, CLP[66] was probably the most successful in solving a wide range of LPs generated by the planner. The COmputational INfrastructure for Operations Research (COIN-OR) project [67] has a variety of free codes available for solving different kinds of optimization problems. Although not as fast as the commercial codes, CLP performed well compared to `linprog` and GLPK for the problems considered here; it has been successfully tested on problems with up to 1.5 million constraints. A free MEX interface for CLP is available called `mexclp` [68].

MOSEK [69] is a commercial software package that can be used to solve a variety of optimization problems, including linear and mixed integer problems. For these problems, the software uses both interior point and simplex methods, as well as a branch and bound and cut algorithm for MIPs. It is usually faster than the freely available codes and also appeared to be more robust. It can use multiple processors

which further speeds up solution time. A full-featured student license is available for the LP solver, which interfaces nicely with MATLAB^{TM}.

ILOG CPLEX [70] is the most well-known optimization software available. It is a commercial package and is very good at solving the problems that result from the methods presented in this thesis. It contains interior-point, dual and primal simplex, barrier algorithms, and others for solving problems. Internally, it uses many heuristics to help speed up the solution. and it is typically the fastest solver for the problems considered here.

2.4 Formation Flying Test Cases

Now that the formation-wide optimization has been developed, this section will demonstrate the flexibility of the planner and examine some of the trade-offs between the different objectives more closely. Examples are included for both a circular orbit as well as an elliptical orbit. The circular orbit is modeled after a geosynchronous orbit. The orbital elements (2.1) are

$$\mathbf{e}_{ref}(t_0) = \begin{bmatrix} 6.6107 & 0.005 & .01 & 0 & 0 & \pi \end{bmatrix}^T \tag{2.41}$$

The first element of e_{ref}, the semimajor axis, is in units of earth radii, the eccentricity is unitless, and the inclination, ascending node, argument of perigee, and mean anomaly are in radians. Likewise, the elliptical orbit is chosen as

$$\mathbf{e}_{ref}(t_0) = \begin{bmatrix} 4.69549 & 0.471 & 1.10497 & 4.24115 & 3.7350 & \pi \end{bmatrix}^T \tag{2.42}$$

2.4.1 Position Cost Comparison

Two types of position constraints were discussed for the formation-wide optimization. The first type constrained each spacecraft in the formation relative to a reference orbit while the second type constrained follower spacecraft to a leader spacecraft. Consider

a formation of spacecraft that is initially separated only in the radial direction. Recalling the definition of the LVLH vector (2.20), the initial states of the spacecraft, in the LVLH frame, are

$$\begin{aligned}
\Delta\mathbf{x}_{d_1} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_2} &= \begin{bmatrix} 250 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_3} &= \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_4} &= \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T
\end{aligned} \tag{2.43}$$

Position are in meters and velocities are in meters per second. Radial separation (the x direction) corresponds to a mismatched semimajor axis, which will introduce a large drift if no corrective action is taken [55]. To eliminate this tendency, the desired configuration after one orbit is an in-track formation with the same separation distances:

$$\begin{aligned}
\Delta\mathbf{x}_{d_1} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_2} &= \begin{bmatrix} 0 & 250 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_3} &= \begin{bmatrix} 0 & 500 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_4} &= \begin{bmatrix} 0 & 1000 & 0 & 0 & 0 & 0 \end{bmatrix}^T
\end{aligned} \tag{2.44}$$

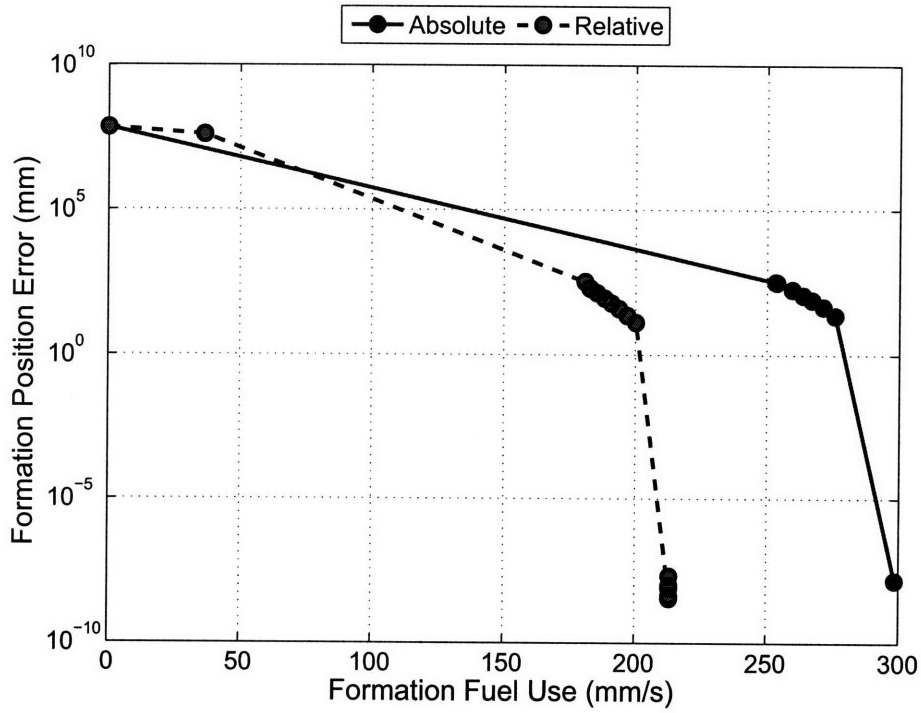
The initial and desired formations are the same for both the elliptical and circular test cases, and they can be enforced with either geometry constraints or relative position constraints. If the geometry constraint is used, then the formation center is simply the reference orbit, and the desired positions for each spacecraft correspond to specific points in space. If instead the relative position constraints are used, then the formation center is allowed to stray some distance from the reference orbit, as long as the follower spacecrafts' positions relative to the leader are maintained. Instead of a single configuration with zero position cost, there are many; the formation is free to move away from the reference orbit if doing so reduces the overall cost. In a sense, the optimization is implicitly computing an optimal virtual center for the formation.

Therefore, it is expected that by using relative position constraints rather than the hard geometry constraint, additional fuel savings might be possible.

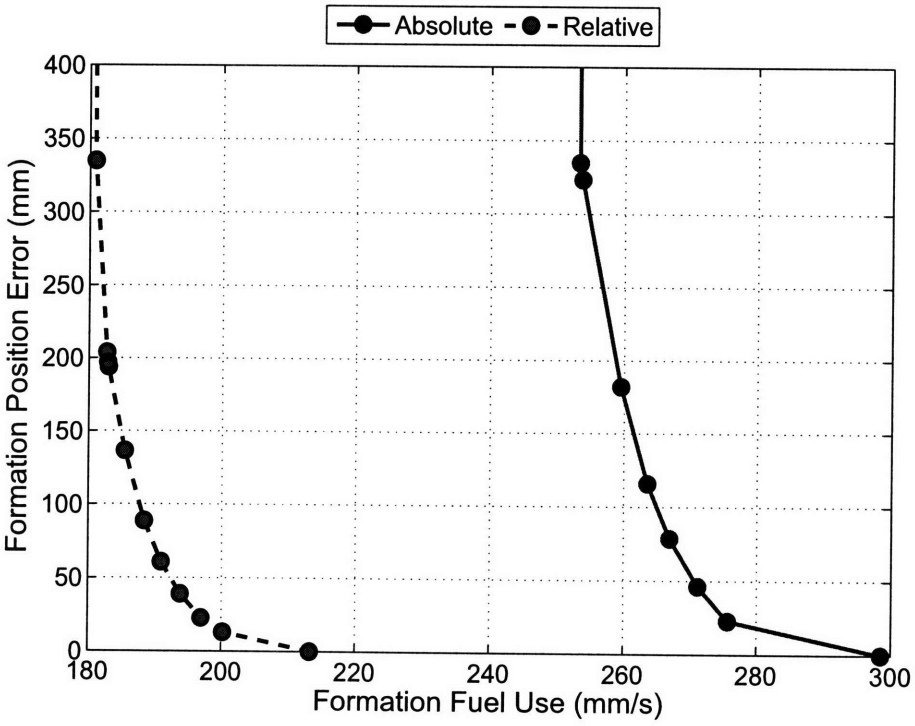
Figures 2-4 and 2-5 show the tradeoff between fuel use and position error when either the geometry constraint or the relative position constraint is used. The plotted fuel usages and position errors are the sum total for the four spacecraft formation for the entire maneuver. As expected, using more fuel improves the position performance. The data points are obtained by varying the scalar weights Q_x, Q_r, and Q_u in the cost function. When the geometry constraint is used, Q_r is set to zero. Likewise, when the relative position constraint is used, Q_x is set to zero. This prevents both types of position constraints from being active simultaneously. For these tests, Q_u was held fixed at 1000, while the position constraint weighting was varied from 100 to 10^{10}.

In both the circular and elliptical cases, applying no control results in very large errors, about 68 km and 17.5 km respectively. These are the sums of the errors in the x, y and z directions for the follower spacecraft. The cause of the excessive drift is the mismatched semimajor axes of the spacecraft in the formation. As more fuel is used ($\frac{Q_x}{Q_u}$ or $\frac{Q_r}{Q_u}$ increases), the performance improves until the desired formation is precisely achieved. The best solutions obtained yielded final position errors of less than 10^{-6} mm, which is on the order of the tolerance of the LP solver (and, of course, far better than will be achieved in practice). To achieve this level of performance for the geometry constraint cases required fuel expenditures of 298.5 mm/s for the circular orbit, and 303.9 mm/s for the elliptical orbit. When the relative constraint was used instead, the fuel expenditures were 213.2 mm/s and 217.1 mm/s. By specifying the formation as a set of relative constraints, rather than absolute geometry constraints, fuel savings of 28.6% are achieved.

Figure 2-6 shows the planned trajectories for the spacecraft for the circular orbit with the geometry constraint. The coordinate frame is absolute, with the reference orbit held fixed at the origin. A ♦ marks the starting position of a spacecraft, and a ■ marks the final position. It is clear that the planned trajectories guide each spacecraft to its desired position. Note that on-orbit, the actual trajectories will

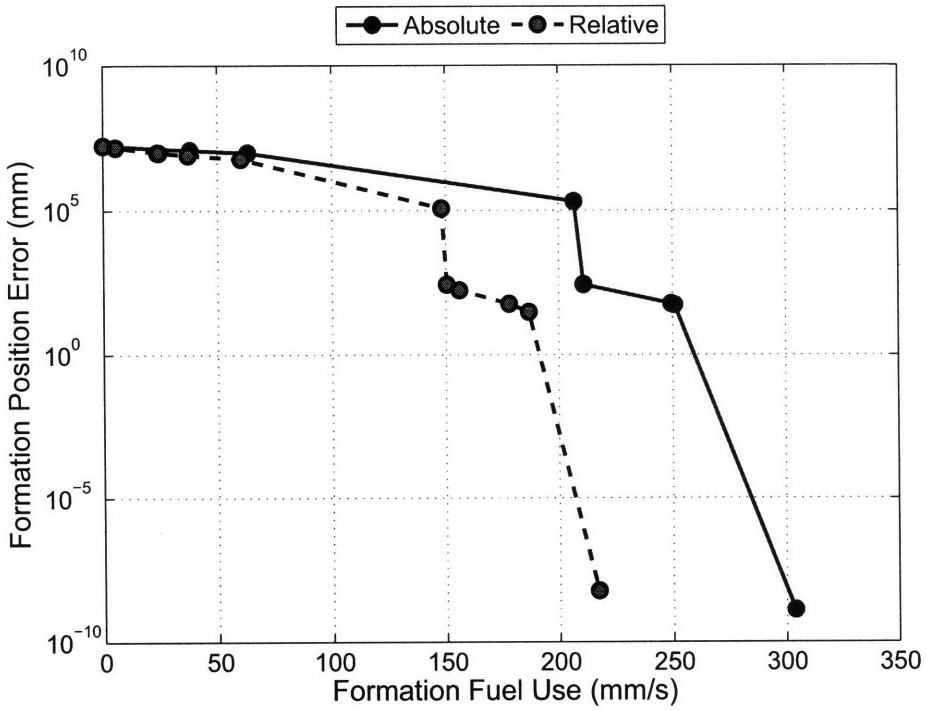


(a) Circular Orbit

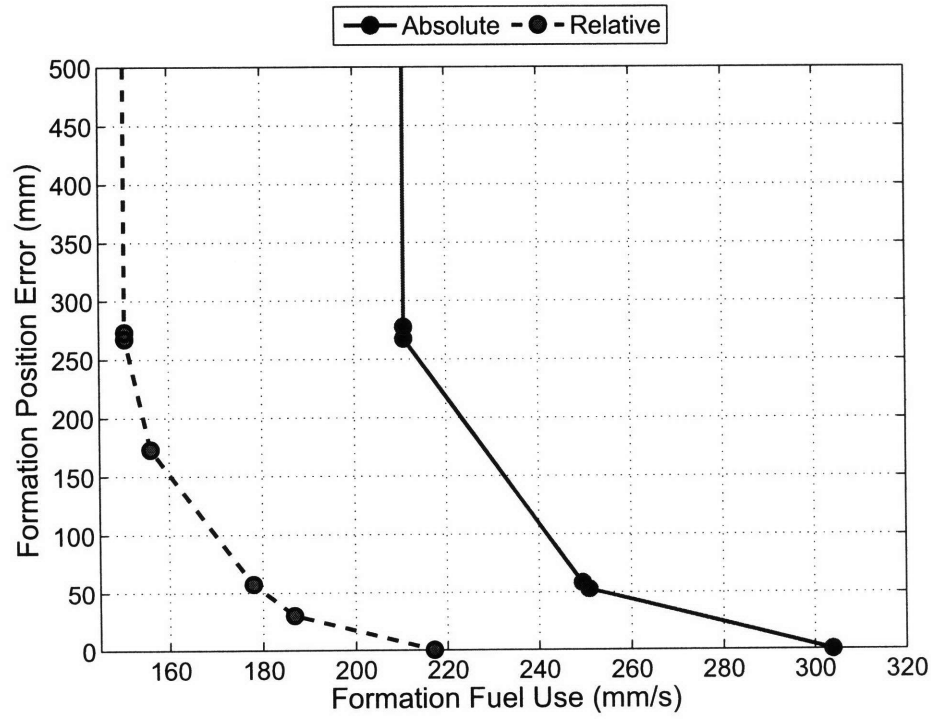


(b) Detail of Circular Orbit

Figure 2-4: Effect of fuel use on position cost for a circular orbit.

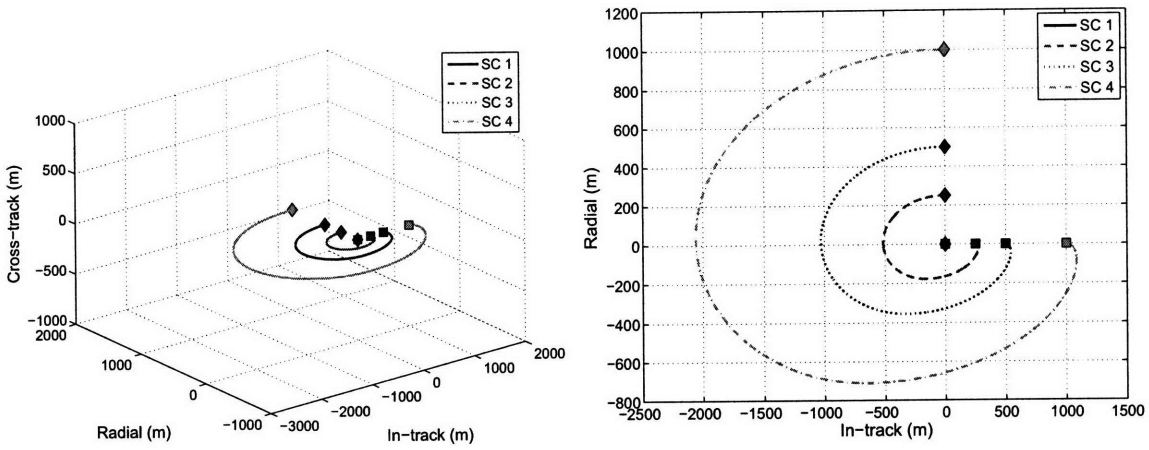


(a) Elliptical



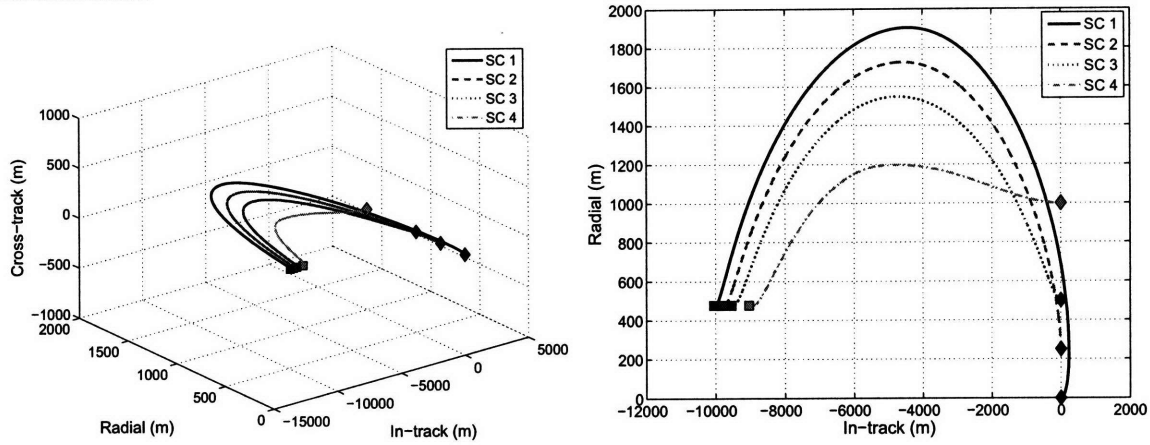
(b) Detail of Elliptical Orbit

Figure 2-5: Effect of fuel use on position cost for an elliptical orbit.

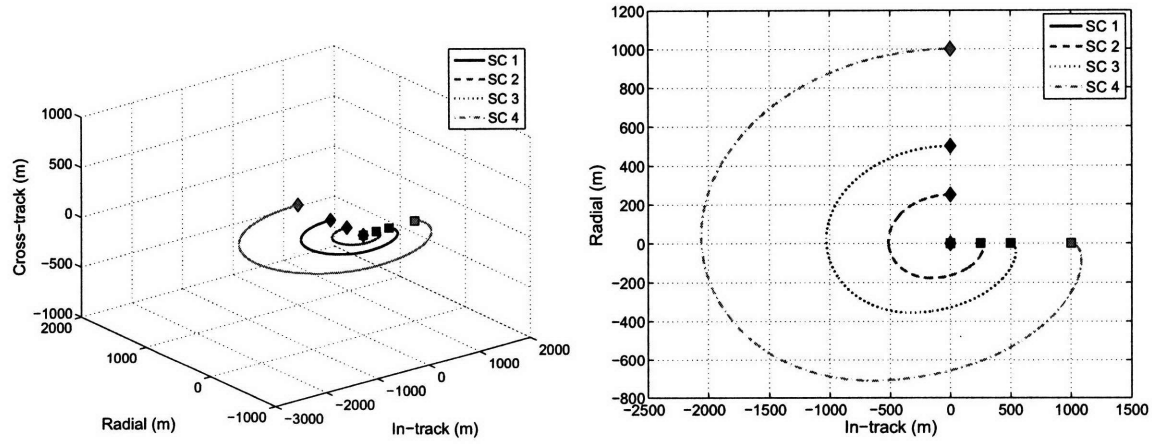


(a) Absolute frame (reference orbit fixed at origin)

Fig. 2-6: Reconfiguration maneuver for a circular orbit using absolute geometry constraints.

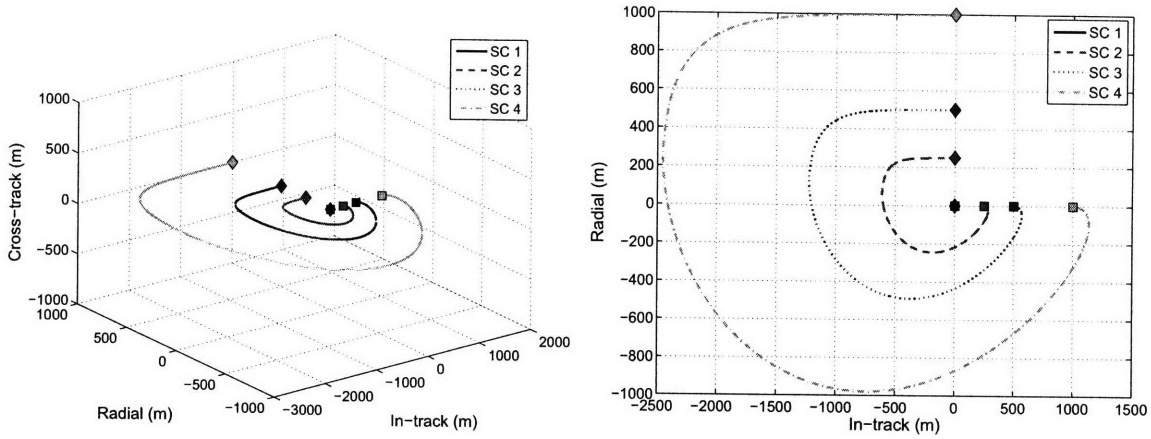


(a) Absolute frame (reference orbit fixed at origin)



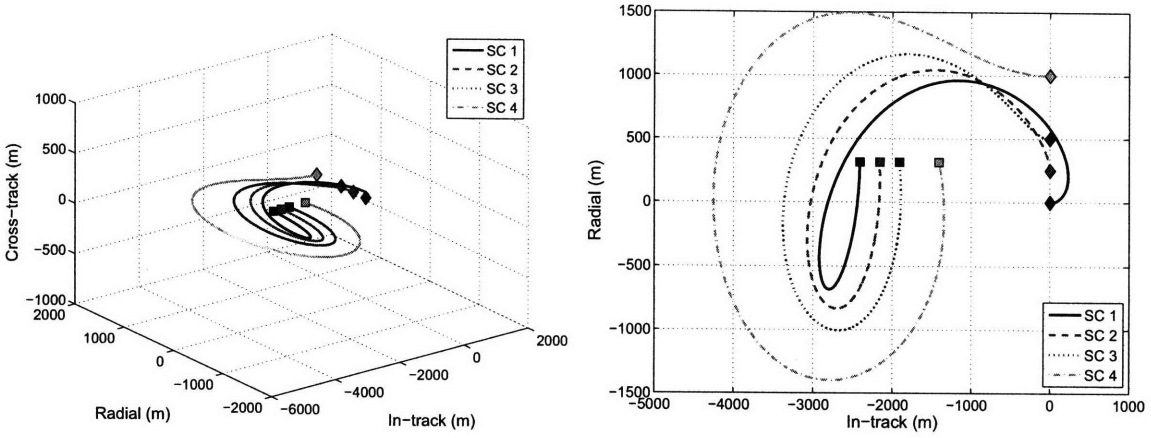
(b) Relative frame (leader spacecraft fixed at origin)

Fig. 2-7: Reconfiguration maneuver for a circular orbit using relative position constraints.

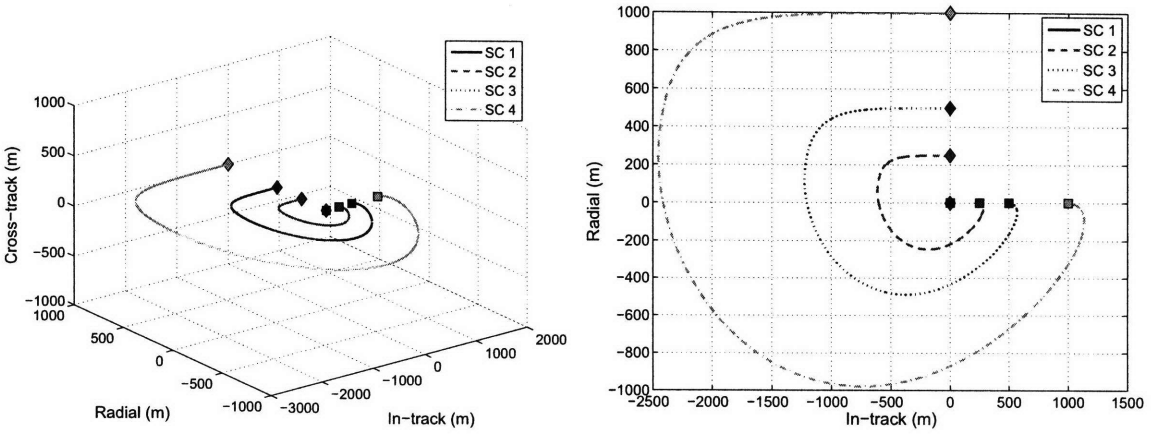


(a) Absolute frame (reference orbit fixed at origin)

Fig. 2-8: Reconfiguration maneuver for an elliptical orbit using absolute geometry constraints.



(a) Absolute frame (reference orbit fixed at origin)



(b) Relative frame (leader spacecraft fixed at origin)

Fig. 2-9: Reconfiguration maneuver for an elliptical orbit using relative position constraints.

differ slightly from the planned trajectories because of process error, linearization error in the dynamics, and disturbances not modeled in the planner. Chapter 5 will validate the planner via realistic simulations where these effects are included. Figure 2-7 illustrates what happens when the geometry constraints are exchanged for relative position constraints. For this case, the trajectories are plotted in two different coordinate frames. The trajectory is plotted in the absolute frame (which is the same as previously described) in Figure 2-7(a). The same trajectory is plotted in the relative frame in Figure 2-7(b). In the relative frame, the origin is fixed to the leader spacecraft. The added flexibility provided by using the relative constraint is clear in Figure 2-7(a). The formation is not strictly bound to the reference orbit, and the 28.6% reduction in fuel use is obtained by shifting its center by 10 km. However, the relative separations between the leader spacecraft and its followers are preserved. For most formation flying missions, it is the relative formation geometry, not the absolute geometry, that is critical. For this reason, the relative position constraints are very useful for reducing fuel usage. The trajectories for the elliptical test case are shown in Figure 2-8 and Figure 2-9. Although the exact trajectories expectedly differ from those for the circular orbit, the general characteristics are the same. In this case, however, the formation center only shifts by 2.43 km.

The dynamics used in the optimization are linearized about the reference orbit. When discarding the absolute geometry constraints in favor of the relative position constraints, care must be taken to ensure that these linear dynamics remain valid. By removing all of the absolute geometry constraints, the formation is free to move away from the reference orbit if doing so saves fuel. Reference [29] discusses the accuracy of the linear approximations and the ranges over which they are valid. There, two test cases are presented, a circular orbit and a highly elliptical orbit. For the circular orbit, the acceptable separation was around 25 km, and for the elliptical orbit, 50 km. Acceptable velocity ranges were computed as 40 m/s and 2 m/s, respectively. Staying within these boundaries keeps the linearization accurate to within 1%. Given any arbitrary reference orbit $\mathbf{e}_{ref}$, computation of the acceptable separations is fairly straightforward. The exact amount of acceptable linearization error will depend on

mission requirements.

Error box constraints like those discussed in [28, 57] can be used to guarantee that the spacecraft remain within a specified distance of the reference orbit and ensure that the dynamics remain valid throughout the maneuver. For example, the leader spacecraft can be "anchored" to the reference orbit by only permitting it to move a certain distance away. The desired positions of the follower spacecraft can then be specified relative to the leader, without danger of the entire formation moving too far away from the reference orbit and encountering unacceptable amounts of linearization error. The added degree of freedom available to the leader (compared with the absolute geometry case) can still result in fuel savings. For both examples presented here, the formation remained well within the recommended distance of the reference orbit.

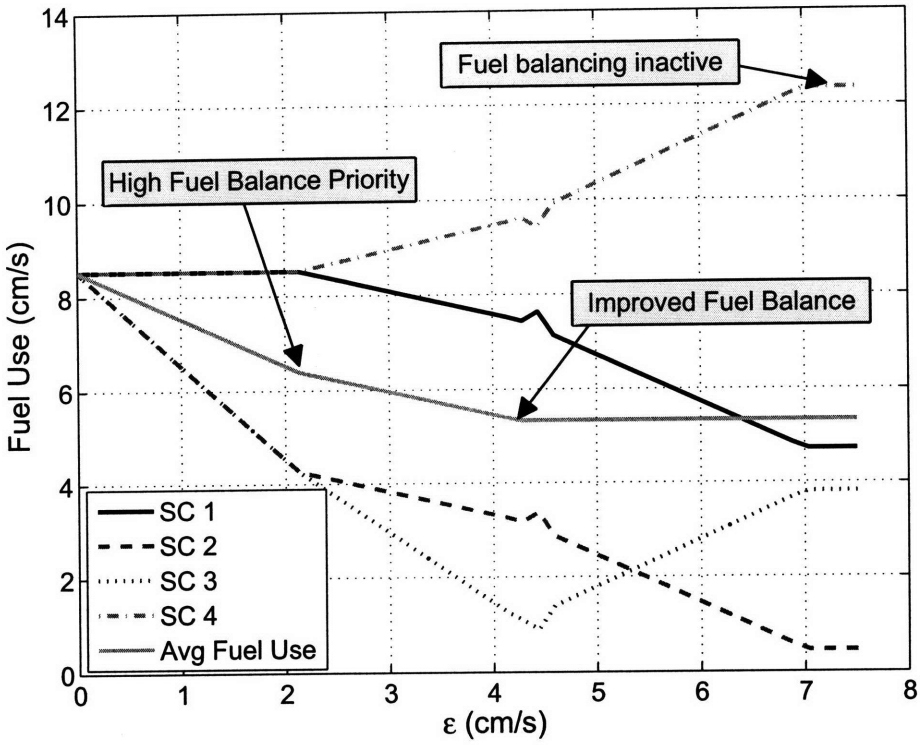
2.4.2 Fuel Balancing Example

If fuel balancing is added to the scenario discussed in the previous section, several new planning possibilities emerge. For this analysis, the most general form of the fuel balancing constraint is used: the fuel window. The circular orbit given by (2.41) is used, and the initial and desired positions are the same as in (2.43) and (2.44). For maximum flexibility, the relative position constraints are used to specify the formation geometry. In addition to the minimum fuel and relative position costs, the fuel window cost (2.36) is now included. The scalar weights were set as follows:

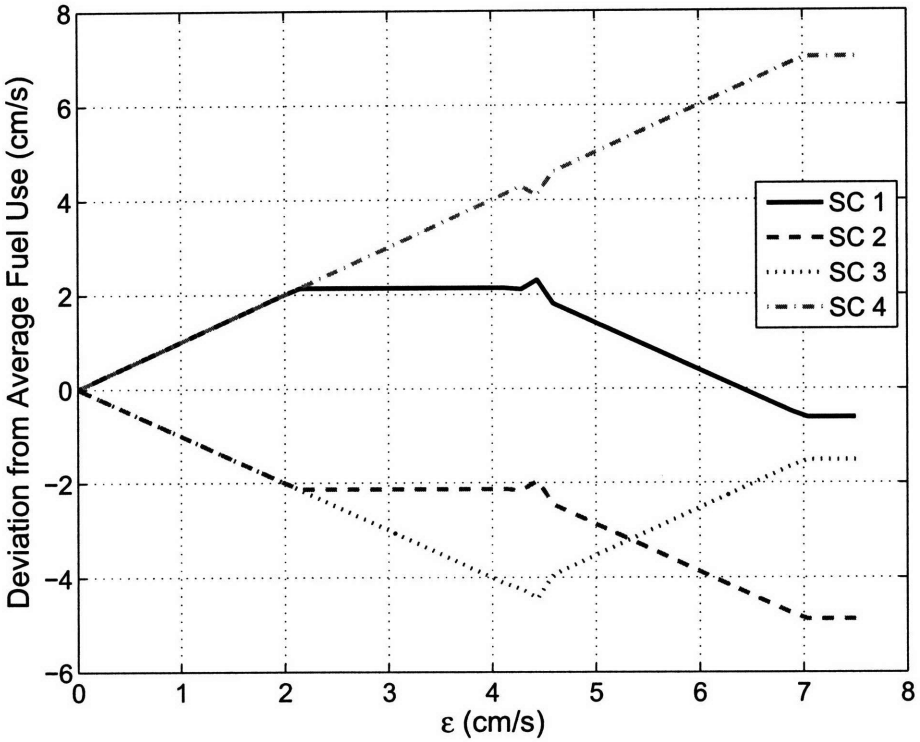
$$Q_u = 10, \quad Q_r = 100, \quad Q_{f_{win}} = 1000 \tag{2.45}$$

These choices emphasize that the formation geometry is maintained and that the fuel window is strictly enforced. The effect of the fuel window on the plan will depend on the selection of ϵ. A large value will not cause any change in the plan from the previous solution because the constraint will not be active. As ϵ is reduced, the plan will be adjusted to keep fuel expenditure within the window.

Figure 2-10 shows the evolution of fuel usage for the four spacecraft as ϵ is adjusted.



(a) Individual and average fuel use



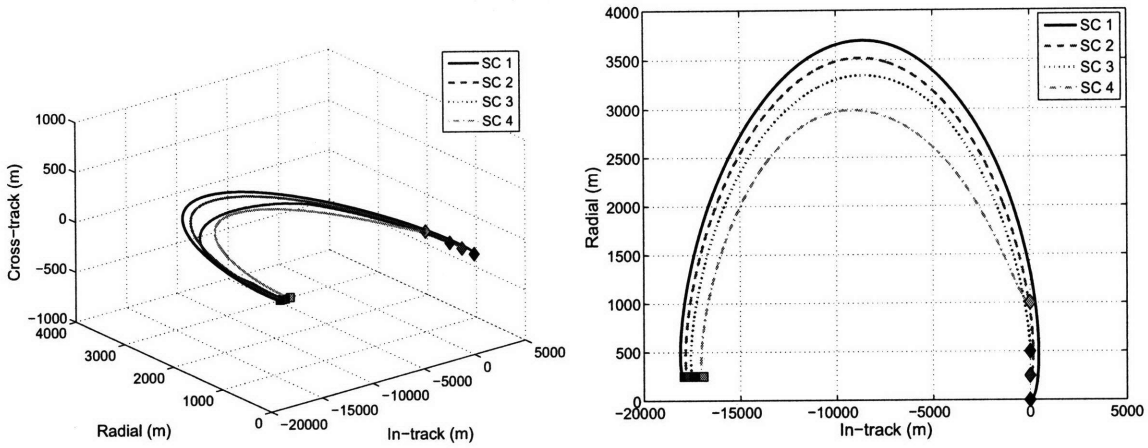
(b) Deviations from average fuel use

Fig. 2-10: Fuel management as fuel balancing window size ϵ is adjusted.

Figure 2-10(a) plots the fuel use for each spacecraft along with the average fuel use. For $\epsilon > 7$ cm/s, the fuel balancing constraint is not active, as the optimal plan from the previous section satisfies this window. Examining the plans, spacecraft 2 uses only 0.45 cm/s of fuel, while spacecraft 4 uses 12.3 cm/s of fuel. Given this wide range, it would be beneficial to obtain a better balance across the fleet. In fact, the range can be reduced somewhat without using any more fuel. The average fuel use is constant down to $\epsilon = 4.25$ cm/s. Although spacecraft 4 still uses the most fuel, it has been reduced to 9.6 cm/s. Spacecraft 3 now uses the least fuel, but this lower bound has been increased to 1.1 cm/s. It is important to note that the overall fuel use of the formation has not increased; it has just been more evenly distributed.

There is still a fairly large discrepancy, and if fuel balancing is a top priority, the window can be shrunk even more, but now at the cost of increased overall fuel use. The smallest window of practical value that can be achieved for this example is for $\epsilon = 2.2$ cm/s. For this case, spacecraft 1 and spacecraft 4 each use 8.5 cm/s of fuel, and spacecraft 2 and spacecraft 3 each use 4.25 cm/s. The variability in fuel use has been significantly reduced, although the average fuel use has increased from 5.33 cm/s to 6.31 cm/s. This corresponds to an 18% increase in overall fuel use. Additional fuel balancing has been gained, though: with no fuel balancing, fuel use between spacecraft varied by nearly 12 cm/s. With fuel balancing, it has been reduced to 4 cm/s, a notable improvement. Further reductions to ϵ drive the fuel use of spacecraft 2 and spacecraft 3 to that of the other two spacecraft, but nothing practical is gained from this; they simply burn more fuel to satisfy the fuel window constraint. Figure 2-10(b) shows the deviation from the average fuel use for each spacecraft as ϵ changes, and the effect of the contracting window is evident.

Figure 2-11 depicts the trajectories followed by the formation. Overall, they are quite similar to the trajectories with fuel balancing absent, but there are two important differences. The first is that the absolute destination has been adjusted to about 18 km from the reference orbit (still within the valid linearization range). The second is that spacecraft 3 has some cross-track motion, visible in the three-dimensional trajectory graph.



(a) Absolute frame (reference orbit fixed at origin)

Fig. 2-11: Reconfiguration maneuver for a circular orbit with fuel balancing (ϵ = 2.15 cm/s).

2.5 Chapter Summary

This chapter extended a planning architecture, previously used for a single spacecraft, for use with multi-spacecraft formations. The separation of the initial state from the desired state allows the framework to be used for a broad spectrum of formation initialization problems, including both formation-keeping and formation reconfiguration. Previously, formation flying with partial J_2-invariance had been achieved by combining several independent optimizations, and although this capability still exists, the expansion of the formulation to a fleet-wide optimization introduces a range of new possibilities. The concept of a leader and followers was applied to add relative position constraints that can reduce overall fuel usage for reconfiguration maneuvers. Fuel balancing was also addressed, and the implementation of a fuel window technique improved the performance of the planner in this area. Fleet-wide planning capabilities have been improved considerably through the enhancements discussed in this chapter. Use of the planner is no longer limited simply to formation-keeping; it can now be used for complex maneuvers and reconfigurations.

Chapter 3

Implementation of ΔV Commands

Historically, sensor noise has been a key factor in the analysis of system performance for formation flying spacecraft [37, 45]. Since formation flying missions require coordination between multiple spacecraft, knowledge of the relative states must be as accurate as possible. This knowledge is typically used when planning trajectories to meet specific criteria, such as minimizing fuel use or maintaining a desired formation geometry (as in Chapter 2). The planning process depends on knowledge of the initial conditions of the spacecraft [39], and degrades with increasing error. Specifically, Tillerson showed that velocity estimation error was the primary cause of poor performance, and that an error of just 2 mm/s can result in errors on the order of 30 m after just one orbit. At that time, the best navigation filters, using Carrier-Phase Differential GPS (CDGPS) signals, achieved velocity accuracy on the order of 0.5 mm/s [46]. Since then, the state-of-the-art has improved significantly, with Leung and Montenbruck [47] demonstrating a filter that can estimate relative position and velocity to within 1.5 mm and 5 μm/s, respectively. While those numbers represent a best-case performance for real world operation, with these improvements, it is nonetheless important to understand how other sources of noise can affect formation flight. The PRISMA formation flying demonstration mission aims to validate sensor and actuator technologies for formation flight and rendezvous and docking. Consisting of a two-spacecraft formation deployed in a sun-synchronous 700 km orbit, detailed simulations of the estimation and control processes have indicated that a

driver of system performance is thruster performance [36]. Specifically, the minimum impulse bit of 0.7 mm/s is only adequate for controlling the relative mean in-track formation to within 60 m when using the impulsive control scheme presented by [71], indicating that incorrect implementation of a thruster burn is a significant source of error.

Thruster burns are commonly specified by the spacecraft's desired change in velocity (ΔV), and the propulsion subsystem is then responsible for applying this ΔV to the spacecraft. Mission critical maneuvers require precise implementation of thruster burns. For example, for the Cassini Saturn Orbit Insertion ($\Delta V \approx 625$ m/s), an algorithm that measured the energy change of the spacecraft was used. This energy change was monitored autonomously by Cassini during the burn, using real-time measurements from onboard accelerometers [50]. The insertion maneuver was successfully terminated when the desired energy change was reached, and Cassini became the first spacecraft to orbit Saturn. Similarly, the smart impactor of the Deep Impact mission successfully targeted the comet Tempel 1 on July 4, 2005. Precision implementation of trajectory correction maneuvers (TCMs), made possible by the ADCS software and accelerometers, is credited as a primary reason for mission success [51]. The TCM-1 maneuver had a command ΔV of 28.568 m/s, and the control system delivered 28.561 m/s — a performance within 0.03% accuracy.

This chapter discusses the impact of ΔV implementation error on the performance of spacecraft formations and rendezvous and docking scenarios. It also explores how accelerometers can be used to improve performance by providing accurate measurements of the applied ΔV.

3.1 Plan Implementation

If knowledge of the initial state of the spacecraft is sufficiently accurate to determine a good plan, the next key step is to accurately implement the plan. A good plan is one that meets performance objectives (formation geometry, fuel management, drift-free, etc). If such a plan is improperly implemented, the performance objectives might

not be met. At best, this probably means replanning and trying again, resulting in a loss of time and fuel. At worst, it could lead to the loss of a spacecraft or an entire formation. A spacecraft executes a ΔV command as a sequence of one or more thruster firings. The on and off times of the thrusters can be pre-computed beforehand or calculated on-the-fly. Considering the simple case of a single thruster pointing in the direction of motion, one option is to calculate the burn duration based on an idealized thruster model, and then execute the burn. This open-loop strategy is

$$t_b = \frac{m\Delta V_d}{T_d} \tag{3.1}$$

where t_b is the burn duration, m is the mass of the spacecraft (assumed constant throughout the burn), ΔV_d is the desired velocity change, and T_d is the expected force that is provided by the thruster. This approach is easy to implement, but it is usually a poor idea. Although data on the expected thrust is typically be available, this data is likely based on laboratory testing under specific or idealized conditions. As the operating conditions of the thruster change, so too will its performance. This makes it difficult to predict the performance of the thruster or the open-loop system. The actual delivered thrust can instead be modeled as

$$T_a = T_d(1 + \delta) + w \tag{3.2}$$

where T_a is the actual thrust delivered, δ is a small number representing a bias on the expected thrust, and w is a small random variable that accounts for any additional variations in the thrust. These additional variations are not limited to just unpredictable fluctuations in the engine thrust, but could also account for a spacecraft that is spinning slowly while thrusting (provided the time scale of the spin is small compared to the burn duration). Using the strategy in (3.1), and imposing the actuator model in (3.2), the actual ΔV implemented using the open-loop strategy can be written as

$$\Delta V = \left(\frac{T_d}{m}\right) t_b + \left(\frac{T_d\delta + w}{m}\right) t_b \tag{3.3}$$

which is just ΔV_d plus an error term. It is clear that the resulting error from this implementation will be proportional to δ as well as the length of the burn. For high ΔV maneuvers, this error could be quite large, and is therefore unacceptably risky.

3.1.1 Using Accelerometers to Improve ΔV Implementation

A straightforward way to improve the performance of the actuator is to design a feedback control loop that uses sensors to measure thruster performance. Rather than relying on GPS or CDGPS to measure the ΔV changes *after* the burn, this chapter explores using other sensors as a means to measure and track the ΔV *as it is applied.* This approach uses direct feedback and the addition of an axial accelerometer along the thrust direction to achieve this objective. To simplify the following analysis, it is assumed that the thruster can deliver a continuous range of thrusts. Discrete thrusts are discussed later; specifically, the control system that is developed for SPHERES in Chapter 4 uses discrete on-off thrusters. Figure 3-1 depicts a block diagram for the closed-loop thruster control system. For clarity, the state is chosen as

$$\mathbf{x} = \begin{bmatrix} V^* \\ V_a \\ \hat{V} \\ e \end{bmatrix} \tag{3.4}$$

The commanded velocity is V^*, the actual velocity of the spacecraft is V_a, and the estimated velocity of the spacecraft is $\hat{V}$. The final element of the state vector, e, is the integral of the estimated velocity error. It will be used later by the controller. Although each of these velocities is actually a velocity *change*, the Δ is omitted for notational simplicity. The velocity command enters the system through the control input a^*, such that

$$\dot{V}^* = a^* \tag{3.5}$$

The actual acceleration experienced by the spacecraft is influenced by a number of factors. First, the acceleration command a^* is sent directly to the thruster, but

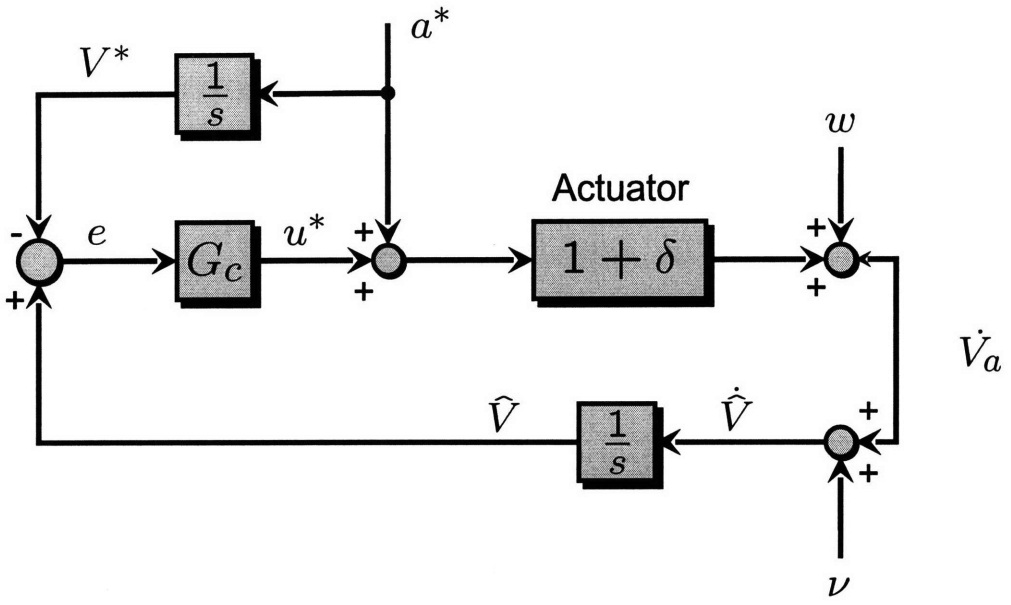


Figure 3-1: Closed-loop control system for single thruster.

is distorted by the actuator ("actuator" and "thruster" are used interchangeably) dynamics. Additionally, a corrective acceleration u^* is applied by the controller, but once more, the actuator dynamics distort u^* so that $u = (1+\delta)u^*$. This corrective acceleration should be thought of as a throttling of the thruster. For example, if $\delta > 0$, then u^* will be negative, and the thrust command will be reduced to compensate for the unexpected amplification of the signal that happens in the actuator. Finally, random thruster process noise w is added.

$$\dot{V}_a = (a^* + u^*)(1+\delta) + w \tag{3.6}$$

The accelerometer is mounted on the spacecraft and measures the actual acceleration directly with noise ν

$$\dot{\hat{V}} = \dot{V}_a + \nu \tag{3.7}$$

It is assumed that both w and ν are uncorrelated Gaussian, white-noise processes

$$w \sim \mathcal{N}(0, \sigma_w) \text{ and } \nu \sim \mathcal{N}(0, \sigma_\nu) \tag{3.8}$$

As noted above, e is the integral of the estimated velocity error.

$$e = \int_0^t (\hat{V} - V^*)dt \tag{3.9}$$

A PI controller $G_c(s)$ with gains k_p and k_i can be used to drive the estimated velocity error to zero, and thus is a reasonable choice. The state of the integrator is e, as shown in (3.9). The control law for the PI controller may be written as

$$u^* = -k_p(\hat{V} - V^*) - k_i e \tag{3.10}$$

Substituting (3.10) into (3.6) and (3.7) yields the following state relationships

$$\dot{V}^* = a^* \tag{3.11a}$$

$$\dot{V}_a = [-k_p(\hat{V} - V^*) - k_i e)](1+\delta) + a^*(1+\delta) + w \tag{3.11b}$$

$$\dot{\hat{V}} = [-k_p(\hat{V} - V^*) - k_i e)](1+\delta) + a^*(1+\delta) + w + \nu \tag{3.11c}$$

$$\dot{e} = \hat{V} - V^* \tag{3.11d}$$

Or, in matrix form

$$\begin{bmatrix} \dot{V}^* \\ \dot{V}_a \\ \dot{\hat{V}} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ (1+\delta)k_p & 0 & -(1+\delta)k_p & -(1+\delta)k_i \\ (1+\delta)k_p & 0 & -(1+\delta)k_p & -(1+\delta)k_i \\ -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V^* \\ V_a \\ \hat{V} \\ e \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ (1+\delta) & 1 & 0 \\ (1+\delta) & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a^* \\ w \\ v \end{bmatrix} \tag{3.12}$$

Equations (3.11) and (3.12) provide a convenient form for simulating the system response to a range of acceleration commands. To gain more insight into the steady

state behavior of this controller, begin with the generic system

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B_u\mathbf{u} + B_w\mathbf{w} \\ \mathbf{y} &= C\mathbf{x} + \nu\end{aligned} \tag{3.13}$$

where $\mathbf{x}$ is as in (3.4). In this formulation, the control inputs $\mathbf{u}$ and the random perturbations $\mathbf{w}$ are separated. If the control law is chosen as linear state feedback, such that

$$\mathbf{u} = -K\mathbf{x} \tag{3.14}$$

then (3.13) may be written as

$$\begin{aligned}\dot{\mathbf{x}} &= (A - B_u K)\mathbf{x} + B_w\mathbf{w} \Rightarrow \\ \dot{\mathbf{x}} &= A_{cl}\mathbf{x} + B_w\mathbf{w}\end{aligned} \tag{3.15}$$

The dynamics in (3.15) are that of a system driven by random process noise. For a linear, time-invariant (LTI) system, if the process noise $\mathbf{w}$ is stationary, then the mean square value of the state, as $t \to \infty$, satisfies the Lyapunov equation

$$0 = A_{cl}X_{ss} + X_{ss}A_{cl}^T + B_w R_{ww} B_w^T \tag{3.16}$$

Given positive definite R_{ww}, (3.16) has a positive definite solution for the state covariance matrix X_{ss} if A_{cl} is stable [72]. Although (3.12) is convenient for simulating all of the parameters of interest, the A_{cl} matrix has two eigenvalues of 0, and therefore solution of (3.16) is not possible. By introducing $\epsilon = \hat{V} - V^*$ and $n = (w + \nu)$, a lower order system is obtained:

$$\begin{bmatrix} \dot{\epsilon} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -(1+\delta)k_p & -(1+\delta)k_i \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \epsilon \\ e \end{bmatrix} + \begin{bmatrix} \delta & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a^* \\ n \end{bmatrix} \tag{3.17}$$

Although a^* is not actually a random process noise, this formulation is still useful; provided $k_p > 0$ and $k_i > 0$, then A_{cl} for this system will be stable and there is a

unique solution to the Lyapunov equation. While there are numerical algorithms for solving the equation, explicitly solving for the 2×2 case is feasible. The spectral intensity matrix R_{ww} for this problem is

$$R_{ww} = \begin{bmatrix} \sigma_{a^*}^2 & 0 \\ 0 & \sigma_n^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_n^2 \end{bmatrix} \tag{3.18}$$

Since $\sigma_{a^*}^2$ is the reference input, it is known exactly and therefore the upper left entry of R_{ww} is 0 in (3.18). The steady state covariance matrix for the state is

$$X_{ss} = \begin{bmatrix} \sigma_\epsilon^2 & \rho_{\epsilon e}\sigma_\epsilon\sigma_e \\ \rho_{e\epsilon}\sigma_e\sigma\epsilon & \sigma_e^2 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \tag{3.19}$$

As usual, σ and ρ represent standard deviations and correlation coefficients, respectively. If the entries of A_{cl} are numbered in the same manner as (3.19), then the solution of the Lyapunov equation is straightforward. Performing the matrix multiplications in (3.16) yields a system of four equations for the four unknowns in X_{ss}.

$$2A_{11}x_{11} + A_{12}x_{12} + x_{21} = \sigma_n^2 \tag{3.20}$$

$$A_{21}x_{11} + (A_{11} + A_{22})x_{12} + A_{12}x_{22} = 0 \tag{3.21}$$

$$A_{21}x_{11} + (A_{11} + A_{22})x_{21} + A_{12}x_{22} = 0 \tag{3.22}$$

$$A_{21}x_{12} + A_{21}x_{21} + 2A_{22}x_{22} = 0 \tag{3.23}$$

Simultaneous solution of these equations gives the steady state performance of the closed-loop control system driven by the random input **w**

$$X_{ss} = \begin{bmatrix} \dfrac{(\sigma_w + \sigma_\nu)^2}{2(1+\delta)k_p} & 0 \\ 0 & \dfrac{(\sigma_w + \sigma_\nu)^2}{2(1+\delta)k_p k_i} \end{bmatrix} \tag{3.24}$$

The diagonal elements x_{11} and x_{22} in (3.24) are the expected variances in the state variables ϵ and e, respectively. As might be expected, if either the process noise or

sensor noise increases, the steady state behavior of the estimate degrades.

However, an important distinction is that (3.24) describes how well the *estimated velocity* will converge on the command velocity. Of perhaps more importance to overall performance is how well this estimated velocity tracks the actual imparted velocity, V_a. From Figure 3-1

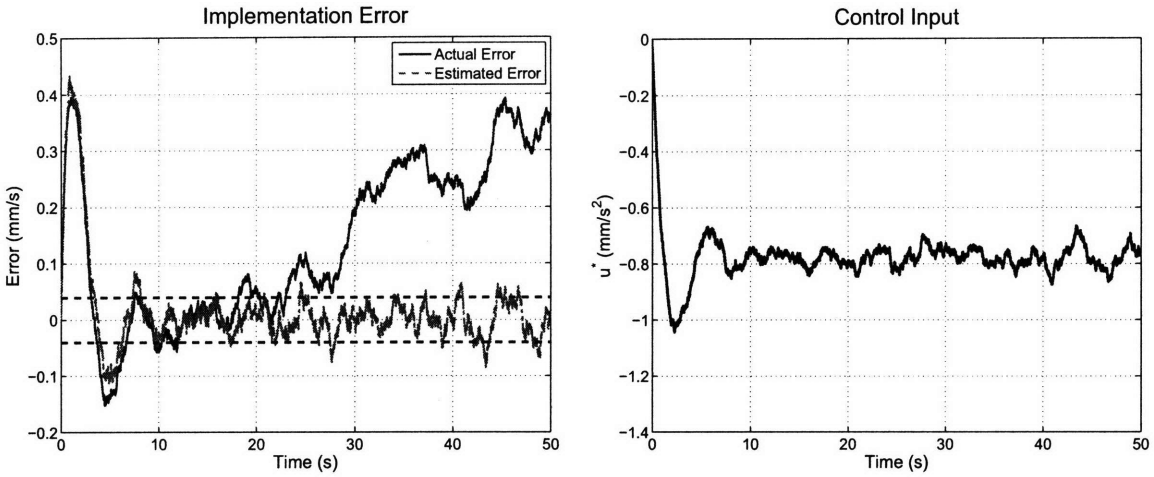
$$\dot{\hat{V}} - \dot{V}_a = \nu \;\Rightarrow\; \hat{V} - V_a = \int_0^t \nu dt \tag{3.25}$$

As seen in (3.25), the estimation error during the burn grows as the integral of the random sensor noise ν. While the expected value of the error is zero for any burn duration, the variance of the estimate will increase for longer burns. This means that for long burns, although the expected implementation error remains zero, the uncertainty grows. Fortunately, the sensor noise for a good accelerometer will be quite small, and the benefit of the added information far outweighs any potential pitfalls. For example, the Mars Reconnaissance Orbiter (MRO) accelerometers had measurement noise level of just 0.005 mm/s^2 [73]. Furthermore, for long burns, it might be possible to periodically obtain velocity measurements from other sensors to reduce the estimate error.

Increasing the gains k_p and k_i in the controller will cause the estimated velocity to converge faster to the command velocity, but it does not effect the long-term tracking of the actual velocity; the ΔV accuracy is purely limited by the sensor noise. As mentioned previously, provided the accelerometer is reasonably good quality, then $\nu \ll \left(\frac{T_d}{m}\right)\delta$ and the error for the closed-loop system will grow much more slowly than for the open-loop system.

3.1.2 Discrete Example

The analysis in Section 3.1.1 is for a continuous system, but in reality, a control system would be implemented digitally. This section discusses the extension of the previous results to the digital domain. The steady state performance of the system can be directly obtained from (3.24) by scaling the process and sensor noise. The



(a) (b)

Figure 3-2: Results of a discrete simulation of the closed-loop algorithm.

relationship between continuous and discrete noise models is discussed in [74]. For this case, they are

$$\sigma_w = \sigma_{Dw}\sqrt{T} \tag{3.26a}$$

$$\sigma_\nu = \sigma_{D\nu}\sqrt{T} \tag{3.26b}$$

$$\sigma_n = \sigma_{D\nu}\sqrt{T} \tag{3.26c}$$

where $\sigma_{D(*)}$ represents the standard deviation of a digital measurement or process, and T is the sample period. Substituting (3.26) into (3.24) gives the mean square performance of the digital estimate.

$$\sigma_\epsilon{}^2 = \frac{(\sigma_{Dw} + \sigma_{D\nu})^2 T}{2(1+\delta)k_p} \tag{3.27}$$

Now, the steady state convergence can be improved by increasing the sampling frequency (and reducing T). Figure 3-2 shows the results of a simulation of the closed-loop control system. The parameters for the simulation were set as follows: $a^* = 4$ cm/s^2, $V^* = 200$ cm/s, $\delta = 0.02$, $\sigma_{Dw} = 0.08$ cm/s, $\sigma_{D\nu} = 0.1$ cm/s^2, $T = 0.001$ s, $k_d = 1$, and $k_i = 1$.

For a shorter burn, the gains could be set higher to obtain a faster convergence of the estimate, but they were left low in this case so that the corrective action of the controller is clearly visible. Since $\delta > 0$, the actual thrust is initially higher than the commanded thrust. Therefore, near the beginning of the burn, the accelerometers detect the variation and the estimated error starts to increase. The controller then applies a differential thrust in the negative direction to counteract δ. Because a^* is 4 cm/s^2 and $\delta = 0.02$, this differential thrust should be about -0.8 mm/s^2. The control history shown in Figure 3-2(b) confirms this prediction.

For the values used in this simulation, (3.27) predicts that $\sigma_\epsilon \approx 0.04$ mm/s. This region is marked by the dashed line in Figure 3-2(a) and closely matches the actual behavior. The data also shows how the actual error tends to drift from the estimated error as a consequence of (3.25). The overall performance of the closed-loop system results in an error of 0.36 mm/s, for a burn of 2000 mm/s, or less than 0.02% error. For the open-loop controller, the error would have been δ, or 2%. The closed-loop system delivers 100 times better performance.

3.2 Impact on Autonomous Rendezvous and Docking

Autonomous rendezvous and docking is an area of ongoing research and promises to enable both space exploration and on-orbit assembly and servicing. The effect of process noise on an autonomous rendezvous scenario was simulated for two satellites initially separated by about 150 meters. One satellite, the chaser, must maneuver to the location of the second satellite, the target, over the course of one orbit. Only the chaser satellite fires its thrusters, and the orbit is circular at an altitude of 335 km (the approximate altitude of the International Space Station).

For a circular orbit, the time-invariant relative dynamics of two spacecraft separated by a short distance are described by the Hill-Clohessy-Wiltshire equations [19,

75] that can be written in state space form as

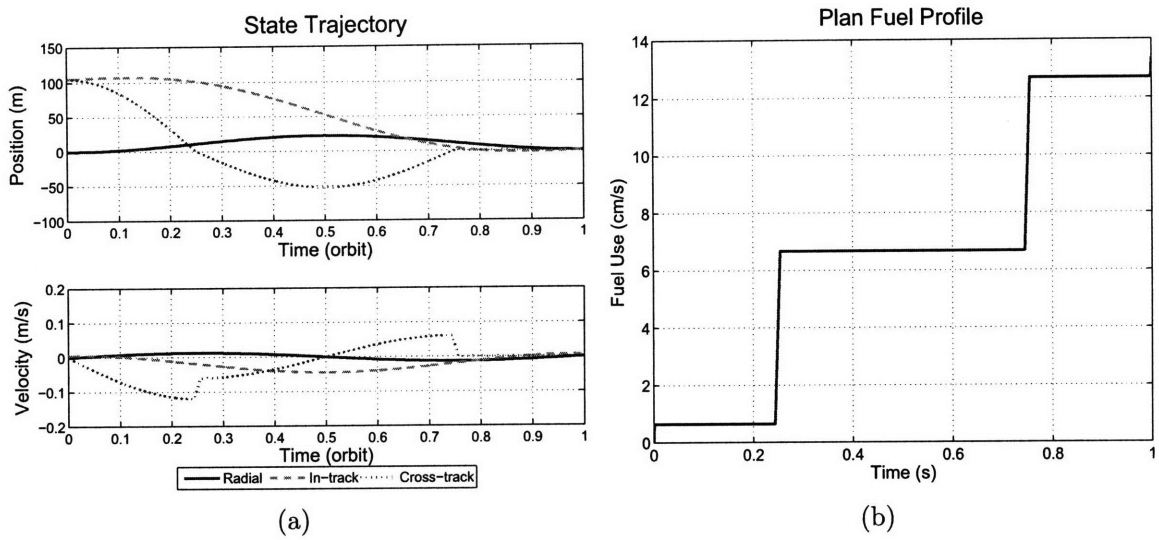
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{3.28}$$

where $n = \sqrt{\frac{\mu}{a^3}}$ is the natural frequency of the orbit, μ is the gravitational parameter, and a is the semimajor axis of the orbit. The axes x, y, and z correspond to the radial, in-track, and cross-track directions, respectively. These three directions make up the local vertical/local horizontal (LVLH), or Hill's frame (see Figure 2-3). The origin of this frame is the reference orbit, and for this study, the target spacecraft is located there. The linear nature of (3.28) enables the use of convex optimization techniques to calculate a fuel-optimized plan [28] in a manner similar to Chapter 2. The chaser's objective is to fire its thrusters and move to the origin (target satellite) over one orbit period. The orbit period is discretized into 1000 segments, with control inputs allowed during every segment. The only objectives for the rendezvous problem are to minimize fuel use and reach the target, and these objectives yield the simple optimization

$$\min_{U} = \|U\|_1 \qquad \text{subject to} \quad \mathbf{x}_f = \mathbf{x}_d \tag{3.29}$$

$\mathbf{x}_f$ is the actual state of the chaser at the end of the orbit, $\mathbf{x}_d$ is its desired state, and U is the sequence of control inputs applied at each step of the plan. The orbital dynamics enter through the constraint $\mathbf{x}_f = \mathbf{x}_d$, and the initial condition and control inputs are chosen to satisfy it.

Ideally, the control system would add no process noise, and the plan would be executed perfectly. Figure 3-3 shows the ideal trajectory. At the end of the orbit, the chaser has reached the target (Figure 3-3(a)). The total fuel consumed is 13.34 cm/s, with control applied at four different points in the orbit; two burns at the beginning



(a) (b)

Figure 3-3: The ideal rendezvous trajectory and plan.

and end of about 6.4 mm/s each, and two more burns of 6 cm/s around a quarter and three quarters of the way through the orbit. As this is the ideal performance case, all real control systems will perform worse, either by using more fuel or by failing to satisfy the terminal constraint.

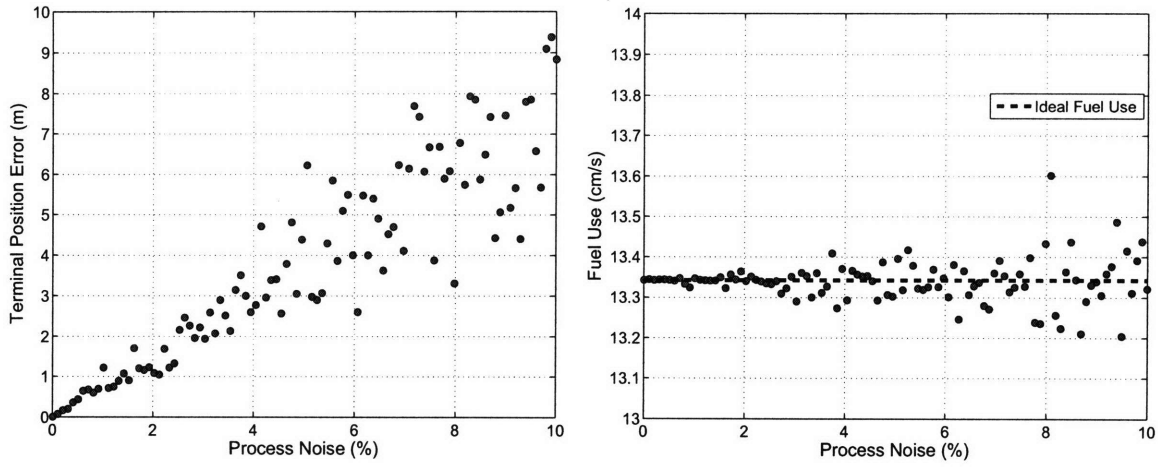
Two parameters were independently varied for the simulations: process noise and replan frequency. Process noise is modeled in the same manner as in (3.2), with a random δ for each burn which acts as a percentage error on the magnitude of each thruster firing. This way, longer burns with an inaccurate actuator lead to larger implementation errors. The random δ has a mean value $\bar{\delta}$ and a standard deviation σ_δ. If $\bar{\delta} \neq 0$ then the actuator is biased. As the process noise varies from one simulation to the next, σ_δ is the parameter that changes. Reducing $\bar{\delta}$ or σ_δ is equivalent to improving the actuator. Tested values for σ_δ ranged anywhere from 0.01% to 10% error. The replan frequency f_p is how many times the chaser satellite re-solves the optimization during a single orbit rendezvous mission. When the optimization is re-solved, the initial condition of the chaser satellite is adjusted to its current position. Errors in the initial conditions due to sensor limitations are not considered here; it is assumed that the relative positions and velocities are exactly known.

For the ideal case (with no process noise), replanning is not needed because the

control inputs are applied precisely, but when process noise is introduced, errors in the implementation of the plan will cause the chaser satellite to deviate from its expected position. If left unchecked, this error will propagate all the way to the end of the trajectory. For a rendezvous mission, if this error is large enough, both the chaser and target satellites could be put at risk. Replanning provides a way to compensate for process error by detecting deviations and modifying the rendezvous trajectory accordingly.

In general, reducing process noise results in improved performance, as does increasing the replan frequency. Figure 3-4(a) shows how varying the process noise influences the performance of the rendezvous. The position error is measured as the rectilinear distance from the chaser to the target satellite at the end of the maneuver. For this set of simulations, the actuator was unbiased ($\bar{\delta} = 0$) and only the magnitude of σ_δ was varied. The control plan was created at the start of the maneuver and executed from start to finish, without replanning. Since no replanning was done, the process noise should have no effect on average fuel use because the deviations caused by incorrect ΔV implementation are ignored; the planner does not expend fuel to try to correct them later. Figure 3-4(b) confirms this. The increased dispersion for larger σ_δ reflects the increased randomness of the thrusting, but there is no trend in the average value. For smaller σ_δ, the fuel use converges to the ideal case.

If instead, process noise is held constant while varying the replanning frequency, the performance changes as shown in Figures 3-5(a) and 3-5(b). The process noise σ_δ was held fixed at 5% while replanning frequencies from 1 to 20 times per orbit were investigated. Additionally, in these tests, a bias of $\bar{\delta} = 10\%$ was introduced. As expected, the terminal position error is improved by increasing the replanning rate because replanning allows the effects of inaccurate thrusting to be caught and corrected. This enables the chaser to still reach the target satellite at the desired time. Still, as a result of the positive bias in the actuator, the fuel use remains above the ideal value. The true cause of the problem is not in the plan, but in the actuator that poorly implements it. Continuing to use the open-loop strategy while simply increasing the replanning frequency does nothing to address this root cause. The key distinction is



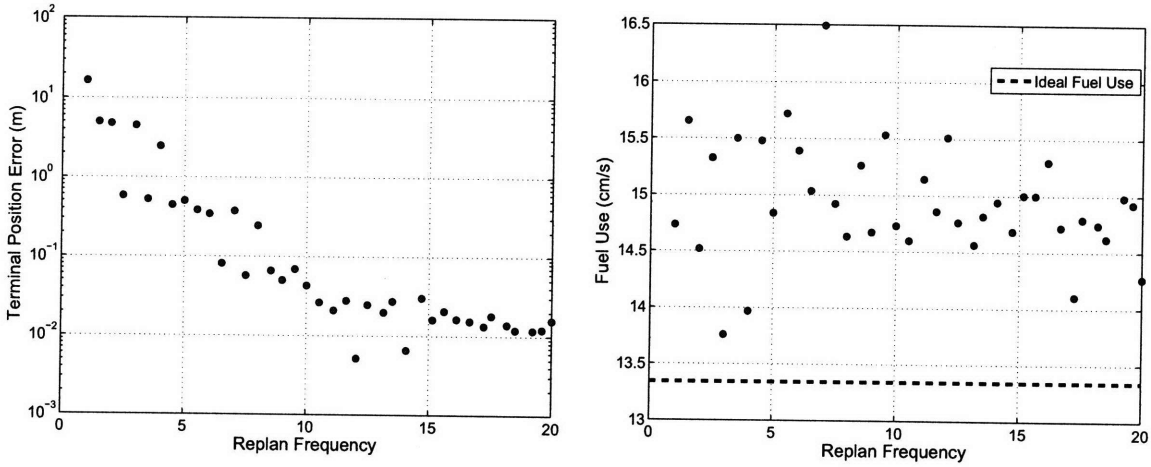
(a) Effect on terminal position

(b) Effect on fuel use

Fig. 3-4: The effect of varying process noise on rendezvous performance.

that increasing the planning rate is a *reactive* solution; the errors must happen before they can be observed and corrected by the planner. For maneuvers where a specific trajectory must be tracked closely or constraints avoided safely, replanning alone may not be enough to achieve the necessary performance. Additionally, the ability of a satellite to replan its trajectory is limited by several considerations. Accurate position and velocity estimation is required for both the chaser and the target satellites; repeatedly planning trajectories based on bad estimates is not only inefficient, but it could even result in a completely incorrect result. Moreover, the planning strategy or available computing power might place an upper bound on how often a trajectory can be replanned. Finally, thruster impingement constraints or science goals might limit the number and timing of possible firings.

On the other hand, using accelerometers to monitor the burn performance and more accurately implement ΔV commands is a *proactive* solution; errors can be prevented from even happening. Provided the accelerometers are properly calibrated, the control system in Figure 3-1 eliminates bias and significantly reduces the equivalent process noise.



(a) Effect on terminal position

(b) Effect on fuel use

Fig. 3-5: The effect of replanning frequency on rendezvous performance, with $\bar{\delta} = 10\%$ and $\sigma_\delta = 5\%$.

3.3 Impact on Formation Reconfiguration

This section investigates the performance degradation of a formation reconfiguration maneuver when process noise is added to the system. It utilizes the planner developed in the previous chapter. A formation of five spacecraft are in the elliptical orbit defined by (2.42). They begin in an in-track formation, separated by 50 m, with the leader at the center of the formation. The desired configuration is a passive aperture, with the leader at the origin and the following relative states (as in (2.20) with distance in meters):

$$\begin{aligned}
\Delta\mathbf{x}_{d_2} &= \begin{bmatrix} 50 & 25 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_3} &= \begin{bmatrix} 50 & -25 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_4} &= \begin{bmatrix} -50 & 25 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\
\Delta\mathbf{x}_{d_5} &= \begin{bmatrix} -50 & -25 & 0 & 0 & 0 & 0 \end{bmatrix}^T
\end{aligned} \tag{3.30}$$

This formation emulates an interferometer with four telescopes (the followers) and a combiner located at the center (the leader). A plan was generated using the optimization in (2.22) (with the addition of a fuel window cost (2.36)), with the scalar

settings

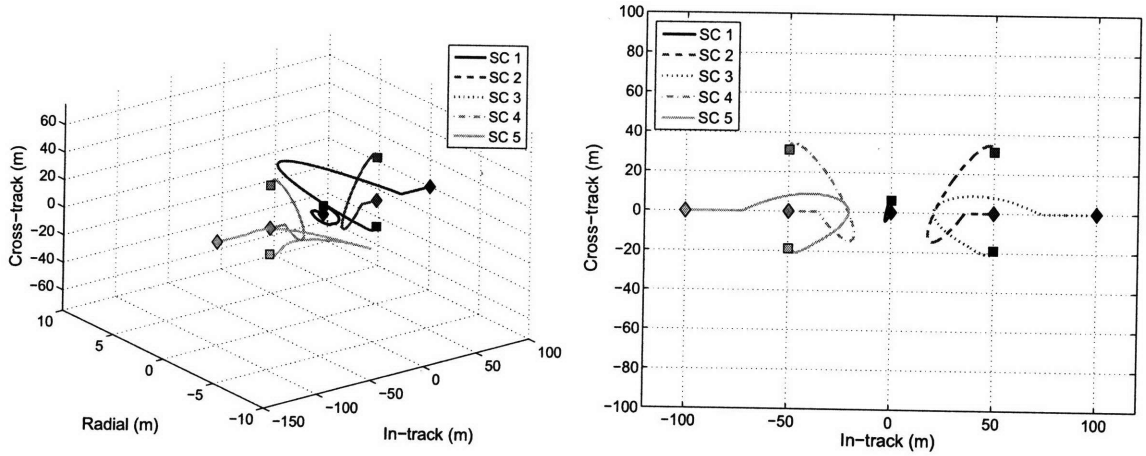
$$Q_u = 10, \quad Q_r = 100, \quad Q_d = 100, \quad Q_{f_{win}} = 1000 \tag{3.31}$$

The fuel window size was $\epsilon = 0.1$ mm/s. The nominal plan is shown in Figure 3-6. Fuel consumption for each spacecraft (Figure 3-6(b)) is 2.54 mm/s, 2.59 mm/s, 2.74 mm/s, 2.59 mm/s, and 2.74 mm/s, exhibiting good balance. The total fuel use is 13.20 mm/s. Spacecraft 3 and spacecraft 5 have further to move to reach their desired positions, and so the formation center shifts 6.06 m in the +z direction in Figure 3-6(a) to help equalize the fuel use.

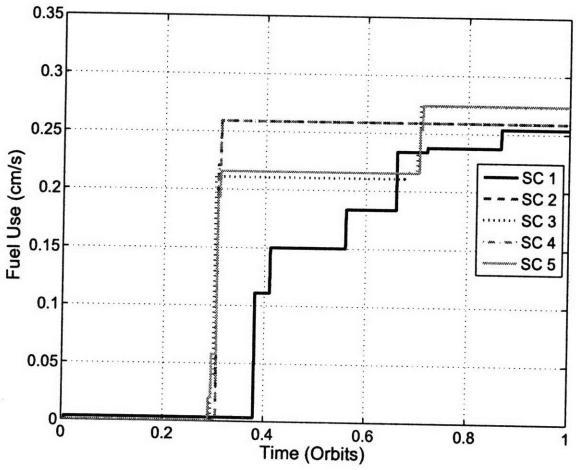
A series of simulations was run with the formation implementing the plan with varying levels of process noise (0–20%). At the end of the maneuver, the orbit was allowed to continue to propagate for an extra orbit during which no control inputs were allowed (the drift orbit). The position errors for each spacecraft, measured as the rectilinear distance from the actual state to the desired state, were summed at both times. Each process noise level was simulated 20 times and the averaged results are plotted in Figure 3-7. The circular points mark the position errors at the immediate conclusion of the plan, and the diamonds mark the position errors after the drift orbit. As expected, as the process noise increases, the formation deteriorates. However, the performance of the drift orbit also worsens; if errors are not quickly corrected, they compound. In each case, the position error grew during the drift orbit. The nominal plan does not visibly drift because the planner favors drift free orbits (recall that Q_d was set to 100 when generating this plan). Drift free orbits are very sensitive to the initial conditions and this helps explain the rapid deterioration of formations with high implementation error.

3.4 Chapter Summary

As sensing technology and the ability to design optimal trajectories for formations of spacecraft continue to improve, the need for control systems that accurately im-



(a) Absolute frame (reference orbit fixed at origin)



(b) Fuel use for the fleet

Fig. 3-6: Nominal five spacecraft in-track to passive aperture reconfiguration maneuver.

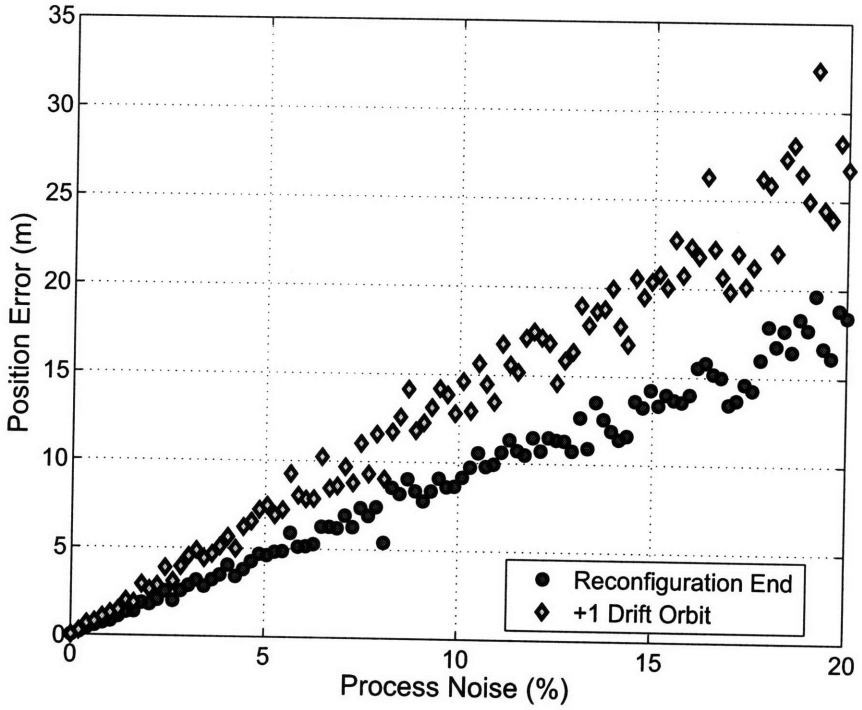


Fig. 3-7: The effect of varying process noise on a formation reconfiguration.

plement ΔV is becoming more pronounced. This chapter developed a model of ΔV implementation error and used it to investigate the dangers of poor thruster performance. For a rendezvous and docking scenario similar to what might be attempted on the ISS, small implementation errors of a few percent were found to result in terminal errors on the order of meters. For formation reconfigurations, the dangers are twofold: the desired relative geometry is not initially met, and the formation deteriorates more rapidly if it is allowed to drift. A feedback control system, using accelerometers to directly monitor thruster performance, was developed and analytical expressions describing its expected performance were obtained. For a thruster with 2% deviation from the expected thrust and realistic performance parameters, the closed-loop system improved the accuracy of the implemented ΔV by two orders of magnitude. Although simply replanning to correct errors is a possibility, allowable thrust windows are often governed by science goals or other constraints; plans must be implemented correctly without the assumption that mistakes can just be corrected later.

Chapter 4

Development of a Closed-Loop Controller for SPHERES

The previous chapter recommended the usage of accelerometers in a closed-loop control system to improve a spacecraft's implementation of ΔV commands. The actual development of such a system involves a number of challenges and practical considerations that might not be immediately apparent from that discussion. Nevertheless, IMUs have been very successful in real applications that push the boundaries of spaceflight, such as Cassini [50] and Deep Impact [51]. While those two missions used the IMU in a traditional application for monitoring ΔV, there is the potential for IMUs to function in more unexpected roles; accelerometer data from the Mars Global Survey, Mars Odyssey, and Mars Reconnaissance Orbiter has been used to model the Martian atmosphere [73]. As the technology improves, real-time measurements from accelerometers will continue to play a necessary role in formation flying missions like TPF [52]. This chapter documents the creation of a new low-level, closed-loop thrust controller for the SPHERES satellites that processes IMU measurements in real-time. First, background material and relevant specifications on the satellites are given. Details of the algorithm follow, culminating with a series of tests on the ISS verifying its performance.

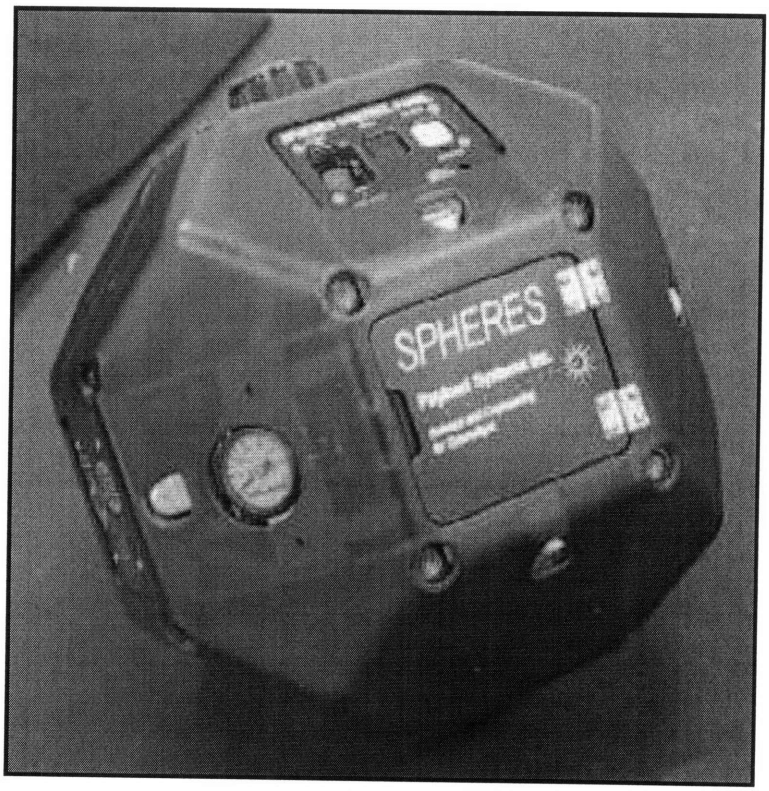


Figure 4-1: A SPHERES satellite.

4.1 SPHERES Background

The Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) are a group of nano-satellites (spheres)[1] developed by the MIT Space Systems Laboratory (SSL) to enable the development and testing of control, estimation and autonomy algorithms [76–80]. They measure approximately 22 cm across, and are capable of controlling their position and attitude in a six degrees of freedom (DOF) environment. Microgravity 6 DOF operations are conducted onboard the International Space Station (ISS), but the spheres are also able to operate on an air table in a 3 DOF laboratory setting. The satellites have fully functional power, guidance, communications and propulsion subsystems. Figure 4-1 is a photograph of a sphere. The following subsections describe the most important aspects of a SPHERES satellite as it relates to implementing closed-loop thrust control.

[1] "SPHERES" refers to the testbed as a whole, and "sphere" refers to an individual satellite.

Table 4.1: Ideal force and torque axial directions for the SPHERES thrusters.

Thruster	1	2	3	4	5	6	7	8	9	10	11	12
Force	+x	+x	+y	+y	+z	+z	-x	-x	-y	-y	-z	-z
Torque	+y	-y	+z	-z	+x	-x	-y	+y	-z	+z	-x	+x

4.1.1 Propulsion System Characterization

During the development and construction phase of the spheres, extensive testing was done on each of the various subsystems. As part of this process, the design and performance characteristics of the propulsion subsystem were thoroughly documented [81]. The position and attitude of a sphere is controlled by twelve thrusters, with two located on each face of the satellite. The propellant used by the spheres satellites is CO_2, stored as a liquid in a cylindrical tank at 860 psig. It is passed through a regulator (nominally set to 35 psig), expanded to a gas and fed to the thrusters through teflon tubing.

Table 4.1 lists each of the twelve thrusters along with their thrust directions and torque axes. Thrusters are referenced by their number. Each thruster exerts both a force and a torque on the satellite, so to perform a pure axial acceleration two thrusters must be used simultaneously. For example, to move in the +y direction, thrusters 3 and 4 would both be used. The same principle applies for pure rotations.

Several aspects of thruster performance play a role during typical SPHERES operations. First of all, while the average performance of a thruster is known with some degree of accuracy, actual performance varies from one thruster to the next. Furthermore, when multiple thrusters are opened at the same time, the thrust delivered at each thruster drops. Lastly, the thrusters are not perfectly aligned with the body axes due to practical limits in the manufacturing process. While some experimentation has been done using more sophisticated models that attempt to account for these effects, the most commonly model is based on an ideal case. This ideal case assumes that the thruster alignment is perfect, and that the provided thrust is constant at each thruster, no matter how many are open. Moreover, the mass and inertial properties of the sphere are taken as average values. Force commands are converted to thruster

times from (3.1) for translations, and for rotations

$$t_{burn} = \frac{I\Delta\omega_d}{F_T\ell} \tag{4.1}$$

where I is the moment of inertia about the axis of rotation, F_T is the force from the thruster, $\Delta\omega$ is the desired change in angular velocity, and ℓ is the moment arm of the thruster. The thruster layout on the sphere is symmetric, such that the moment arm for each thruster is 9.7 cm.

A code module called the mixer receives force and torque commands for all axes simultaneously, as vectors, and a mixing matrix relates force and torque directions to specific thrusters. Multiplying the force and torque vectors with the mixing matrix yields the force required from each thruster. These forces are then converted to on/off times. The amount of ΔV or $\Delta\omega$ that can be applied in one control cycle is limited by the available thrust, and if the maximum is exceeded in any direction, all thrust commands are scaled down so that the proportionality is maintained. More detail on the mixer and the formulation of the mixing matrix can be found in [82].

4.1.2 Instrumentation Details

Each sphere is equipped with an inertial measurement unit (IMU) comprised of three rate gyroscopes [83] and three accelerometers [84]. These are arranged so as to provide measurements for all three of the body axes [85]. While the performance of the gyroscopes is very stable, the accelerometers exhibit a ringing effect during thruster actuation that has deterred their use. The resulting oscillations take approximately 150 ms to die down. Figure 4-2 depicts the raw accelerometer readings from a ΔV event in the +x direction.

4.1.3 Global Metrology

The motion of a spheres satellite is described by the thirteen-element state vector

$$\mathbf{x} = \begin{bmatrix} r_x & r_y & r_z & v_x & v_y & v_z & q_1 & q_2 & q_3 & q_4 & w_x & w_y & w_z \end{bmatrix}^T \tag{4.2}$$

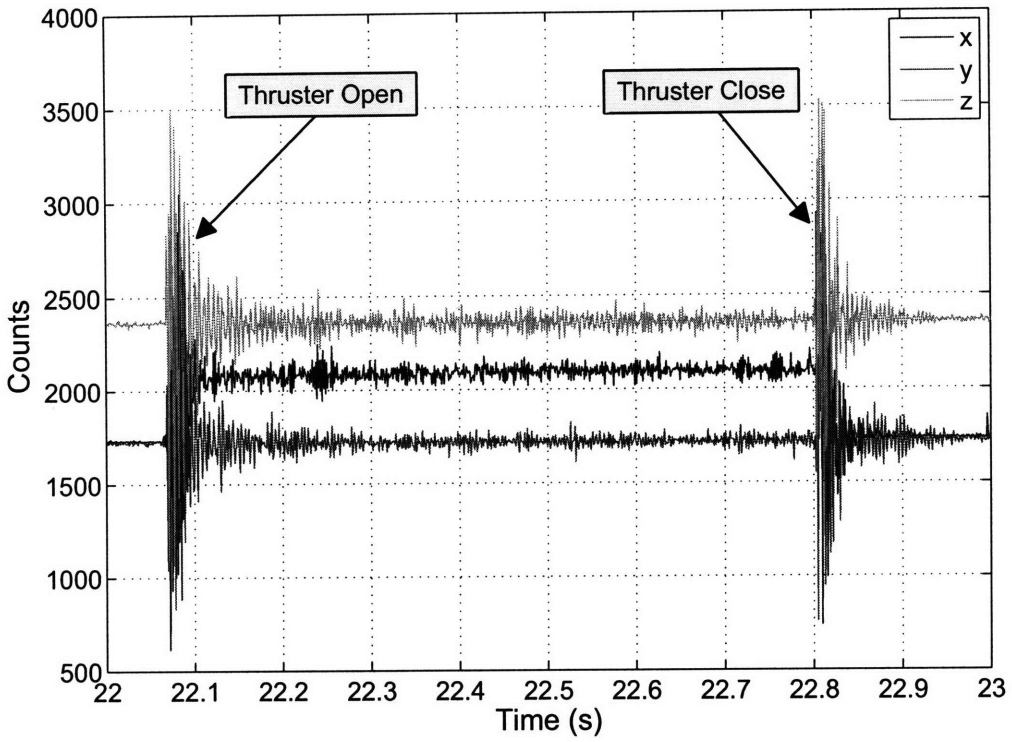


Figure 4-2: Thruster ringing effect that results from thruster actuation.

where **r** and **v** represent translational position and velocity, respectively, the four-element quaternion **q** represents the attitude, and the vector **w** represents the angular rates. An Extended Kalman Filter (EKF) on each sphere estimates its own state vector with assistance from five ultrasound beacons positioned in known locations around the test volume. When commanded by a sphere (via an infrared flashing light), the beacons ping the satellite in a specific pattern. Each sphere is equipped with ultrasound microphones on its six primary faces, and can compute the time of flight for the ultrasound pulses. The EKF uses this information, as well as measurements from the gyroscopes, to estimate the state vector. More information on the workings of the estimator can be found in [78, 80].

4.1.4 Current Control Scheme

Since the SPHERES satellites are designed to test many different aspects of formation flight, a library of functions is available to perform most of the basic tasks of operating the satellite. This helps speed the development process by allowing a scientist the

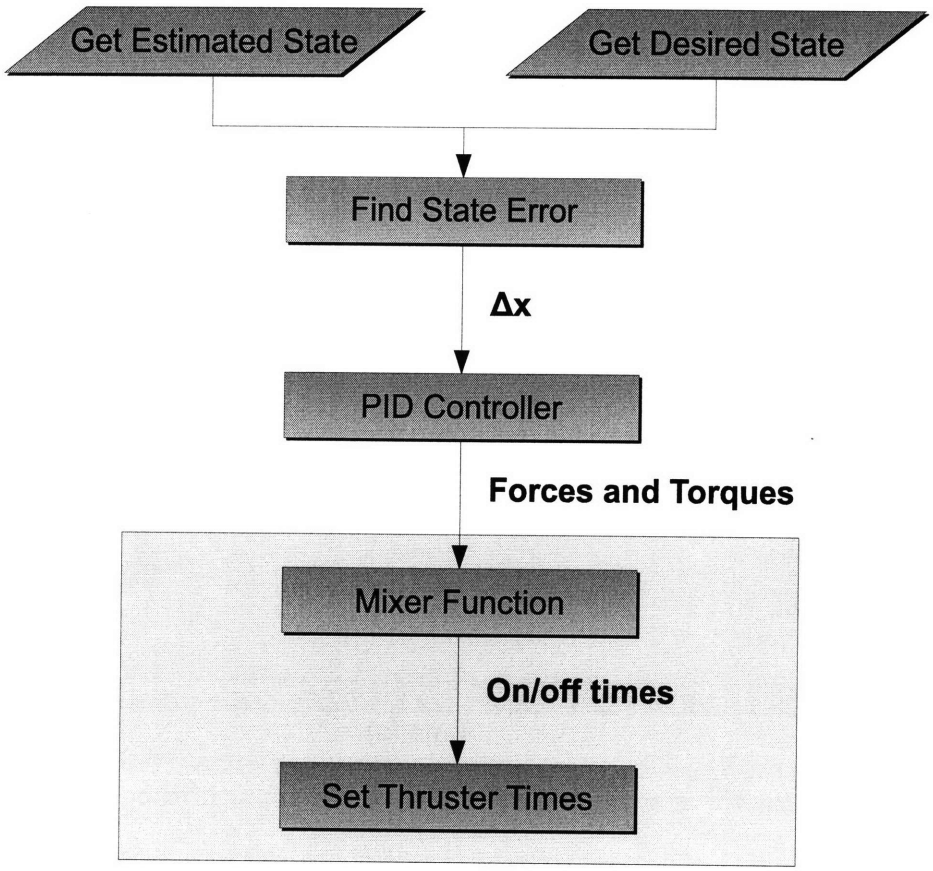


Figure 4-3: Flowchart for a typical SPHERES control cycle.

flexibility to focus on a specific aspect of a formation flying problem; time is not wasted writing code for topics that do not interest the scientist.

The control functions available as part of the default library are all based on an ideal force model. Figure 4-3 illustrates the flow process for a typical SPHERES control cycle. First, an estimate of the satellite's current state is obtained from the global estimator, as discussed in Section 4.1.3. Next, the desired state is computed. This can be accomplished in a variety of ways; if the spacecraft is following a pre-computed trajectory, this step could be as simple as a table look-up based on the elapsed maneuver time. If multiple SPHERES are flying in formation, it might involve a calculation based on the states of the other SPHERES as well as pointing constraints.

Once the desired and estimated state are known, they are passed as arguments to a function designed to determine the state error. The state error vector, $\tilde{\mathbf{x}}$ is

calculated from the estimated state vector, $\hat{\mathbf{x}}$ and desired state vector, $\mathbf{x}_d$. They may also be written as

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{r}} \\ \tilde{\mathbf{v}} \\ \tilde{\mathbf{q}} \\ \tilde{\mathbf{w}} \end{bmatrix} \qquad \hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{q}} \\ \hat{\mathbf{w}} \end{bmatrix} \qquad \mathbf{x}_d = \begin{bmatrix} \mathbf{r}_d \\ \mathbf{v}_d \\ \mathbf{q}_d \\ \mathbf{w}_d \end{bmatrix} \tag{4.3}$$

The position, velocity and rate vectors are differenced directly to obtain the error vectors.

$$\begin{aligned} \tilde{\mathbf{r}} &= \hat{\mathbf{r}} - \mathbf{r}_d \\ \tilde{\mathbf{v}} &= \hat{\mathbf{v}} - \mathbf{v}_d \\ \tilde{\mathbf{w}} &= \hat{\mathbf{w}} - \mathbf{w}_d \end{aligned} \tag{4.4}$$

However, the calculation of the attitude error involves quaternion math. Let the reference frame be A, the current estimated orientation be B, and the desired orientation be C. Then $\mathbf{q}_d$ is the quaternion describing the rotation $A \rightarrow C$, and $\hat{\mathbf{q}}$ is the quaternion describing the rotation $A \rightarrow B$. The error quaternion, $\tilde{\mathbf{q}}$ describes the rotation $B \rightarrow C$ and must be computed. It is shown in [86] that the three quaternions are related by the quaternion multiplication rule

$$\mathbf{q}_d = \mathbf{R}\tilde{\mathbf{q}} \tag{4.5}$$

In (4.5), the matrix $\mathbf{R}$ is made up of the individual elements of $\hat{\mathbf{q}}$

$$\mathbf{R} = \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 & \hat{q}_1 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 & \hat{q}_2 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 & \hat{q}_3 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 & -\hat{q}_4 \end{bmatrix} \tag{4.6}$$

$\mathbf{R}$ is an orthonormal matrix, and therefore satisfies the property $\mathbf{R}^T\mathbf{R} = I$ [87].

Multiplying both sides of (4.5) by $\mathbf{R}^T$ yields the following solutions for the error quaternion.

$$\begin{aligned}
\tilde{q}_1 &= \hat{q}_4 q_{d1} + \hat{q}_3 q_{d2} - \hat{q}_2 q_{d3} - \hat{q}_1 q_{d4} \\
\tilde{q}_2 &= -\hat{q}_3 q_{d1} + \hat{q}_4 q_{d2} + \hat{q}_1 q_{d3} - \hat{q}_2 q_{d4} \\
\tilde{q}_3 &= \hat{q}_2 q_{d1} - \hat{q}_1 q_{d2} + \hat{q}_4 q_{d3} - \hat{q}_3 q_{d4} \\
\tilde{q}_4 &= \hat{q}_1 q_{d1} + \hat{q}_2 q_{d2} + \hat{q}_3 q_{d3} + \hat{q}_4 q_{d4}
\end{aligned} \tag{4.7}$$

The result of the complete state error calculation is a vector of required corrections to the satellite's location, attitude, velocity and angular rate. This vector is passed to a control function that calculates the forces and torques to be applied to the sphere. The control function typically uses either a PD or PID control law to calculate these forces and torques from the state error, although alternative laws are possible. Attitude maneuvers (torques) and translations (forces) are usually computed by separate controllers and then combined. The final step in the process is sending the forces and torques to the mixer, which converts them to thruster on/off times. The burn is then executed open-loop.

4.2 Closed-Loop Inertial Control System

The current control scheme for the SPHERES testbed makes no use of the onboard accelerometers. Furthermore, although the gyros are used by the global estimator to obtain rate information, this is only during the estimation process. Once a desired ΔV is calculated and the mixer computes thruster firing times, these thruster times are executed open-loop; burn performance is not monitored. This can lead to inaccurate implementation of ΔV commands, such as in Figure 4-4. There, a sphere was performing a 180° rotation about its x axis. The maneuver began nominally, with the satellite accelerating to 18° per second. After the rotation was complete, a braking maneuver was performed to halt the spin of the satellite, but a thruster malfunction delivered only half the expected thrust. Since the firing times were executed

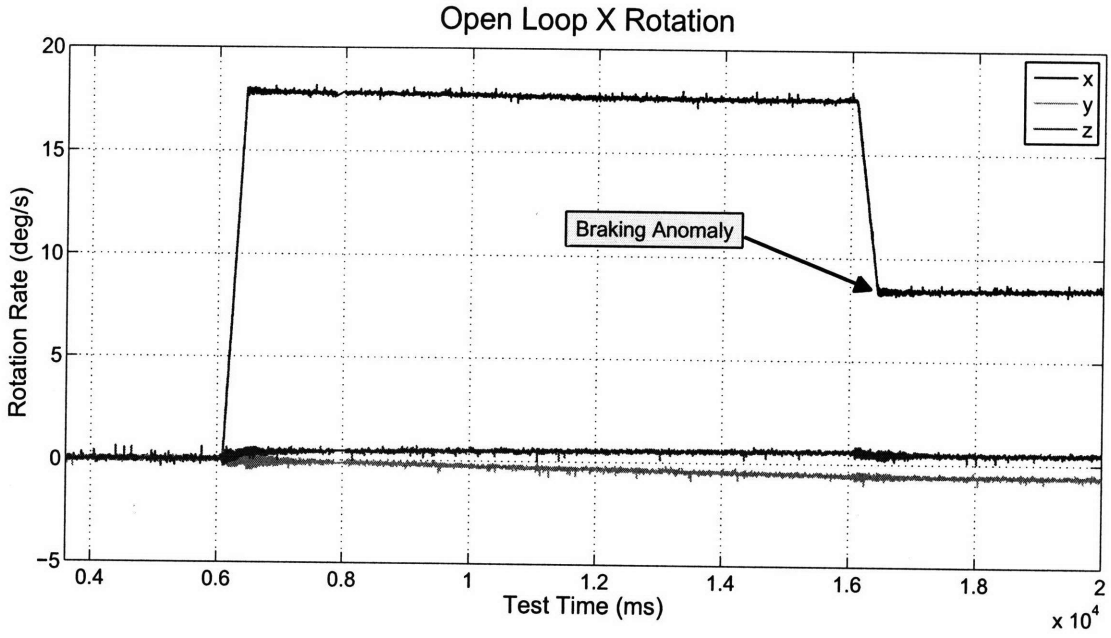


Fig. 4-4: Example of an open-loop braking anomaly from a test flight on the ISS.

open-loop, this was not detected during the burn, and the satellite was left with a significant residual spin.

For most space missions, maneuvers are planned in advance as a sequence of control inputs at specific times. In [26] and [71], formations of spacecraft are initialized by implementing impulsive burns, the magnitudes of which are computed analytically, at specific points in the orbit. In Chapter 2, convex optimization techniques were used to calculate a sequence of control inputs for a complete orbit for one or more spacecraft. In all cases, success of the maneuvers is predicated on the accurate execution of control inputs, as demonstrated in Chapter 3. Therefore, a good control system for a spacecraft must be able to accurately execute ΔV commands. This section describes the development of such a system on the SPHERES testbed.

The basic idea behind the closed-loop inertial control system is to use the IMU to actively monitor the ΔV imparted during a burn. The IMU plays an active role in determining the individual thrusters to use as well as when to terminate the burn. Rather than precompute thruster on and off times, the thrusters are only closed when a) the desired velocity is reached (within some tolerance), or b) a timeout event

occurs. Selection of the timeout criteria depends on the timing of the rest of the control cycle on the satellite.

Such a system requires several capabilities. First, useful measurements from both the accelerometers and the gyroscopes must be available in real-time. Second, the IMU measurements must be used to propagate the velocity and angular rate states at a high bandwidth. Third, the state information must be used by a low-level controller to determine when to actuate specific thrusters.

Two approaches are considered here. The first approach has been tested on SPHERES hardware and shown to provide a performance improvement over the open-loop controller. The second is an improved method, and makes use of the lessons learned in designing, implementing, and testing the first approach. It also addresses some of the weaknesses of the first method and has potential applications for estimation.

4.2.1 Filtering the Accelerometers

In order to obtain a useful and stable measure of the acceleration during a burn, the accelerometer measurements must be filtered. Several types of filters were evaluated before deciding on a specific one. The main decision was to use either an infinite impulse response (IIR) or a finite impulse response (FIR) filter, and both types were tested on actual accelerometer signals available from earlier tests on the ISS. Please see Appendix A for a brief review of digital filter concepts.

The first attempts at filtering the accelerometers on SPHERES were done with a 2^{nd} order IIR filter. This decision was made purely out of considerations for computational speed, but after extensive hardware testing, both on the satellite hardware as well as on a simulator for the TI C6701 DSP [88], it was decided that more intensive filtering was feasible. At that point, the filtering was switched to a 19^{th} order FIR filter. The performance of the two filters is very similar; Figure 4-5 compares the magnitude and phase responses of each filter, side by side. The IIR filter was created from an analog Butterworth filter, with a cutoff frequency of 50 Hz, and the phase response is fairly linear over the passband (see Figure 4-5(a)), so nonlinear phase does

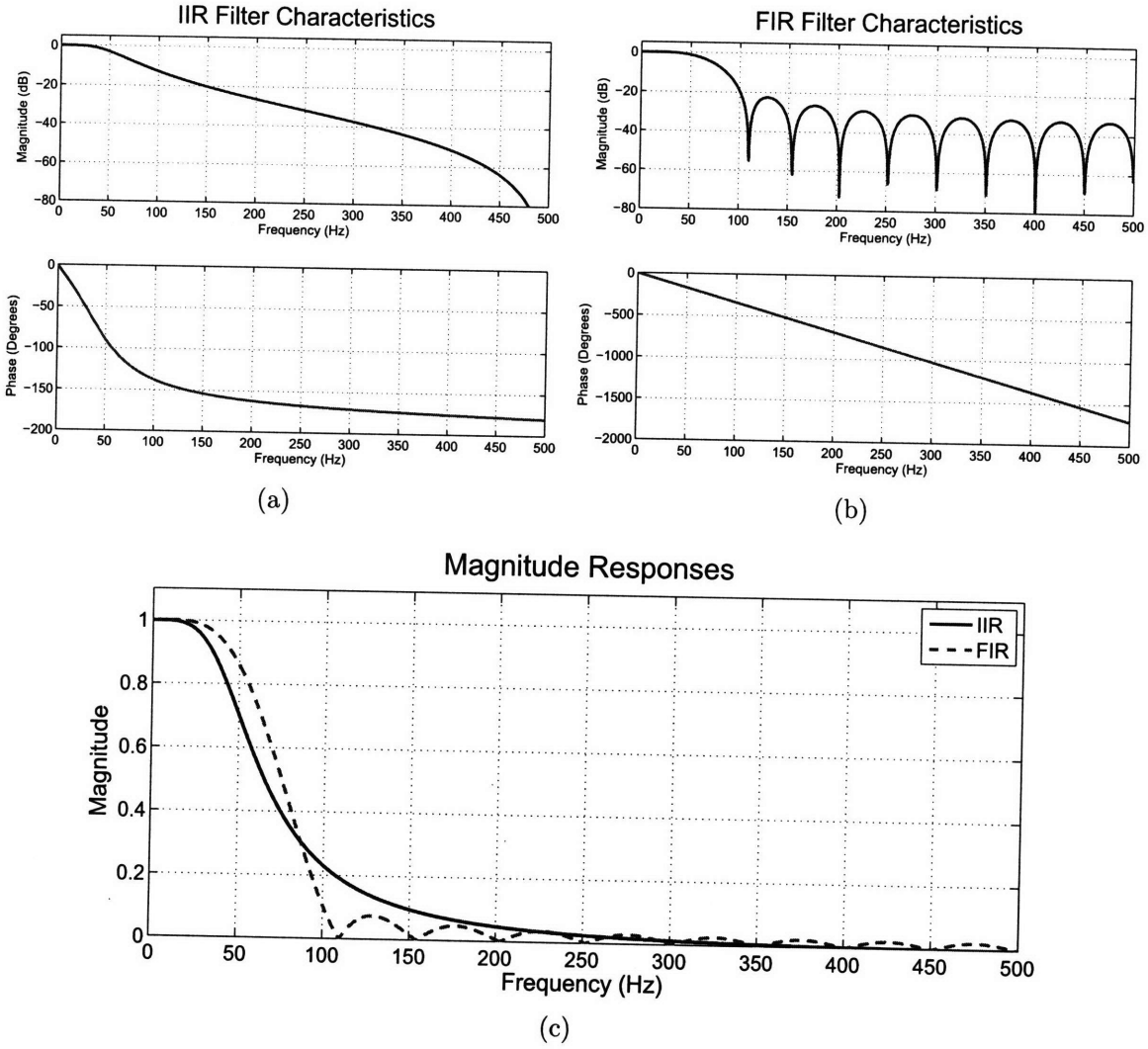


Figure 4-5: Filter characteristics for the IIR and FIR filters.

not pose much of a problem for this application. The FIR filter was created using a Kaiser-windowing method, with a cutoff frequency of 75 Hz and $\beta = 1.2$ [89].

Although the IIR filter achieves better attenuation at higher frequencies (compare Figure 4-5(b) to Figure 4-5(a)), the FIR has a narrower transition band and a flatter frequency response in the passband. Plotting the magnitude response in absolute magnitude, as in Figure 4-5(c), rather then in decibels, helps clarify this. Additionally, the IIR filter only attenuates better at frequencies greater than about 250 Hz, but a spectral analysis of the thruster ringing effect revealed that greater attenuation at high frequencies would not yield better performance.

Like some other spacecraft, the SPHERES thrusters are either on or off; when a

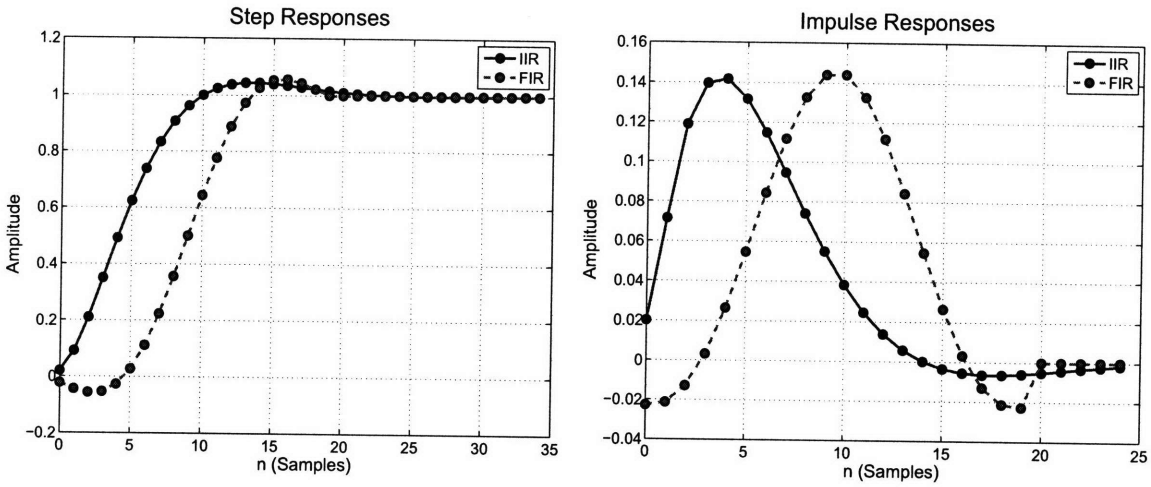


Figure 4-6: Step and impulse responses for the IIR and FIR filters.

thruster is opened, after a short transient period, the force applied reaches a steady, maximum value. This causes a sudden jump in acceleration that can be closely approximated by the step function. It is important to understand how the candidate filters react to a step function, and how quickly their outputs converge to the new input value, because it will limit the minimum burn duration that the accelerometers can monitor in real-time. For example, if a burn of 100 ms is needed, but it takes 200 ms for the filter to react to the change in acceleration, it cannot be used in a real-time feedback control system. The step responses of the IIR and FIR filters are plotted in Figure 4-6(a). The initial response of the IIR filter is faster than the FIR, but it takes longer to settle down to the correct value. Both filters also overshoot slightly and take about 20 ms to converge.

More insight into the performance of the two filters can be gleaned from the impulse response. It is desirable that the filter react minimally to an impulse; if the thruster ringing is modeled as a sequence of rapidly alternating impulses, the filter should reject them. Figure 4-6(b) shows the impulse responses for the two filters. Both magnitudes peak at around 14% of the input, yielding comparable performance. Again, the IIR filter is slightly faster, but not by enough to be significant. After 20 ms, the impulse of the FIR filter disappears completely (since it is a finite impulse response), while the output of the IIR filter continues to diminish.

Computationally on SPHERES, the 2nd order IIR filter is only about twice as fast as the 19th order FIR filter. Given the wide gap in filter orders, this is actually less of a speed advantage than might be expected. Two factors allow the FIR filter to compete:

1. The IIR filter uses 32-bit floating-point coefficients, while the FIR filter uses 16-bit fixed-point coefficients. Accuracy is not lost because the FIR filter is less sensitive to numerical quantization of the coefficients. Fixed-point arithmetic is easier on the DSP, and so the multiplications for the FIR filter take less cycles.

2. For an odd-order FIR filter, the impulse response, and therefore the coefficients, are symmetric. This allows the number of multiplications to be cut in half, because the first and last inputs to the filter ($\mathbf{x_0}$ and $\mathbf{x_{(-N)}}$) can be summed and then multiplied by a single coefficient, the second and second-to-last inputs can be summed and multiplied by a single coefficient, etc.

From the previous discussion, the expectation is that both the IIR and FIR filters will perform similarly in practice. Testing the filters on actual flight data from the satellites confirms this suspicion. Figure 4-7(a) shows some raw acceleration measurements from a ΔV in the +x direction. The on and off times of the thrusters are marked by the vertical, dashed black lines. The thrusters are commanded to open at 21.061 s, and there is a slight delay (of about 6 ms) before thrust is actually delivered. The accelerometers ring for approximately 150 ms before settling down until the thrusters are shut. The ringing occurs again after the burn is completed, but this does not matter as it is assumed the spacecraft does not accelerate once the thrusters are closed. The acceleration for the raw measurements reaches peak values of ± 150 mm/s^2, which is clearly incorrect, as the maximum acceleration for a sphere is around a third of that.

Filtering the accelerations helps smooth out the signal, as seen in Figures 4-7(b) and 4-7(c). The erroneous peak values are reduced, although there is some initial non-minimum phase behavior immediately after the thruster opens. This behavior could not be eliminated solely by using a low-pass filter designed to run at real-time

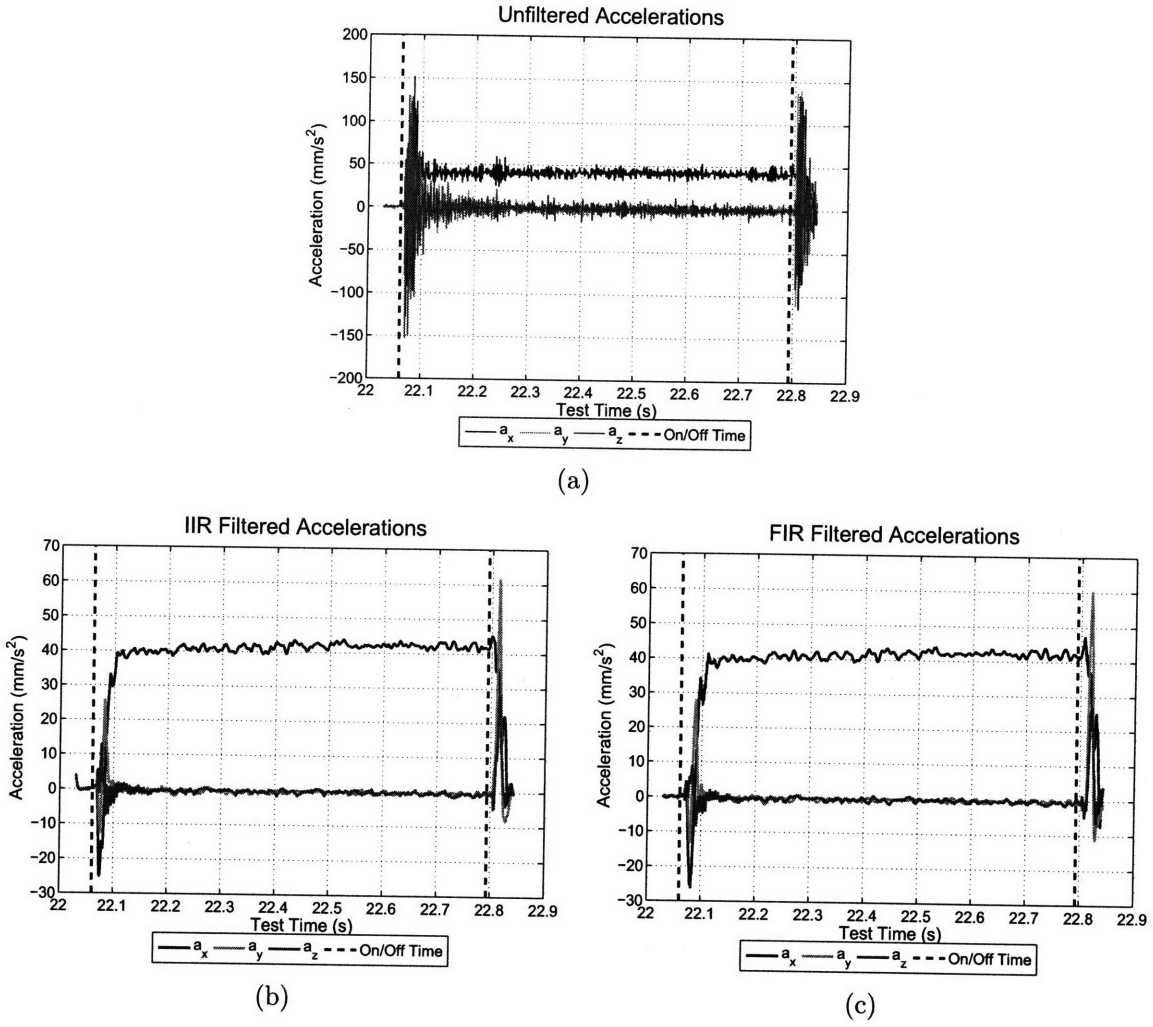


Fig. 4-7: Raw and filtered acceleration measurements for a burn in the +x direction, taken from an experiment on the ISS.

on the satellite; the ringing is too prominent. However, after 40 ms, the filters have converged to an acceleration of just over 40 mm/s^2, and the oscillations present in the filtered measurements are significantly less than those in the raw measurements. It is interesting to see that the performance of the IIR and FIR filters are so similar; in practice, they are probably equally effective. Despite this, a few slight differences are noticeable upon close inspection.

4.2.2 Effects of Spacecraft Spin on Accelerometer Readings

Unfortunately, measuring the true acceleration of a sphere is not as straightforward as filtering a few accelerometers. Each of the three accelerometers measures the acceler-

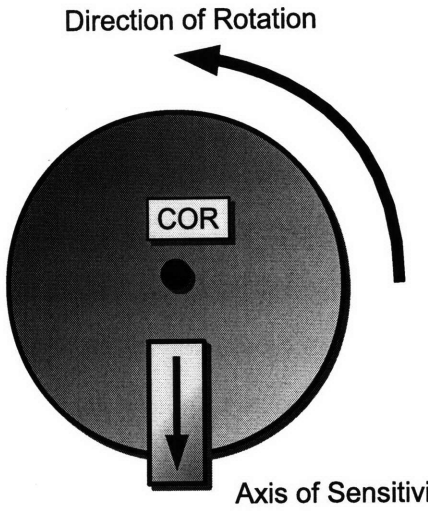


Fig. 4-8: A rotating spacecraft with an accelerometer mounted away from the center of rotation.

ation along a single axis, in the frame of the accelerometer itself. If the accelerometers were all located at the satellite's center of rotation, this would not be a problem, but due to design constraints, they are not. Figure 4-8 depicts a spinning spacecraft with an accelerometer mounted at a location away from the center of rotation (COR). Let ω be the spin rate, $\mathbf{r}$ be the vector from the COR to the accelerometer, $\hat{\mathbf{s}}$ be the axis of sensitivity (a unit vector) for the accelerometer, and θ be the angle between the vectors $\mathbf{s}$ and $\mathbf{r}$. The acceleration of the accelerometer is then given by

$$\mathbf{a} = -\omega^2 \mathbf{r} \tag{4.8}$$

Unless the accelerometer's axis of sensitivity is perpendicular to the radial direction, it will measure a nonzero acceleration due purely to the rotation of the satellite. This is the centripetal acceleration of the accelerometer and does not mean the satellite's linear velocity is changing, so to avoid erroneous acceleration measurements, it must be accounted for. The bias of the accelerometer, caused by the centripetal force, can be written as

$$a_{accel} = |\mathbf{a}| \cos\theta = \mathbf{a} \cdot \hat{\mathbf{s}} \tag{4.9}$$

which makes use of the dot product property

$$\cos\theta = \frac{\mathbf{a}\cdot\hat{\mathbf{s}}}{|\mathbf{a}||\hat{\mathbf{s}}|} \tag{4.10}$$

For the three-dimensional problem, the linear acceleration $\mathbf{a}_{accel}$ experienced by an accelerometer mounted at location $\mathbf{r}_a$ on the spacecraft is

$$\mathbf{a}_{accel} = \mathbf{a}_{sc} + \alpha \times \mathbf{r}_a + \omega \times (\omega \times \mathbf{r}_a) \tag{4.11}$$

In this equation, $\mathbf{a}_{sc}$ is the linear acceleration of the spacecraft, α is the angular acceleration vector of the spacecraft, and ω is the angular velocity vector of the spacecraft. For the implementation of a ΔV command, the linear acceleration of the spacecraft is what must be measured, so the effect of any angular acceleration or angular velocity must be accounted for in (4.11). Subsequently, information on the angular rates and accelerations is needed in addition to accelerometer measurements. On SPHERES, this additional information is provided by the gyroscopes.

The impact that a spacecraft's spin has on the accelerometer readings can be significant. Figure 4-9(a) shows IMU data from a sphere performing a rotation about its +z axis. The data is unfiltered, so the prominent thruster ringing discussed in Section 4.2.1 is present. Note that the spacecraft is *not* accelerating in a linear direction, but the x and y measurements read between 2 and 3 cm/s. The bias caused by the spin is graphed in Figure 4-9(c). The contribution from the angular acceleration, $\alpha \times \mathbf{r}_a$, accounts for a constant offset, but as the angular rate of the sphere increases throughout the burn, the contribution of $\omega \times (\omega \times \mathbf{r}_a)$ becomes larger and further increases the biases. The z bias remains very close to zero throughout since that is the axis the spacecraft is trying to rotate about. With the bias is accounted for, the linear acceleration of the spacecraft is given by Figure 4-9(b), and as expected, the accelerations in the x and y directions are now closer to zero.

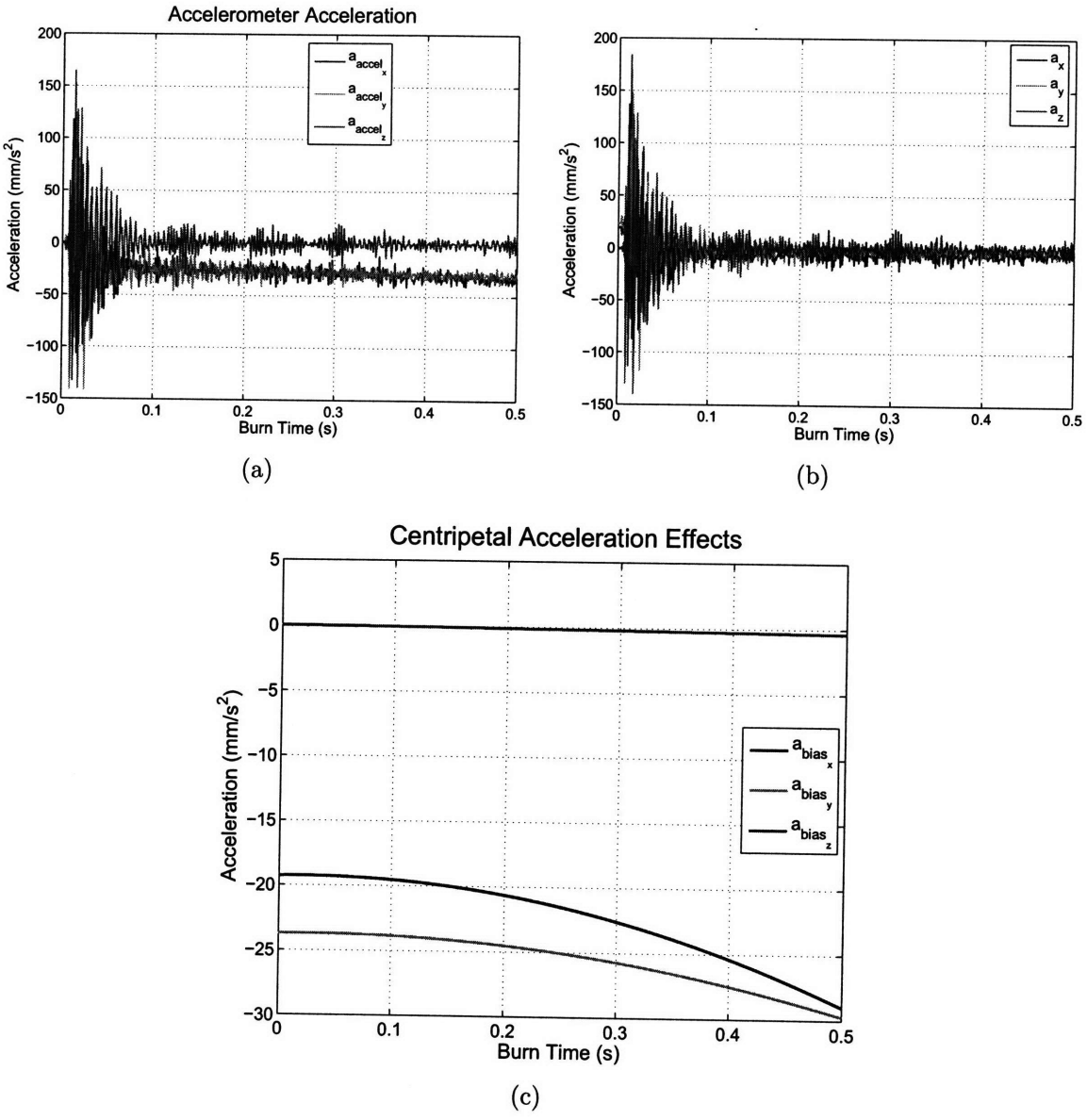


Fig. 4-9: Data from a SPHERES satellite performing a rotation about the +z axis. In 4-9(a), centripetal effects are ignored, and in 4-9(b) a correction based on (4.11) is applied. The expected bias of the accelerometers is shown in 4-9(c).

4.3 Recursive Least-Squares Estimator Algorithm

The approach described in Section 4.2 was tested successfully on the ISS, but it is has some weaknesses. Because the filtered IMU readings take about 40 ms to converge, this is the minimum burn length that can be executed using closed-loop control with accelerometers. Although not a major issue when using a sphere on the air table, since the accelerations are low and burns are longer, it is a significant limitation on-orbit, where the satellites are much more agile. This section discusses an alternate algorithm for using IMU data to improve ΔV implementation and presents some results.

The previous closed-loop algorithm directly integrates spacecraft accelerations in real-time to measure the burn ΔV and requires minimal knowledge of the thruster performance characteristics. For example, the algorithm can be used for both throttleable thrusters or on/off thrusters, and knowledge of the expected thrust is unnecessary. However, it is likely that this this information is actually available — after all, the open-loop controller is completely based on an expected thrust. Incorporating these known thruster characteristics into the control system is advantageous, and that is what the alternate algorithm does.

The recursive least-squares estimator is derived in detail in [72], and the relevant results are summarized below. An estimate of the state at the current time, $\hat{\mathbf{x}}_k$, is constructed from the estimate of the state at the previous time, $\hat{\mathbf{x}}_{k-1}$ and a new measurement, $\mathbf{z}_k$, as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k-1}) \tag{4.12}$$

where $\mathbf{K}_k$ is the estimator gain matrix and $\mathbf{H}_k$ is the observation matrix. The observation matrix maps the system state to the available sensors, and the gain matrix is

$$\mathbf{K}_k = \mathbf{P}_{k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4.13}$$

This gain matrix is computed at every iteration from the covariance matrix of the previous estimate, $\mathbf{P}_{k-1}$, and a matrix $\mathbf{R}_k$ of the expected squared errors of the newest

measurement. After the state estimate is updated from (4.12), the covariance of the new estimate is given by

$$\mathbf{P}_k = (\mathbf{P}_{k-1}^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k)^{-1} \tag{4.14}$$

If the force applied by a thruster is constant over the duration of a burn, as with the SPHERES satellites, the recursive least-squares estimator can be applied to provide a continuously improving estimate of the ΔV, in real-time. During a burn, the algorithm estimates the state

$$\mathbf{x} = \begin{bmatrix} \mathbf{a}_{sc} \\ \alpha \\ \omega_0 \end{bmatrix} \tag{4.15}$$

using the sphere's accelerometer and gyroscope measurements. The linear acceleration of the spacecraft is $\mathbf{a}_{sc}$, the initial angular velocity is ω_0, and the angular acceleration is α. All are 3×1 vectors with components along the x, y, and z axes. From $\mathbf{a}_{sc}$ and the elapsed burn time Δt, the linear ΔV of the spacecraft can be calculated as

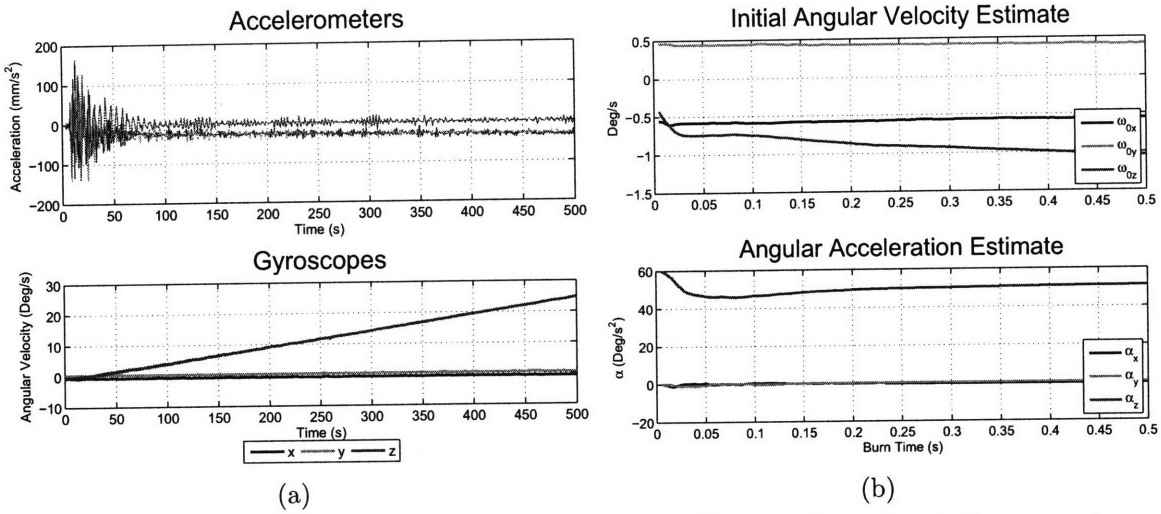
$$\Delta V = \mathbf{a}_{sc} \Delta t. \tag{4.16}$$

The initial angular velocity is required because there is no sensor on SPHERES that can directly measure α. Instead, a line is fit to the angular rate measurements obtained from the gyros. This approach is valid because the thrust, and therefore angular acceleration, is approximately constant. It works well in practice and eliminates the need to differentiate potentially noisy gyroscope signals to obtain α.

Another advantage of the recursive least-squares estimator is that it can take advantage of an initial guess $\hat{\mathbf{x}}_0$ for the state. The initial guess for ω_0 is fairly accurate since it is measured directly by the gyros, and this knowledge is incorporated into the initial covariance matrix, $\mathbf{P}_0$. When starting a burn, the algorithm predicts the value of the other elements of $\hat{\mathbf{x}}_0$ from a lookup table that is indexed by the ΔV direction for each axis. There are three axes for rotations, three for translations, and the ΔV about each axis can be either negative, zero, or positive. To store all combinations

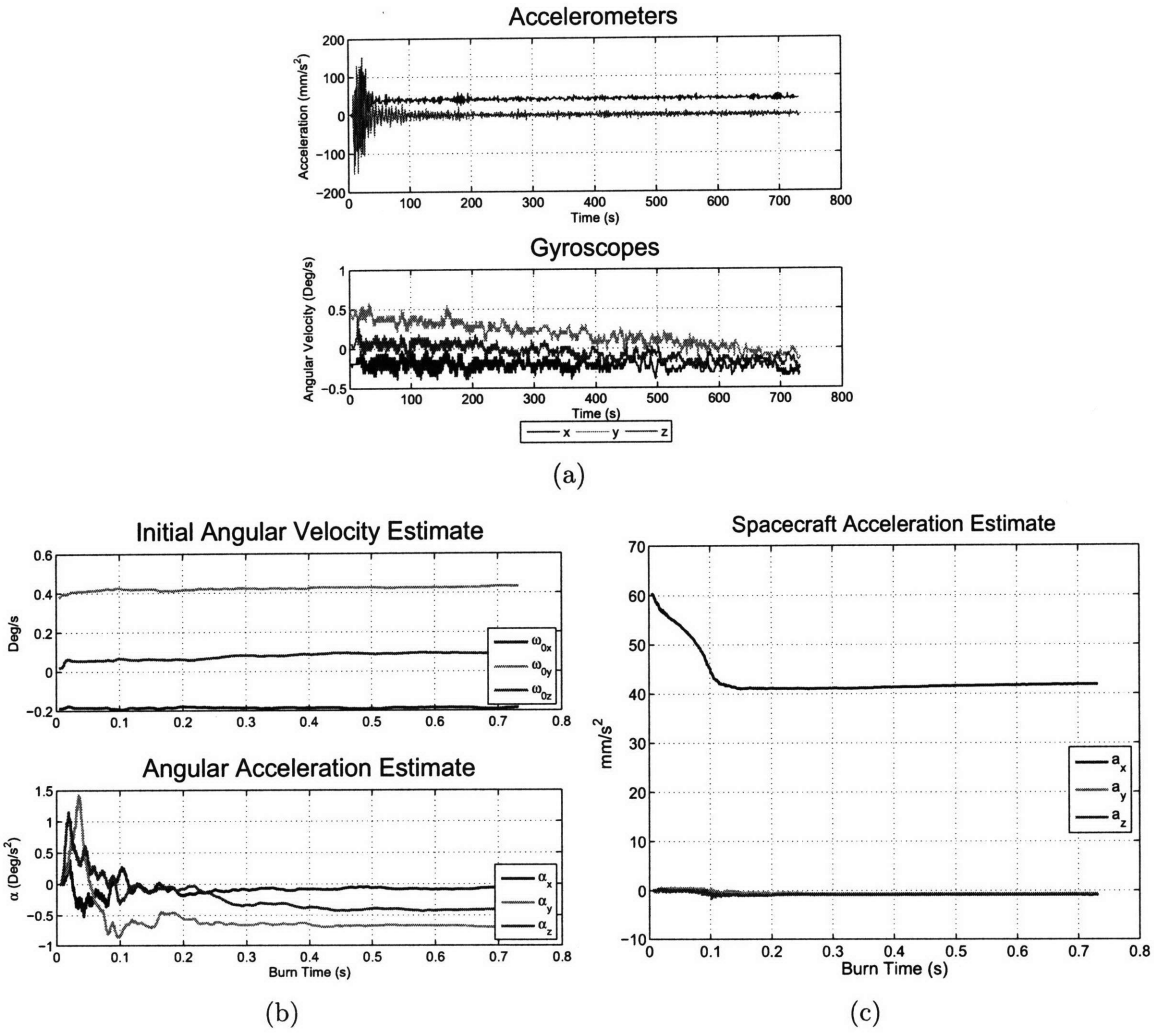
requires a table with $6^3 - 1 = 215$ entries (-1 because the case where all six axes are off is omitted). There is a one-to-one mapping between ΔV axes and thruster combinations, so a table of this size contains information on any thruster combination used in practice. Due to memory limitations on the SPHERES satellites, the lookup table on the hardware implementation was limited to two separate tables: one for pure rotations and one for pure rotations. This reduces the total storage size to $2(3^3 - 1) = 52$ entries. Each entry in the lookup table contains the estimated linear and angular accelerations for that particular combination of thrusters, as well as the covariance for those estimates. The lookup table is initialized with predicted accelerations from the ideal open-loop thruster model. Starting from the estimates in the table, as more measurements are received, the estimator dynamics act to correct errors in $\hat{\mathbf{x}}_0$. If $\delta = 0$ in (3.2), then the measurements will support the initial guess, and the estimate given by (4.15) will remain steady. For nonzero δ, the accelerometer readings will detect the discrepancy, and correct the state estimate. At the conclusion of each thruster burn, the lookup table is updated with the latest (best) estimate and covariance. However, there is a lower bound placed on the covariances so that the algorithm maintains some sensitivity to the most recent IMU measurements. Over time, as the satellite executes more burns, the lookup table is populated with more accurate information on thruster performance. Since the lookup table is used as the initial guess, the implementation of ΔV commands is improved even for very short duration burns. There is no more 40 ms minimum burn duration. Moreover, since the the estimator remains sensitive to recent IMU measurements, it will track long term trends in thrust performance and remain accurate.

Although this algorithm has not yet been tested on the ISS, it has been used to post-process real IMU burn data. The results for a rotation are shown in Figure 4-10; raw IMU measurements are shown in Figure 4-10(a) and the recursive estimates of ω_0 and α are plotted in Figure 4-10(b). Similarly, Figure 4-11 shows the performance for a linear acceleration, with the addition of Figure 4-11(c) for the recursive estimate of $\mathbf{a}_{sc}$. In both tests, the algorithm converges rapidly, and improvements to the initial guess begin immediately after the burn starts.



(a) (b)

Fig. 4-10: Recursive least-squares estimation of a rotation about the +z axis.



(a) (b) (c)

Fig. 4-11: Recursive least-squares estimation of an acceleration in the +x Direction.

4.4 Results of SPHERES Tests

Several tests comparing the performance of the closed-loop and open-loop algorithms were conducted on the air table at MIT as well as in microgravity on the ISS. The results indicate that the closed-loop algorithm offers performance improvements over the open-loop strategy, supporting the analysis of the previous sections. This section discusses the results of the SPHERES tests.

The maneuvers for these tests were kept as simple as possible, for two reasons: first, to allow a straightforward comparison between algorithms, and second, to allow IMU data to be downloaded following each test. The IMU data consists of the raw accelerometer and gyroscope readings and allows verification that the closed-loop ΔV control algorithm behaved appropriately. Downloading the data from a sphere is time-consuming; therefore, the duration of the maneuvers was kept short. Two types of comparison tests were performed: translations, and rotations. The translation tests have the satellite accelerate in a linear direction, and the rotation tests have the satellite change its angular velocity. The latter case could be interpreted as an attitude correction maneuver.

4.4.1 Translation Tests

Translation tests consisted of two burns with a desired ΔV of 3 cm/s. The first was along the +x body axis, and the second was along the -x body axis. The actual delivered ΔV was measured in two ways. In the first, the downloaded IMU data was integrated to obtain a solution using the same algorithm used in the closed-loop controller and discussed in Section 4.2. In the second, the output of the global estimator (see Section 4.1.3) was used. The estimator was allowed approximately 6 seconds to re-converge on a velocity after a burn was performed.

Figures 4-12 and 4-13 show the measured ΔV for the +x burn for the closed-loop and open-loop test cases on the air table. In both cases, the leftmost graph shows the axial and angular ΔV as measured by the IMU throughout the burn. The angular rates are used to subtract the centripetal acceleration as in (4.11). The rightmost

graph shows the global metrology system's estimated body velocities for the sphere. Note that the global metrology system only runs at around 4 or 5 Hz, so the frequency of measurements is much less than for the IMU data, which runs at 1 kHz. Also, the velocities given by the estimator are absolute, whereas the measurements from the IMU are relative (and initially zeroed at the start of a burn). As expected, the closed-loop results coincide with the IMU measurements very precisely. However, the global estimator results give an additional, independent perspective. Initially after the burn, the estimator shows a significant overshoot in the measured x velocity, but over the next 5 seconds, it steadily decreases. The thruster integration law the estimator uses for state propagation is to blame for the overshoot. When a sphere fires its thrusters, the estimator attempts to predict the effect of the thruster firing by using the same model used in the open-loop mixer. Therefore, if the acceleration delivered is less than expected by the model (i.e., $\delta < 0$), the initial guess of the estimator for the applied ΔV will be too high. Over the next few seconds, as the estimator receives more ultrasound ranging measurements, it slowly corrects this error. Overshoot behavior is visible in both the closed-loop and open-loop test cases, and the true ΔV applied for each of the burns likely lies somewhere between the results given by the closed-loop algorithm and those given by the global estimator. These results are collected in full in Table 4.2. This table includes the thrust duration (ΔT), the measured ΔV, the average measured acceleration delivered by the thrusters, and the percentage error according to both the closed-loop algorithm and the global estimator. The open-loop mixer clearly has a tendency to deliver less than the desired ΔV, as this trend is reflected by both the IMU and estimator measurements. One contributing factor for this behavior is that there is some friction present on the air table. Since the open-loop model does not account for this friction, it consistently fails to provide enough acceleration. The closed-loop algorithm automatically compensates for friction, and the results are noticeably better. Error for the open-loop tests was around 20%, while for the closed-loop tests it was reduced to about 1%.

After the success of the tests on the air table, they were performed on orbit. Those results are tabulated in Table 4.3, and Figures 4-12 and 4-13 depict the measured ΔV

for the +x burn for the closed-loop and open-loop test cases on the ISS. For these tests, the discrepancy between the global estimator and the closed-loop algorithm was larger than for the air table tests. The closed-loop algorithm maintains that the applied ΔV is very close to the commanded value, but the estimator gives errors of 8.8% and 15.5%. This is still better than the open-loop errors of -16.3% and -12.4% (according to the estimator), or -24.1% and -22.1% (according to the closed-loop algorithm). One possible explanation is that the estimator was not given enough time to reconverge on the new velocity after the burns were performed. Either way, once again the open-loop performance is conclusively shown to underperform and not provide enough ΔV.

4.4.2 Rotation Tests

The rotation tests also consisted of two burns. The first burn applied a ΔV of 25°/s about the +z axis, and the second applied a ΔV of 25°/s about the -z axis. For the air table tests, the IMU results for a rotation are shown in Figure 4-16 for the closed-loop algorithm, and Figure 4-17 for the open-loop algorithm. Numerical data is presented in Table 4.4. The open-loop rotation results are consistent with the translation tests — the algorithm delivers less than the desired ΔV. However, for rotations, the percentage error is even larger than for translations, and once more, friction on the table likely led to less than the expected acceleration. The closed-loop algorithm performed very well on the rotations, achieving ΔV within 1% of the goal. For on-orbit tests, the results are tabulated in Table 4.5, with example burns in Figure 4-18 and 4-19. The open-loop controller performs better because friction does not play a role as it did on the air table. Still, the benefits of closed-loop control remain; the open-loop algorithm undershoots by 8.6% and 4.1% while the closed-loop overshoots by just 1.8% and 2.1%, a noticeable improvement. These results were measured by the global estimator, not the closed-loop algorithm, because the IMU data for one of the open-loop test burns was lost due to an RF-communications lapse. Fortunately, the global estimator estimates angular rates from the gyroscope readings. Since the gyroscopes directly measure these rates at any given instant, there is no need to wait

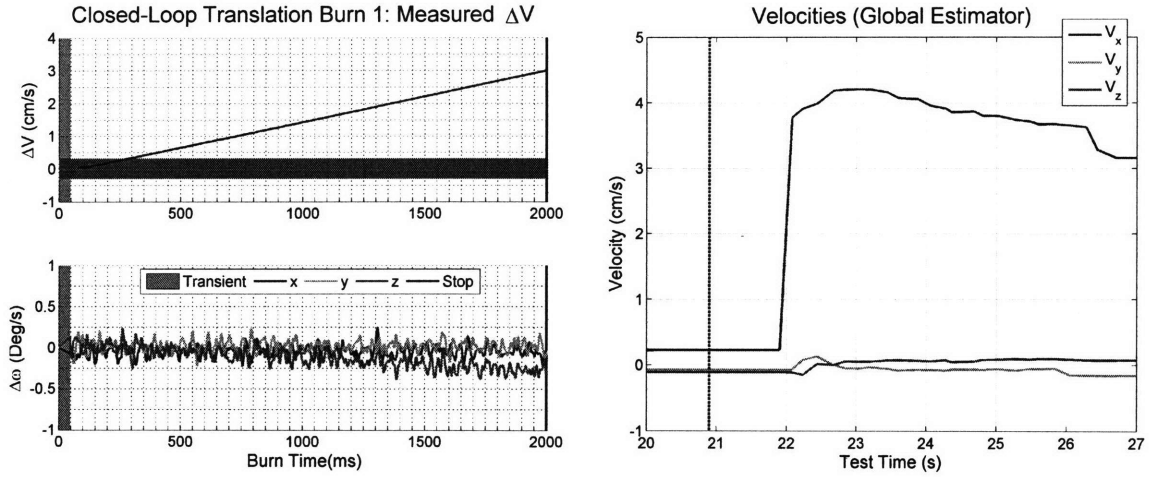
for the estimator to reconverge after a rotation burn. Therefore the confidence level for the measurements in Table 4.5 is high.

4.5 Chapter Summary

With the number of real world missions successfully relying on accelerometer and IMU measurements for accurate ΔV implementation increasing, this chapter detailed the design of a low-level, closed-loop thrust controller for the SPHERES testbed. In constrast to the higher level focus of Chapter 3, questions regarding specific implementation issues as they pertained to SPHERES were discussed and solution strategies were presented. Two different algorithms were described. The first used low-pass filtering to process and directly integrate IMU measurements in real-time. It significantly improved translation and rotation ΔV implementation on the SSL's 3-DOF testbed at MIT, and also showed promise in 6-DOF tests on orbit. Based on the experimental results from the first algorithm, an improved method was presented that used a recursive least-squares filter to provide a continuously improving estimate of thruster performance during a burn. The updated algorithm maintained a lookup table of expected accelerations that, when used as initial guesses, eliminated the minimum burn length limitation of the first algorithm.

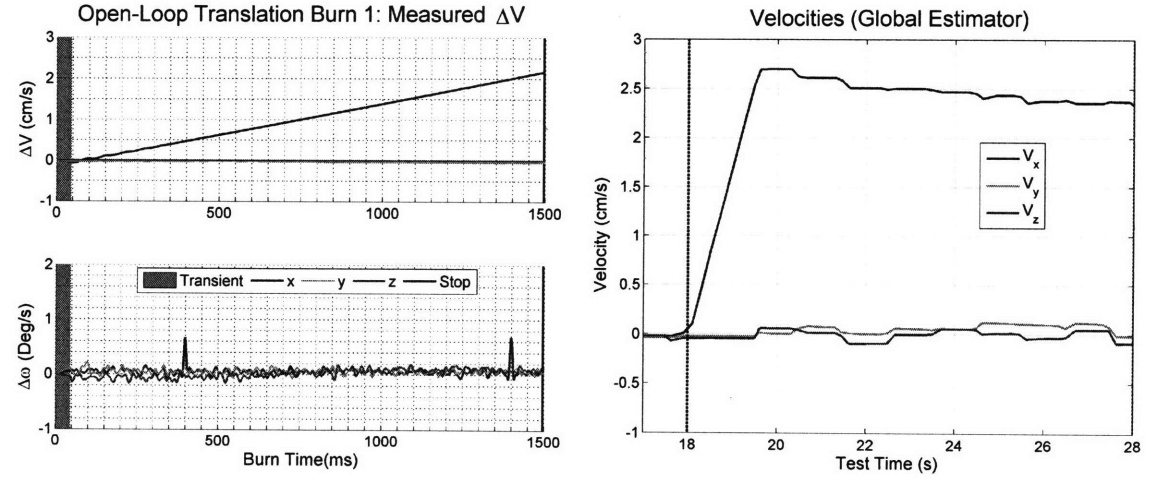
Table 4.2: Test results summary of open-loop and closed-loop translation burns on the air table.

		Closed-loop Algorithm				Global Metrology Estimate			
	ΔT (ms)	ΔV		a		ΔV		a	
Closed-loop Translation	1999	3.00 cm/s		1.50 cm/s^2	0.1%	2.94 cm/s		1.47 cm/s^2	-2.2%
Open-loop Translation	1500	2.17 cm/s		1.45 cm/s^2	-27.7%	2.37 cm/s		1.58 cm/s^2	-20.9%
Closed-loop Translation	1939	-3.00 cm/s		-1.55 cm/s^2	0.1%	-3.03 cm/s		-1.56 cm/s^2	1.0%
Open-loop Translation	1500	-2.21 cm/s		-1.47 cm/s^2	-26.5%	-2.00 cm/s		-1.33 cm/s^2	-33.5%



(a) (b)

Fig. 4-12: Results of a closed-loop translation burn on the air table as seen by the algorithm and the global estimator.

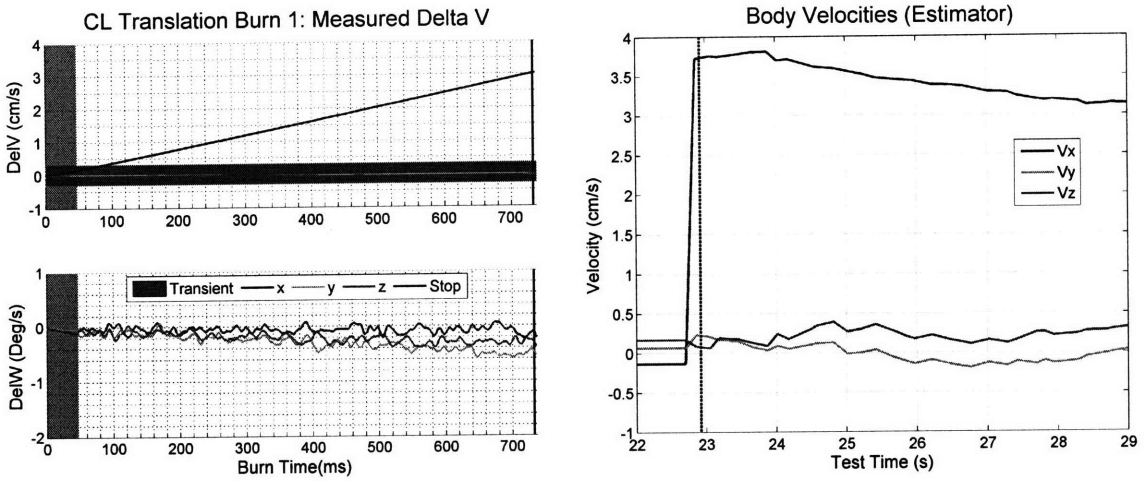


(a) (b)

Fig. 4-13: Results of an open-loop translation burn on the air table as seen by the algorithm and the global estimator.

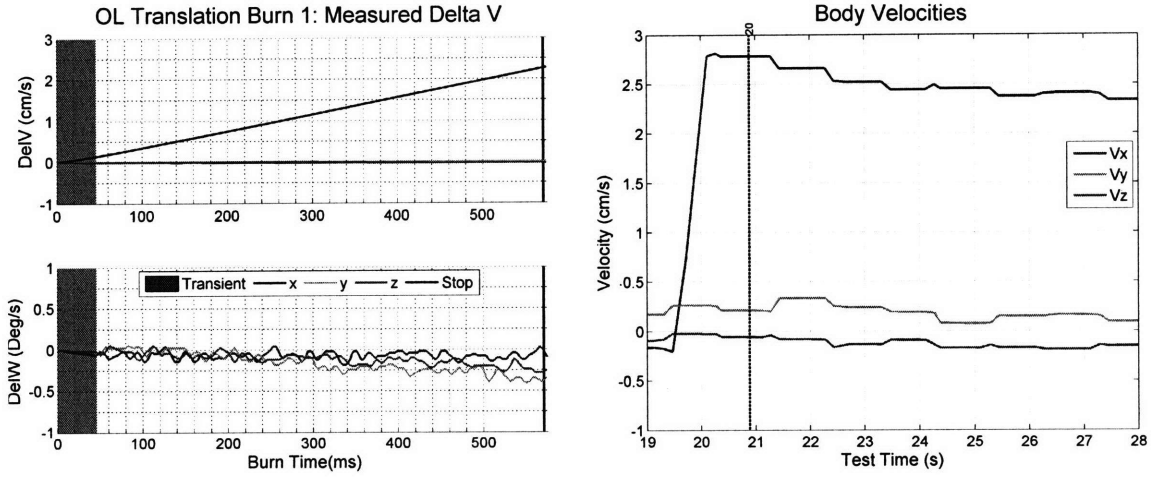
Table 4.3: Test results summary of open-loop and closed-loop translation burns on the ISS.

		Closed-loop Algorithm				Global Metrology Estimate			
	ΔT (ms)	**ΔV**		**a**		**Error**	**ΔV**	**a**		**Error**
Closed-loop Translation	733	3.00	cm/s	4.10	cm/s^2	0.1%	3.26 cm/s	4.45	cm/s^2	8.8%
Open-loop Translation	575	2.28	cm/s	3.96	cm/s^2	-24.1%	2.51 cm/s	4.36	cm/s^2	-16.3%
Closed-loop Translation	732	-3.00	cm/s	-4.10	cm/s^2	0.1%	-3.47 cm/s	-4.73	cm/s^2	15.5%
Open-loop Translation	575	-2.34	cm/s	-4.06	cm/s^2	-22.1%	-2.63 cm/s	-4.57	cm/s^2	-12.4%



(a) (b)

Fig. 4-14: Results of a closed-loop translation burn on the ISS as seen by the algorithm and the global estimator.



(a) (b)

Fig. 4-15: Results of an open-loop translation burn on the ISS as seen by the algorithm and the global estimator.

Table 4.4: Test results summary of open-loop and closed-loop rotation test burns on the air table.

		Closed-loop Algorithm					Global Metrology Estimate				
	ΔT (ms)	ΔV		a		Error	ΔV		a		Error
Closed-loop Rotation	1594	25.02	deg/s	15.70	deg/s^2	0.1%	24.83	deg/s	15.57	deg/s^2	-0.7%
Open-loop Rotation	752	10.97	deg/s	14.59	deg/s^2	-56.1%	10.96	deg/s	14.57	deg/s^2	-56.2%
Closed-loop Rotation	1563	-25.08	deg/s	-16.05	deg/s^2	0.3%	-24.78	deg/s	-15.85	deg/s^2	-0.9%
Open-loop Rotation	752	-11.01	deg/s	-14.64	deg/s^2	-56.0%	-10.99	deg/s	-14.62	deg/s^2	-56.0%

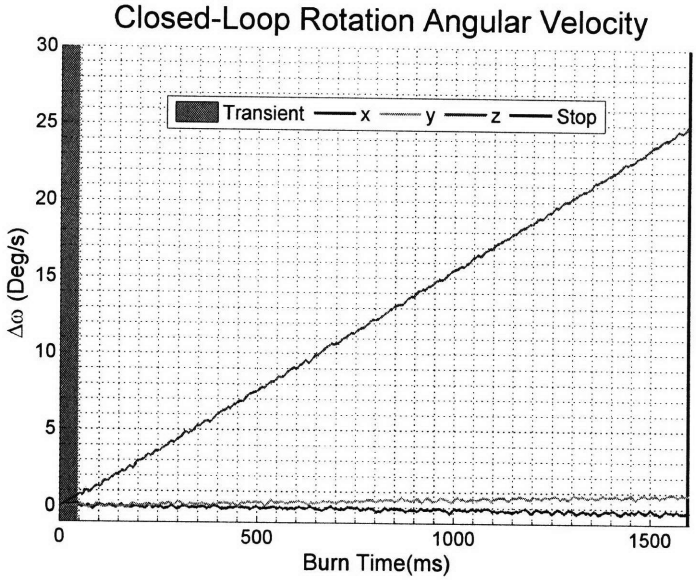


Fig. 4-16: Results of a closed-loop rotation burn on the air table.

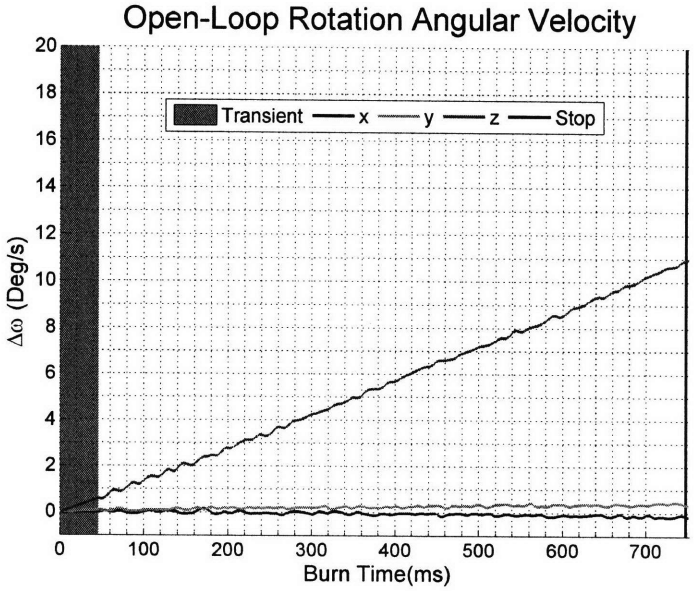


Fig. 4-17: Results of an open-loop rotation burn on the air table.

Table 4.5: Test results summary of open-loop and closed-loop rotation test burns on the ISS.

		Closed-loop Algorithm					Global Metrology Estimate				
	ΔT (ms)	ΔV		a		Error	ΔV		a		Error
Closed-loop Rotation	496	25.02	deg/s	50.45	deg/s^2	0.1%	25.46	deg/s	51.32	deg/s^2	1.8%
Open-loop Rotation	454	—	deg/s	—	deg/s^2	—	22.86	deg/s	50.35	deg/s^2	-8.6%
Closed-loop Rotation	483	-25.08	deg/s	-51.93	deg/s^2	0.3%	-25.54	deg/s	-52.87	deg/s^2	2.1%
Open-loop Rotation	454	-22.54	deg/s	-49.65	deg/s^2	-9.8%	-23.98	deg/s	-52.82	deg/s^2	-4.1%

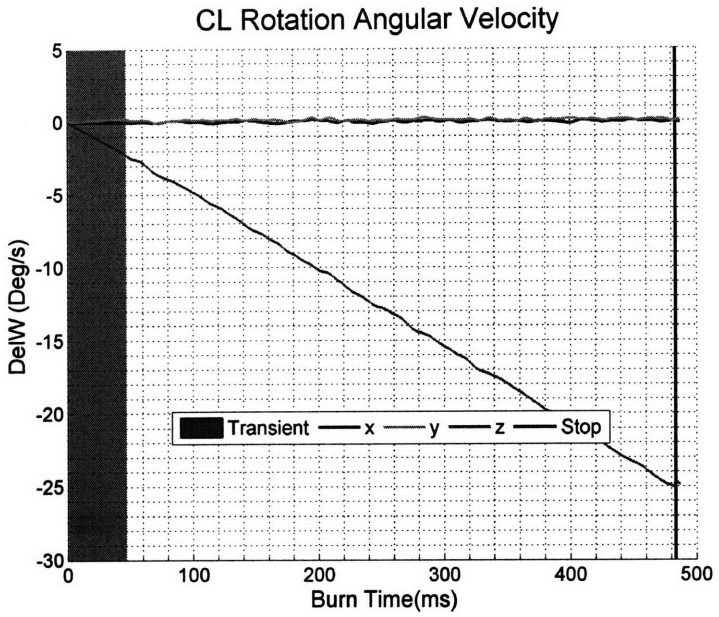


Fig. 4-18: Results of a closed-loop rotation burn on the ISS.

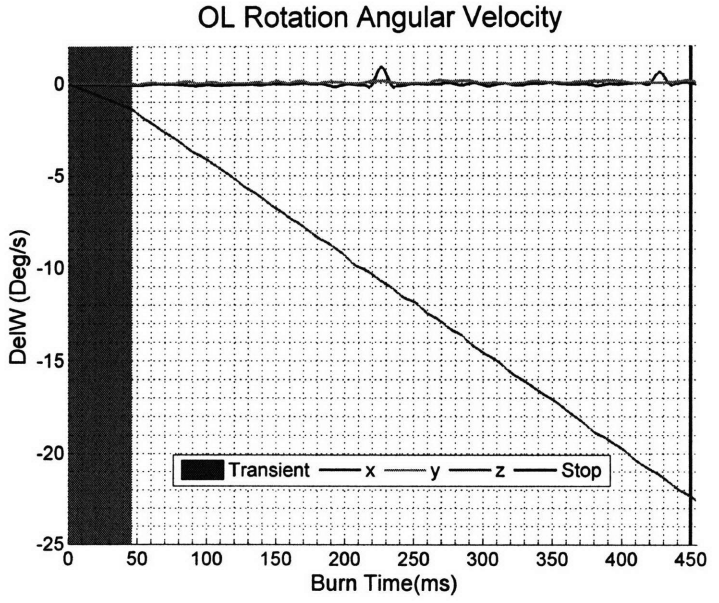


Fig. 4-19: Results of an open-loop rotation burn on the ISS.

Chapter 5

Formation Flying Simulations

To validate the techniques discussed in Chapter 2 and study the impact of ΔV implementation error on realistic formation flying scenarios, a number of extended simulations were conducted. The simulations employed FreeFlyer^{TM}, a commercial, high fidelity, nonlinear orbit propagator [90], interfaced with the MATLAB^{TM} based planner described in Chapter 2. Control inputs for formation reconfiguration and formation-keeping maneuvers were generated with the MATLAB^{TM} planner and then implemented in FreeFlyer^{TM}. In contrast to the planner, which uses GVE-based, linearized dynamics incorporating J_2 effects, the simulation dynamics include disturbances due to Earth's oblateness (J_2 and higher order terms), absolute and differential atmospheric drag, and point mass effects from the sun, moon, and planets. The more complete dynamics model used in the simulations helps expose any deterioration in the plan accuracy that results from the simplifications in the planner dynamics. This chapter details the simulation architecture and presents the results.

5.1 Simulation Architecture Overview

Algorithm 1 gives an overview of the simulation architecture. Each run begins with an initialization procedure that sets the initial conditions of the spacecraft formation in both MATLAB^{TM} and FreeFlyer^{TM}. The number of orbits is decided a priori and assigned to a variable, **orbitsLeft**. Additionally, the flag **needNewPlan** is set to

Algorithm 1: FreeFlyer^{TM} simulation.

spacecraft$_i$: The i^{th} spacecraft
controlInput$_i$: True if control applied at current time step for spacecraft$_i$
planStep : The current step of the plan being implemented
planLength : The number of steps in the plan
needNewPlan: True if a new plan is needed
orbitsLeft : The number of orbits left in the simulation
simResults : Collection of data recorded from the simulation

```
1  begin Initialization
2      forall i do
3          Create spacecraft_i object in FreeFlyer™;
4          Send spacecraft_i initial state from MATLAB™ to FreeFlyer™;
5      needNewPlan = true;
6      Set orbitsLeft;
7  end
8  begin Closed-Loop Control Simulation
9      while orbitsLeft > 0 do
10         forall i do
11             Send spacecraft_i state from FreeFlyer™ to MATLAB™;
12             Record spacecraft_i state;
13         if needNewPlan = true then
14             Update reference orbit to leader's current orbit;
15             Retrieve desired maneuver parameters;
16             Generate plan with MATLAB™ planner;
17             orbitsLeft = orbitsLeft − 1;
18             planStep = 1;
19         forall i do
20             if controlInput_i = true then
21                 Add implementation error from actuator model;
22                 Rotate control from LVLH frame to ECI frame;
23                 Calculate input effect;
24                 Add input effect to spacecraft_i state;
25         if planStep = planLength then needNewPlan = true;
26         else planStep = planStep + 1;
27         forall i do
28             Propagate spacecraft_i in FreeFlyer™;
29 end
30 save simResults;
```

indicate that no current plan exists and one should be created at the first opportunity. The initialization block is executed only once at startup.

The main simulation begins on line 8 and loops as long as there are orbits remaining. Inside the loop, the state of each spacecraft is sent from FreeFlyer™ to a MATLAB™ control script. The script performs some housekeeping on the data and records trajectory histories and other simulation parameters for later review. By keeping track of both the length of the active plan (**planLength**) and the step currently being implemented (**planStep**), the controller knows whether or not a new plan is needed for the fleet. When the last existing step is implemented, the **needNewPlan** flag is set to trigger the planner (see line 25) at the next time step. Any time a new plan is needed, a series of actions occurs. The reference orbit (about which the dynamics are linearized) is reset to the leader's current orbit because the leader may have moved away from the reference orbit during the last maneuver. Then, the controller synchronizes the initial conditions of each spacecraft with their current states in FreeFlyer™, looks up the desired formation configuration, and checks for any additional objectives like fuel balancing, drift minimization, etc. After this information is collected, the planner is invoked to generate an appropriate plan. For these simulations, planning is periodic and performed once per orbit, near apogee. Therefore, every time the planner is called, the **orbitsLeft** variable is simultaneously decremented, and **planStep** is set back to the first step. Each orbit is discretized into 1000 time steps, a number that has previously been shown to work well [29]. The simulation and propagation time steps are adjusted after each planning cycle to maintain this resolution.

Whereas plans are only generated when needed, the next portion of the main simulation loop is executed at every time step. Checks are made for whether any spacecraft has a control input to apply. If so, the command is sent through an actuator model that calculates the actual ΔV applied to the spacecraft. For a perfect actuator, the implementation matches the command; otherwise, the actuator model will slightly distort the control input. A nuance is that the MATLAB™ planner outputs ΔV commands in the LVLH coordinate frame, but FreeFlyer™ computes

velocities in the Earth Centered Inertial (ECI) frame. This requires a transformation on the actual applied ΔV to the ECI frame before the input effect is calculated and added to the spacecraft. The net result of a control input is a change in velocity of the spacecraft, and in these simulations, they are modeled as impulses that occur at the start of each control window.

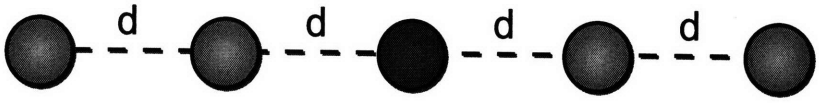
Once all of the control inputs for the present time step (if any) are applied, each spacecraft is propagated in FreeFlyer^{TM}. The simulation loop continues to run until **orbitsLeft** = 0, and then it exits. Before shutting down, all recorded data (**simResults**) is saved to disk.

5.2 Simulation Scenarios

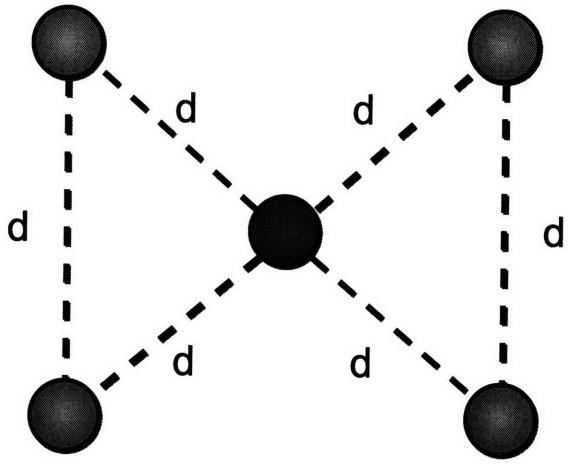
5.2.1 Baseline Scenario

A sequence of maneuvers representative of a formation flying mission was selected as the baseline test scenario and consists of a series of reconfigurations, station keeping, and drift orbits. The formations include an in-track formation and in-track/cross-track passive aperture formations with baselines of 50 m, 500 m, and 5 km. These formations are depicted in Figure 5-1, where "d" denotes the baseline distance. As the name suggests, the in-track formation (Figure 5-1(a)) is separated in the in-track (x) direction. Such a formation allows multiple spacecraft to observe approximately the same area on the Earth's surface within a very short time frame. The passive aperture formations (Figure 5-1(b)) are in the in-track/cross-track (x/y) plane and are of particular interest for interferometry missions because they enable high resolution imaging of the Earth or distant objects in space. During drift orbits, control inputs are forbidden as if science observations were taking place. Five spacecraft comprise the baseline formation, with the leader spacecraft located at the center. The geometry constraints are enforced only at one point in the orbit (apogee), similar to the MMS mission [30].

The baseline scenario was tested for a formation in LEO and one in a highly



(a) The in-track formation



(b) The passive aperture formation

Fig. 5-1: Spacecraft formations for the baseline simulation.

elliptical orbit (HEO). For the LEO case, the orbital parameters (see (2.41) for units) are

$$\mathbf{e}_{LEO} = \begin{bmatrix} 1.0627 & 0.001 & 1.6935 & 0.2393 & 0 & \pi \end{bmatrix}^T \tag{5.1}$$

This translates into a circular orbit at an altitude of 400 km and an inclination of 97.03°.

For the HEO orbit, the orbital parameters are

$$\mathbf{e}_{HEO} = \begin{bmatrix} 9.60735 & 0.875 & 0.3229 & 0 & 0.7854 & \pi \end{bmatrix}^T \tag{5.2}$$

which is an orbit with a semimajor axis of 61277 km, and an inclination of 18.5°. The major difference between the LEO orbit and the HEO orbit is the latter's much higher eccentricity (0.875 compared to 0.001). Both of these orbits have recently been of interest to formation flying missions [91].

The baseline scenario mission sequence cycles through the four different config-

urations (phases) in a systematic fashion. The five spacecraft start in an in-track formation with a 50 m baseline, but small (around 2 m) random errors are added to each initial state to reflect the improbability of the geometry ever being perfect. Maneuvers are performed in the same pattern for each phase of the mission. First, there is a reconfiguration from the old to the new configuration. Four orbits of refining the formation and station keeping follow, and then the formation is allowed to drift for three full orbits while no control is applied. Over this time, the formation deteriorates somewhat; how well the geometry is maintained during this period is one measure of performance. Other elements of performance are also evaluated through this test plan: How accurately can a new configuration be initialized and what is the fuel cost? Once initialized, how expensive is it to maintain a formation, and to what degree of accuracy is the geometry maintained? Finally, any performance degradation in the planner resulting from the use of simplified linear dynamics will be apparent for the wider baseline formations.

5.2.2 Formation-Keeping Scenario

A separate formation-keeping scenario was considered to better understand the fleet's behavior during extended periods spent in the same formation. For this mission, the initial conditions are the same as for the baseline scenario (50 m in-track with small random errors). However, this time the initial reconfiguration is into a 500 m passive aperture, and the fleet remains in that formation for 14 orbits. The total simulation time is 15 orbits, and control is applied during every orbit. Only the LEO orbit was tested.

5.3 ΔV Implementation Error Model

In Chapter 3, the relationship between the actual thrust and desired thrust was modeled as

$$T_a = T_d(1 + \delta) + w$$

At that time, the system under consideration was a single burn, using a single thruster. For the purposes of these simulations, a more general model that applies to the behavior of multiple thrusters over the course of an entire mission is needed. Building on (3.2) yields a new model

$$\Delta\mathbf{V}_{a_{sc}} = (I + \mathbf{A}(t))\Delta\mathbf{V}_{d_{sc}} \tag{5.3}$$

where $\Delta\mathbf{V}_{a_{sc}}$ is the actual $\Delta\mathbf{V}$ implemented, and $\Delta\mathbf{V}_{d_{sc}}$ is the desired ΔV. The second subscript denotes that the velocity command is specified in the frame of the spacecraft. For simplicity, the spacecraft frame is nominally assumed to be aligned with the LVLH reference frame (see below for more details). Both ΔVs are three-element vectors with components nominally in the radial (x), in-track (y) and cross-track (z) directions. Equation (5.3) differs from the previous model through the introduction of the time-varying matrix $\mathbf{A}(t)$ in place of δ and w. The elements of the matrix are

$$\mathbf{A}(t) = \begin{bmatrix} \alpha_x(t) & 0 & 0 \\ 0 & \alpha_y(t) & 0 \\ 0 & 0 & \alpha_z(t) \end{bmatrix} \tag{5.4}$$

The $\alpha_x(t)$, $\alpha_y(t)$ and $\alpha_z(t)$ random variables perform the same function as δ and w in the previous model. Thrusts in the x, y, and z directions are assumed to be performed by different thrusters, hence the need for multiple variables. Recall that δ represented a constant bias in the thrust while w was a random variation about that thrust. In the simulation model, a given $\alpha(t)$ has a mean value $\bar{\alpha}(t)$ and a standard deviation $\sigma_\alpha(t)$. If $\bar{\alpha}(t) \neq 0$ then the control system has a bias. The random variation in performance about that bias (previously w) is captured by $\sigma_\alpha(t)$. In the most general case, the properties of $\alpha(t)$ can vary over time. For example, on the SPHERES satellites discussed in Chapter 4, as fuel is depleted, the thrust decreases. Therefore, $\bar{\alpha}(t)$ could be modeled as a slowly decreasing function.

Throughout this thesis, attitude control has not been considered. The planning process has focused on designing trajectories to move satellites from one position

to another, and the attitude has been left to a separate attitude control system. However, in the real world, attitude error and translational error are coupled. For example, if a satellite tries to perform a burn in the cross-track direction, but its attitude is misaligned, then there will be small components of the thrust in the radial and in-track directions. Suppose that a spacecraft has small attitude errors in its roll, pitch, and yaw. Let these errors be represented by the Euler angles θ, ϕ, and ψ. Successively performing roll, pitch and yaw rotations about the body axes yields a rotation matrix $\mathbf{R}$ from the desired frame to the actual frame [92].

$$\mathbf{R} = \begin{bmatrix} c_\phi c_\psi & -s_\phi c_\theta + c_\phi s_\psi s_\theta & s_\phi s_\theta + c_\phi s_\psi c_\theta \\ s_\phi c_\psi & c_\phi c_\theta + s_\phi s_\psi s_\theta & -c_\phi s_\theta + s_\phi s_\psi c_\theta \\ -s_\psi & c_\psi s_\theta & c_\psi c_\theta \end{bmatrix} \tag{5.5}$$

In (5.5), $\cos x$ and $\sin x$ have been abbreviated as c_x and s_x. Combining (5.3) and (5.5) yields the relationship between a ΔV command and its actual implementation

$$\Delta \mathbf{V}_a = \mathbf{R}(I + \mathbf{A}(t))\Delta \mathbf{V}_{d_{sc}} \tag{5.6}$$

The LHS multiplication by $\mathbf{R}$ rotates the actual velocity into the LVLH frame, so the "sc" subscript is dropped. Although the nominal attitude of the spacecraft was assumed to coincide with the LVLH axes, the same approach could be adapted to arbitrary orientations through the addition of another rotation matrix, relating the desired orientation of the spacecraft to the reference frame.

The scenarios were simulated multiple times with different levels of ΔV implementation error. The first run is with perfect implementation. All planner commands are executed by the spacecraft flawlessly and without attitude error, so this run represents the "best case" operations scenario. The next run includes implementation error that is comparable with a closed-loop accelerometer control system. Assuming the accelerometers are correctly calibrated, any bias in the implementation error is eliminated, but there will still be a slight error for each burn. The last two cases emulate open-loop systems. As these systems lack any real-time monitoring of thruster

Table 5.1: ΔV implementation error parameters for the simulations.

	Ideal	CL Accel	OL Bias Low	OL Bias High
$\bar{\alpha}$	0	0	4%	-4%
σ_α	0	0.1%	2%	2%
σ_θ	0°	0.6°	0.6°	0.6°
σ_ϕ	0°	0.6°	0.6°	0.6°
σ_θ	0°	0.6°	0.6°	0.6°

performance, the variability from burn to burn is higher than it is for the closed-loop system, and there is an additional bias that goes undetected. One simulation used a thruster with a positive bias (tendency to implement more ΔV than desired), and the other used a thruster with a negative bias. The different models are implemented by changing the properties of $\alpha(t)$ and the expected pointing error. These parameters are collected in Table 5.1.

For the simulations just described, no sensor error is added to the system and any performance degradation is due exclusively to poor ΔV implementation. However, to compare the relative importance of good sensing and good plan implementation, additional simulations were run, with perfect implementation and varying levels of sensor error. The sensor accuracies were taken from CDGPS navigation filters developed by three different researchers: Ebinuma [93], Busse [46], and Leung [47]. Since the performance gap between the Leung and Busse filter is quite large, a hypothetical filter (with a "medium" magnitude of error) was added to the mix. The properties of each of the filters are summarized in Table 5.2. In any real system, implementation *and* sensor error would be present, so each simulation was run twice more with different levels of implementation error and medium sensor error. These special cases are discussed last.

Table 5.2: CDGPS navigation filter RMS accuracies.

Name	Year	Position (cm)	Velocity (mm/s)
Ebinuma [93]	2001	5	1
Busse [46]	2002	1	0.5
Medium	N/A	0.5	0.05
Leung [47]	2005	0.15	0.005

5.4 Simulation Results

5.4.1 Baseline Scenario

The formation-flying simulations demonstrate a definite performance advantage to accurately implementing ΔV commands. This advantage creates fuel savings during reconfigurations and allows new formations to be formed more rapidly. For the LEO simulations with implementation error, the fuel use and position error (average per spacecraft) for each phase of the mission is computed and compiled in Table 5.3. To facilitate comparisons between the control systems, these statistics are further subdivided into the initial reconfiguration maneuver (one orbit), formation refinement (four orbits), and the drift portion (three orbits). The refinement division exists because the initial reconfiguration maneuver can leave residual error that requires additional, smaller maneuvers to fix. Fuel spent during the refinement portion increases the overall ΔV cost for a reconfiguration. The reconfigurations themselves are by far the most expensive maneuvers when measured in terms of fuel expenditure. This is expected, as the spacecraft are rearranging themselves, and sometimes traveling a significant distance in a relatively short amount of time (one orbit takes ~ 92.5 minutes). The planned fuel use for these reconfiguration maneuvers is largely unaffected by the control system used; the dominant factor is the change in formation geometry itself.

Figure 5-2 graphs the average position error per spacecraft for the formation throughout each baseline simulation with implementation error. Figure 5-2(a) is the complete mission and allows quick comparisons between the different mission phases, while Figures 5-2(b)–5-2(e) provide detailed views of each phase separately. The dashed horizontal lines mark 1% of the formation baseline distance. Similarly, Figure 5-4 depicts the fuel usage per orbit for each control system. The first data point of each phase corresponds to the reconfiguration maneuver, meaning higher fuel use and higher position error. The next four data points are for the formation refinement portion. During this phase, the position error is steadily decreased as the controller applies corrections to fix any defects remaining after the initial reconfiguration. Like-

wise, fuel use also decreases as the magnitude of these corrections lessens. Finally, the last three data points are for the drift orbits. These portions are characterized by zero fuel use (they are omitted in Figure 5-4) and increasing position error, as the controller is not permitted to correct J_2 effects and other disturbances that act on the formation.

By the end of each phase and immediately prior to the next reconfiguration maneuver, the position accuracy of the formation is approximately the same for all four test cases. Therefore, it is not surprising that the planned fuel uses for the reconfigurations are very similar, regardless of implementation error. On the other hand, the benefit of more accurate ΔV implementation is quite apparent from the position error measurements immediately following a reconfiguration. Compared to the ideal case, when attitude error is added and closed-loop control is used, these position errors are an order of magnitude higher. The ideal reconfiguration from the 50 m in-track to 50 m passive aperture formation (Figure 5-2(b)) leads to an average position error of 5.2 cm per spacecraft. For the closed-loop accelerometer system, the error increases by a factor of 20 to 1.08 m. The performance is worse for the open-loop systems, with 1.61 m for the system with low thrust and 2.91 m for the system with high thrust.

In Figure 5-2, as the formation baseline increases over the first three phases, so too does the average error. Ideal implementation of the 5 km passive aperture reconfiguration (Figure 5-2(d)) results in an error of 2.66 m. The baseline is 100 times that of the 50 m aperture, and the error has increased by about the same ratio. In both cases, inaccuracies are limited to well under 1% of the baseline, indicating excellent overall performance for the planner. When the accelerometer system is used, the error is 11.80 m, or about 0.2% error, still very good. The open-loop systems perform much worse, with 179.94 m and 135.71 m error for the low bias and high bias, respectively. This makes a convincing case for the benefit of accurate ΔV implementation during reconfiguration maneuvers.

Moving on to the formation refinement portion of the mission reveals time and fuel savings for the more accurate control systems because they spend less of both cleaning up errors left over from reconfiguration. In Figure 5-2, although each control

system eventually reaches approximately the same level of accuracy, it takes several orbits longer for those with more implementation error. Meanwhile, the spacecraft burn more fuel, as seen in Figure 5-4. In fact, throughout the formation refinement phase, the open-loop systems consume more than three to five times the fuel of the closed-loop accelerometer system. Missions that regularly switch configurations could conceivably lose significant observation time and at the same time use more fuel by not following the plan accurately. However, by the end of the actively controlled portion of each phase, the formation is controlled to within a tolerance of 0.02% of the baseline, regardless of the control system — sub-centimeter accuracy for the 50 m formation, and sub-meter accuracy for the 5 km passive aperture. Given enough time then, the final position accuracy of a formation is not affected by implementation error, just the time and fuel it takes to get there.

All four test cases offered competitive performance during the drift phase. By the start of the sixth orbit, when drifting began, the formations were already well established; the larger formation errors introduced during reconfiguration by inaccurate ΔV implementation had been rectified. Because the planner favors orbits with low drift, the formations remain fairly stable for all test cases.

Sensor noise affects the formation performance differently than implementation error, as seen in Table 5.4 and Figures 5-3 and 5-5. As the sensor noise increases, the minimum achievable position error also increases. This contrasts with the behavior for poor ΔV implementation, where the fleet eventually reached the same level of position accuracy as the ideal case, even though it took longer. There is a wide gap in performance between the best estimator (Leung) and the worst (Ebinuma). The Leung estimator offers performance practically on the level of the ideal case, but at the other end of the spectrum, the Busse and Ebinuma filters result in final errors of several meters. Unlike the impact of implementation error, the impact of sensor error does not change very much for different baseline lengths. For example, the Busse filter yields about 2 m position error regardless of whether the formation is a 50 m, 500 m or 5 km baseline. This indicates that estimation error of just 1 cm position accuracy and 0.5 mm/s causes errors of much greater magnitude, consistent with findings in [39],

but independent of formation geometry. As seen in 5-3, it is not possible to control formations with short baselines to less than 1% of the baseline when using the less accurate estimators. The fuel use also increases with sensor error, as the formation is constantly trying to correct measured errors (whether real or imagined). Still, the key point is that the system performs near the ideal level when the Leung estimator is used, suggesting that implementation error will dominate.

The HEO simulation results tell much the same story; in terms of position accuracy and fuel use, the numbers in Table 5.5 (implementation error) and Table 5.6 (sensor error) support the conclusions reached above. Detailed results for these simulations are plotted in Figures 5-6 and 5-8 for the implementation error cases, and Figures 5-7 and 5-9 for the sensor error cases. Overall, the HEO formation uses much less fuel to complete the mission, but most of this can be traced to the much longer orbital period of the HEO orbit; clocking in at just under 42 hours, it is about 27 times as long as the LEO orbit. Interestingly, the ratio of fuel use between the HEO and LEO reconfiguration maneuvers ranges from 27 to 30, a nearly one-to-one ratio. The HEO formation is able to implement smaller ΔV commands and let the orbital dynamics do more of the work over a longer time period. Reconfiguration from an in-track to a passive aperture formation requires a little more effort than transitioning from one passive aperture to another, and the ratio here compared to LEO is closer to 30. The price of linearizing the dynamics is greater for the HEO orbit, and overall, the position error ranges from about five to fifteen times that of the equivalent LEO maneuvers. Fortunately, this is still good enough to provide formation-keeping accuracy of about 0.1%.

5.4.2 Formation-Keeping Scenario

The data for the formation-keeping simulations with ΔV implementation error is collected in Table 5.7, and the simulations with sensor error are in Table 5.8. Once again, the first orbit is considered the reconfiguration, and the next four orbits are formation refinement. The last ten orbits are different than before, and for this scenario they consist of active formation-keeping. The baseline simulation has already

provided plenty of information on the reconfiguration and refinement stages, so the extended formation-keeping stage is the most interesting. As seen in Figures 5-10(a) and 5-11(a), the fuel use and position accuracy for formation-keeping is unaffected by the implementation error; the only differences are in the reconfiguration and refinement stages. Once the formation is well established, the fuel required to maintain it is very small (fractions of a mm/s per orbit). With such small ΔV being applied, it's difficult for an actuator error of 2% or 3% to have much of an effect. Sensor error is more costly during formation-keeping. Position error magnitudes are on par with those in the baseline simulation, but more fuel is required for the less accurate estimators. The Ebinuma estimator uses about 8 times as much fuel as the ideal case (Figure 5-11(b)), even though this still amounts to fractions of mm/s. The Leung estimator closely follows the ideal case again.

5.4.3 Simultaneous Implementation and Sensor Error

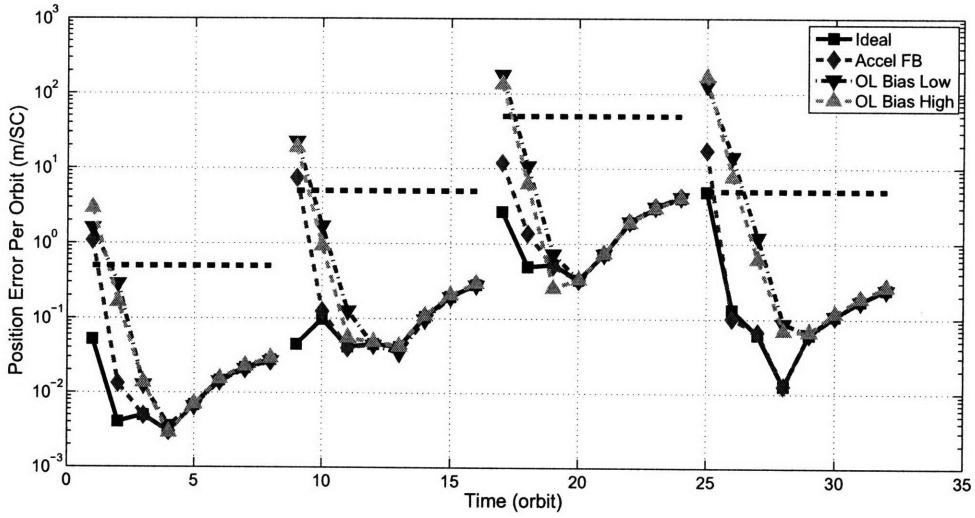
Figure 5-12 plots the per orbit position error and fuel use for two new baseline simulation cases: open-loop biased high ΔV implementation with medium sensor error, and accelerometer feedback ΔV implementation with medium sensor error. For easy comparison, some of the previous results are included again: the ideal case, the biased high ΔV implementation with perfect sensing, and perfect implementation with medium sensor error. As noted previously, poor ΔV implementation dominates the formation position error immediately after a reconfiguration maneuver. The error is initially highest in 5-12(a) for the three cases with imperfect ΔV implementation, regardless of whether there is additional sensor error. As before, the best attainable accuracy is limited by the performance of the sensor, and at the conclusion of the refinement portion of each phase, the two cases with no sensor error are the most accurate, regardless of implementation error. However, upgrading from an open-loop ΔV implementation to accelerometer feedback allows the formation to reach the sensor limit a full orbit earlier. The fuel usage in 5-12(b) follows the same pattern; at first the formations with implementation error use more fuel, but by the end of each phase the fuel use depends on sensor accuracy. Switching from open-loop ΔV imple-

mentation to accelerometer feedback shaves an orbit off of the time to reach steady state fuel consumption. Improving the sensor would further increase the advantage of accelerometer feedback over open-loop ΔV implementation.

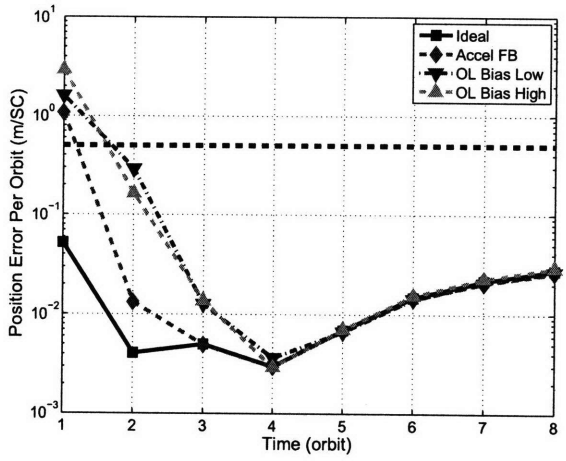
Figure 5-13 displays the same information for the formation-keeping simulation, and the findings are consistent with the baseline simulation. Since there is only one reconfiguration and it occurs at the very start of the formation-keeping simulation, the dominance of the sensor error on long-term performance is a bit clearer. Equally visible is the importance of accurate ΔV implementation during the initial reconfiguration.

5.5 Chapter Summary

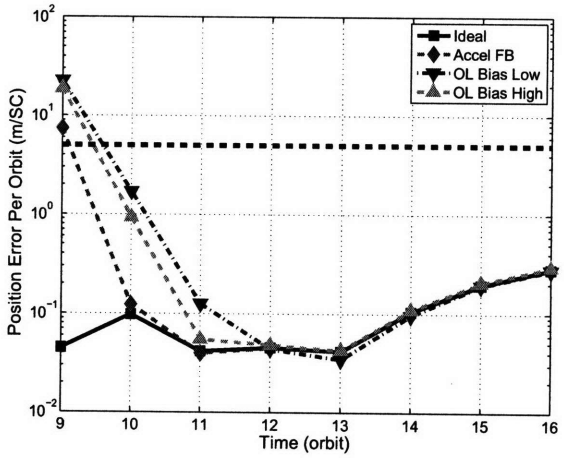
A series of high-fidelity simulations has verified the effectiveness of the planner developed in Chapter 2; formations of spacecraft in a circular LEO orbit and also a highly elliptical orbit were successfully controlled during lengthy simulations. These simulations were used to explore the importance of accurately implementing the ΔV commands that make up the maneuver plans, and control systems of different qualities were tested and compared with the aid of a thruster model. Separate simulations were conducted for an ideal control system, a system modeled after the closed-loop accelerometer feedback approach of Chapter 3, and open-loop systems with varying levels of bias. Study of the average fuel consumption and accuracy attained by each of the systems shows noticeable improvements during reconfiguration maneuvers for more accurate ΔV implementation. Fuel use and position error were compared with those caused by different levels of sensor error, and for the latest CDGPS navigation filters, the ΔV implementation error has a larger effect. In general, sensor error limits the attainable formation accuracy, but implementation error determines how long it takes to reach that accuracy. This observation was confirmed with simulations including both types of error simultaneously. Overall, provided the sensing is very good (consistent with the latest technology), accurate ΔV implementation of plans enables both greater precision in formation flying and more efficient fuel usage.



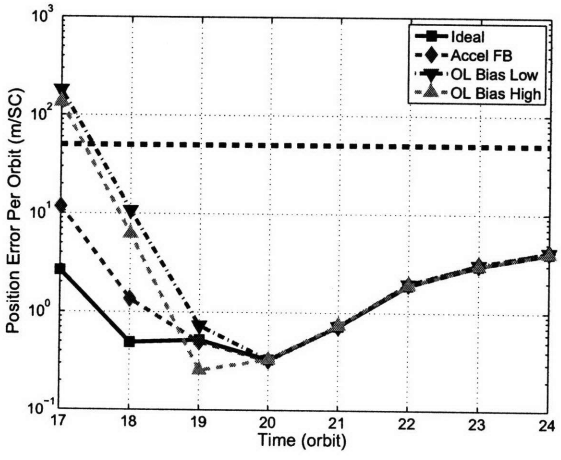
(a) Complete simulation



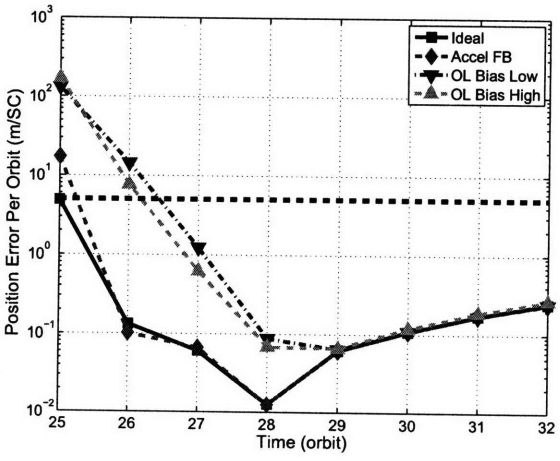
(b) 50 m passive aperture



(c) 500 m passive aperture

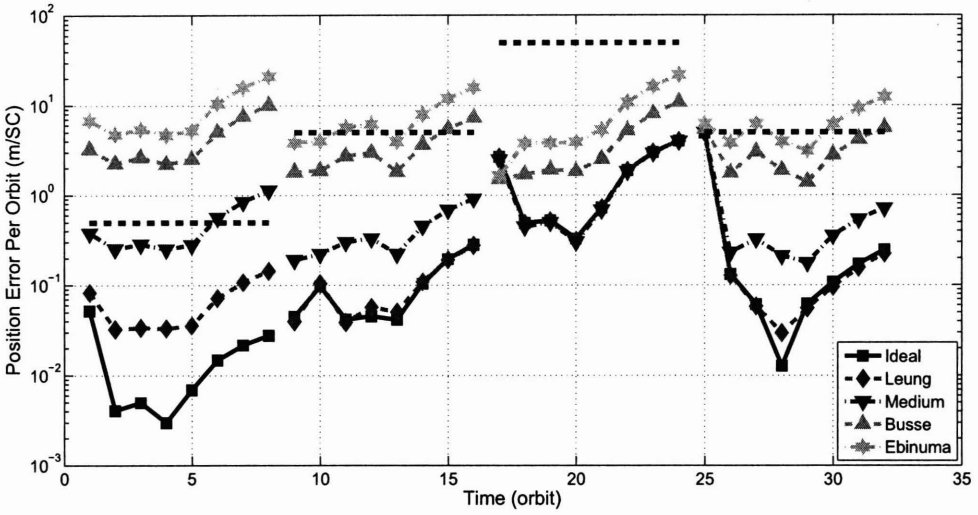


(d) 5 km passive aperture

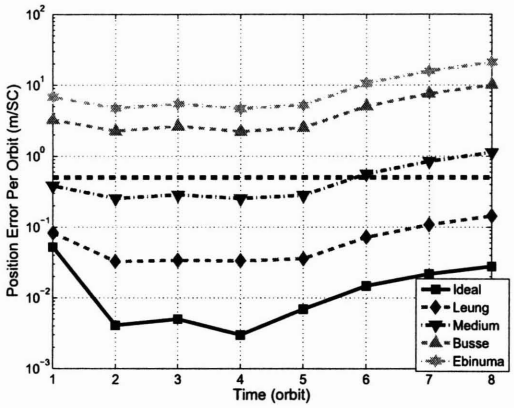


(e) 500 m in-track

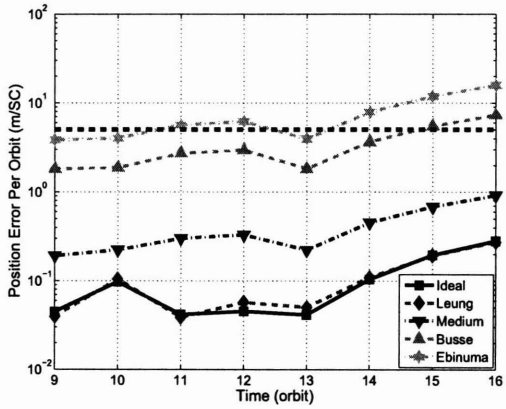
Fig. 5-2: Position error per orbit for the formations in the LEO baseline simulation with implementation error.



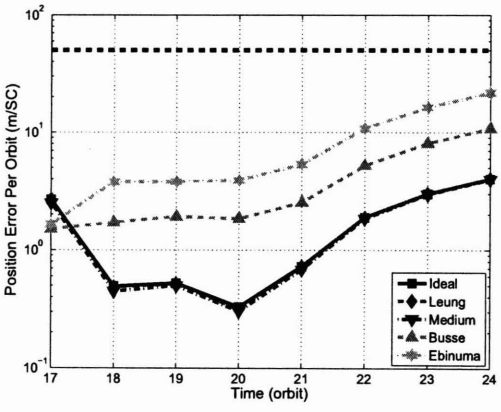
(a) Complete simulation



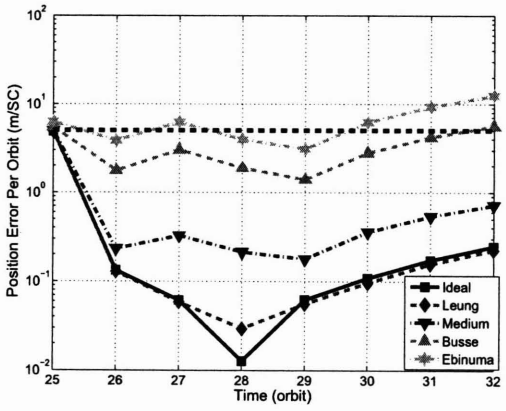
(b) 50 m passive aperture



(c) 500 m passive aperture

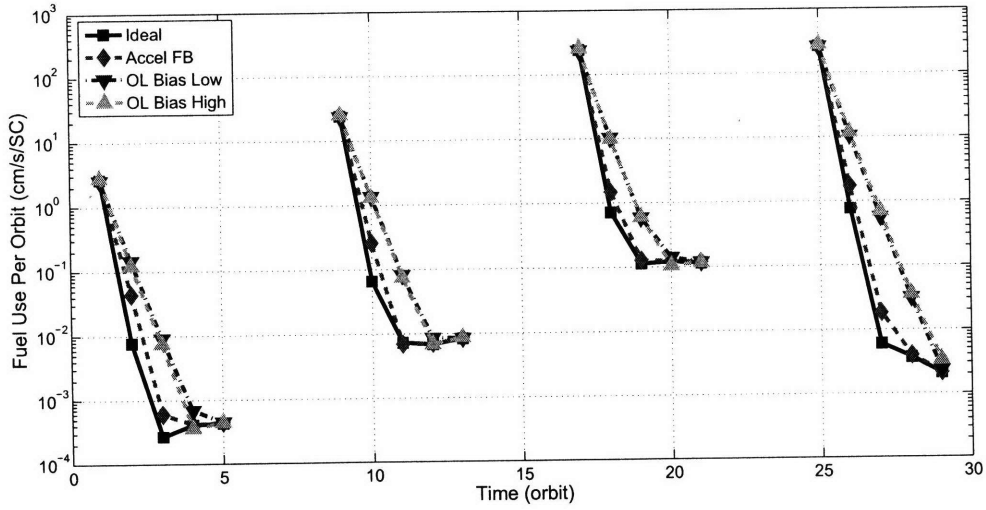


(d) 5 km passive aperture

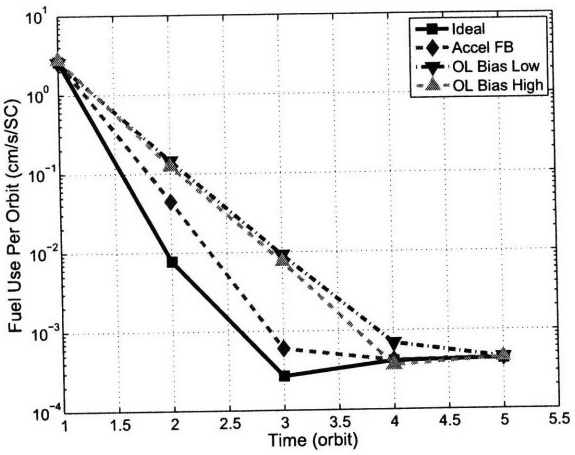


(e) 500 m in-track

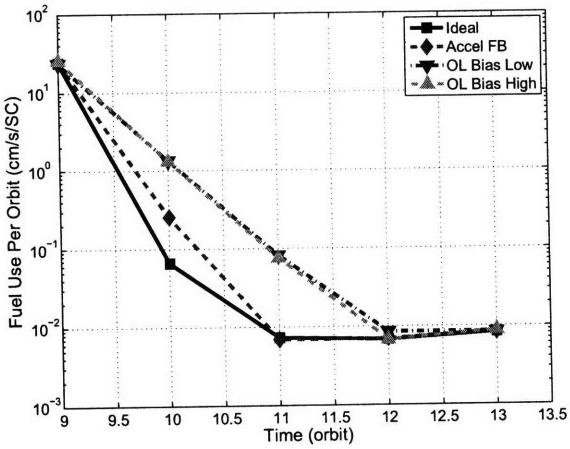
Fig. 5-3: Position error per orbit for the formations in the LEO baseline simulation with sensor error.



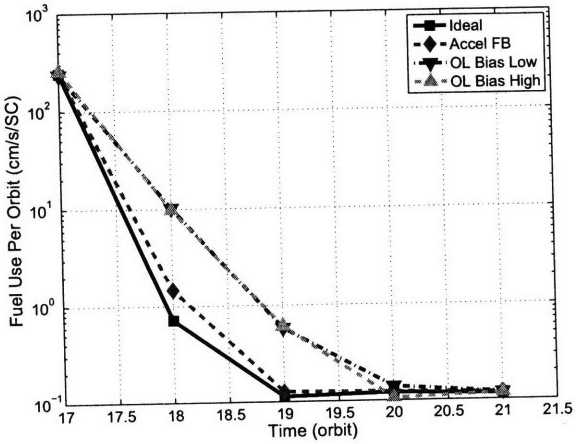
(a) Complete simulation



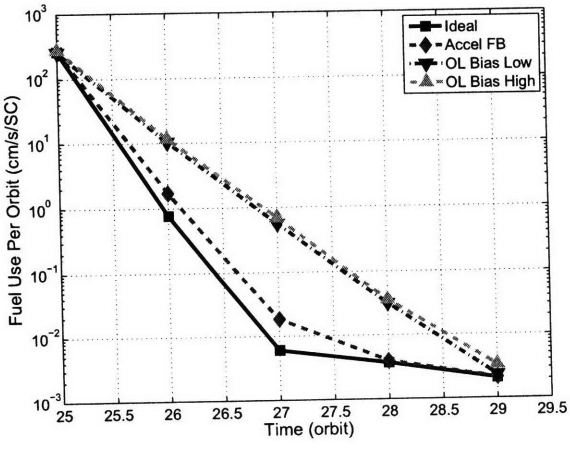
(b) 50 m passive aperture



(c) 500 m passive aperture

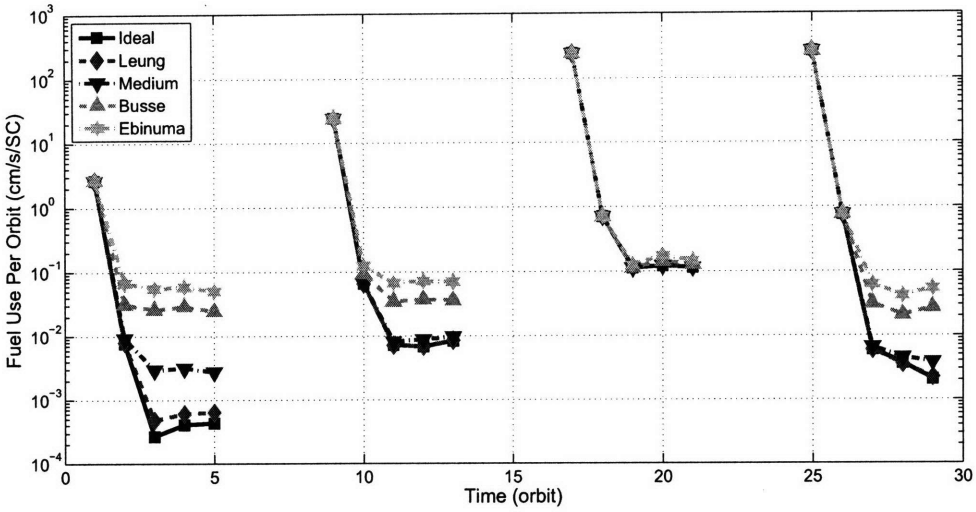


(d) 5 km passive aperture

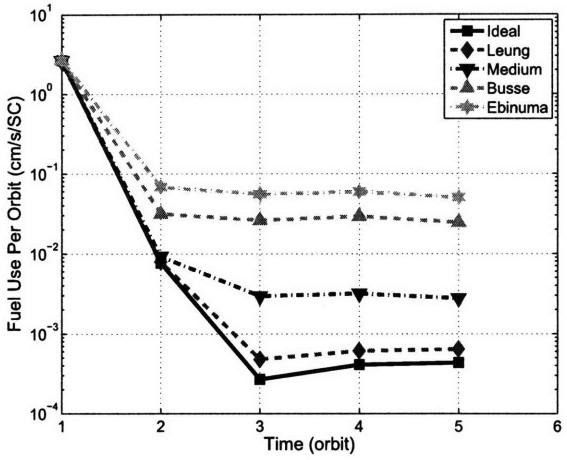


(e) 500 m in-track

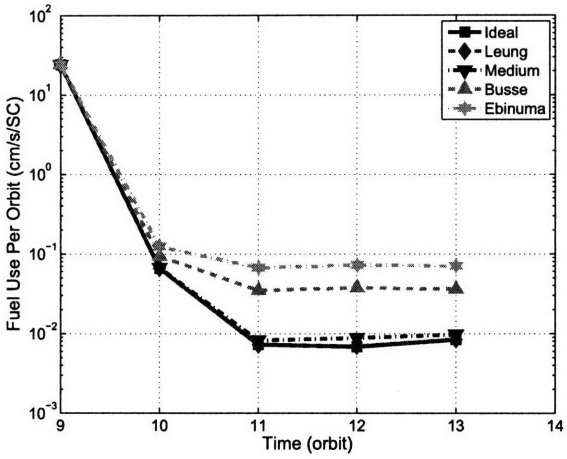
Fig. 5-4: Fuel use per orbit for the formations in the LEO simulations with implementation error.



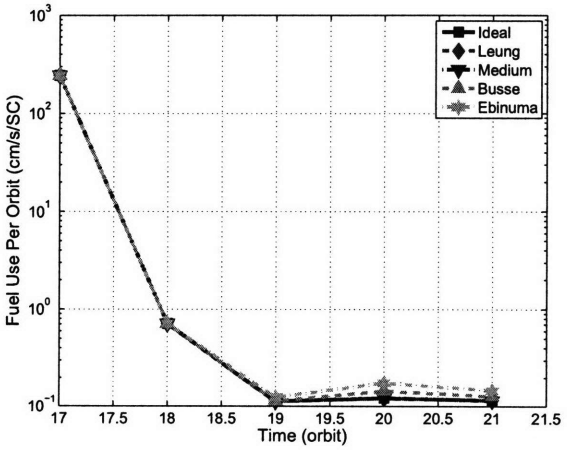
(a) Complete simulation



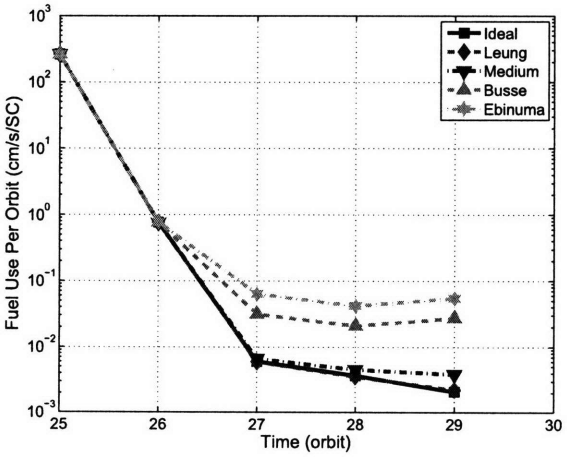
(b) 50 m passive aperture



(c) 500 m passive aperture



(d) 5 km passive aperture



(e) 500 m in-track

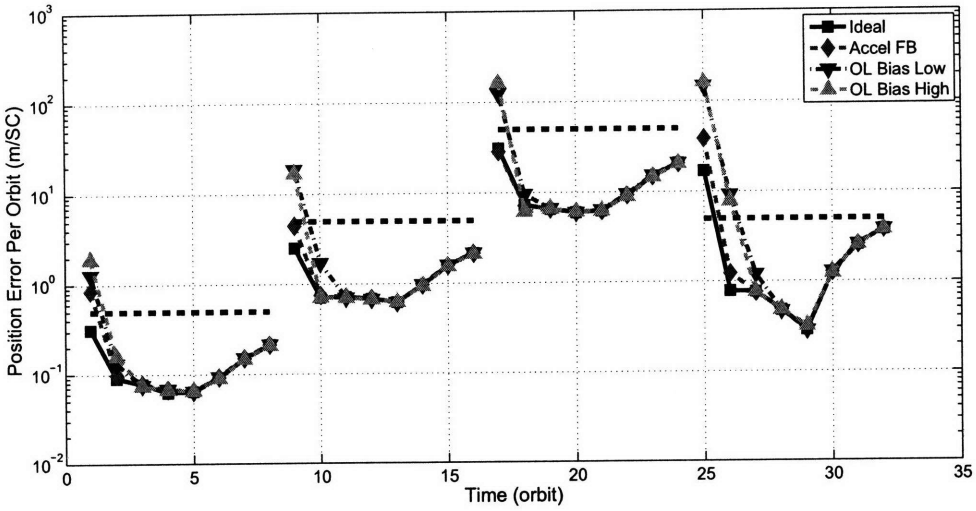
Fig. 5-5: Fuel use per orbit for the formations in the LEO simulations with sensor error.

Table 5.3: Baseline simulation results for the LEO orbit with implementation error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

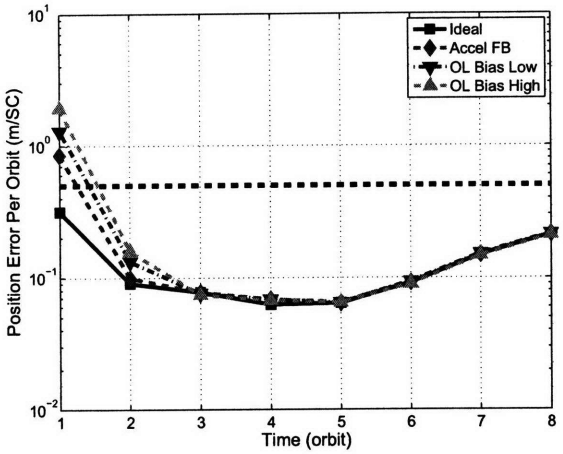
		Ideal	Accel FB	OL Bias Low	OL Bias High
50IT-50PA	Fuel Used	2.652	2.686	2.563	2.778
	Fuel Planned	2.652	2.652	2.652	2.652
	Position Error	0.052	1.080	1.608	2.915
50PA Refine	Fuel Used	0.002	0.011	0.039	0.033
	Fuel Planned	0.002	0.011	0.040	0.031
	Position Error	0.005	0.007	0.078	0.047
50PA Drift	Position Error	0.021	0.020	0.021	0.023
50PA-500PA	Fuel Used	24.166	24.486	23.533	25.554
	Fuel Planned	24.166	24.166	24.166	24.166
	Position Error	0.045	7.417	22.597	18.768
500PA Refine	Fuel Used	0.022	0.069	0.361	0.353
	Fuel Planned	0.022	0.068	0.370	0.333
	Position Error	0.057	0.062	0.475	0.270
500PA Drift	Position Error	0.194	0.198	0.189	0.203
500PA-5kPA	Fuel Used	242.710	246.847	236.596	256.083
	Fuel Planned	242.710	242.710	242.710	242.710
	Position Error	2.659	11.795	179.940	135.709
5kPA Refine	Fuel Used	0.265	0.461	2.771	2.660
	Fuel Planned	0.265	0.451	2.842	2.507
	Position Error	0.517	0.724	3.135	1.888
5kPA Drift	Position Error	2.968	3.078	3.065	2.974
5kPA-500IT	Fuel Used	262.630	266.802	256.983	277.685
	Fuel Planned	262.630	262.635	262.628	262.633
	Position Error	4.894	17.258	131.303	168.501
500IT Refine	Fuel Used	0.194	0.430	2.745	3.112
	Fuel Planned	0.194	0.422	2.789	2.926
	Position Error	0.067	0.061	3.951	2.106
500IT Drift	Position Error	0.176	0.175	0.174	0.192

Table 5.4: Baseline simulation results for the LEO orbit with sensor error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

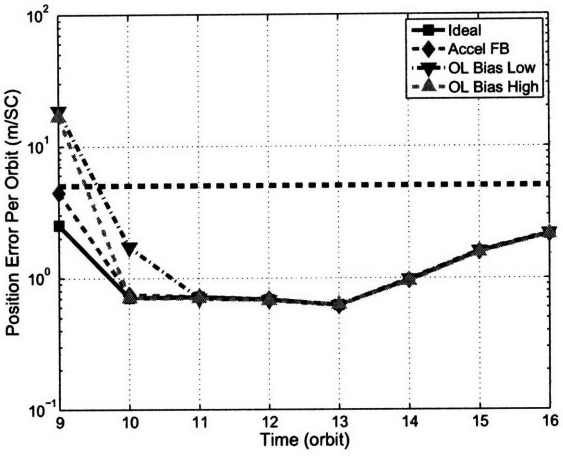
		Ideal	Leung	Medium	Busse	Ebinuma
50IT-50PA	Fuel Used	2.652	2.653	2.654	2.666	2.681
	Fuel Planned	2.652	2.653	2.654	2.666	2.681
	Position Error	0.052	0.083	0.382	3.242	6.819
50PA Refine	Fuel Used	0.002	0.002	0.005	0.028	0.058
	Fuel Planned	0.002	0.002	0.005	0.028	0.058
	Position Error	0.005	0.034	0.269	2.408	5.024
50PA Drift	Position Error	0.021	0.108	0.847	7.546	15.782
50PA-500PA	Fuel Used	24.166	24.166	24.167	24.174	24.182
	Fuel Planned	24.166	24.166	24.167	24.174	24.182
	Position Error	0.045	0.040	0.191	1.818	3.851
500PA Refine	Fuel Used	0.022	0.022	0.023	0.050	0.083
	Fuel Planned	0.022	0.022	0.023	0.050	0.083
	Position Error	0.057	0.063	0.269	2.348	4.951
500PA Drift	Position Error	0.194	0.192	0.684	5.488	11.871
500PA-5kPA	Fuel Used	242.710	242.710	242.705	242.665	242.617
	Fuel Planned	242.710	242.710	242.705	242.665	242.617
	Position Error	2.659	2.636	2.496	1.502	1.641
5kPA Refine	Fuel Used	0.265	0.265	0.265	0.274	0.291
	Fuel Planned	0.265	0.265	0.265	0.274	0.291
	Position Error	0.517	0.509	0.482	1.999	4.209
5kPA Drift	Position Error	2.968	2.958	2.925	8.016	16.301
5kPA-500IT	Fuel Used	262.630	262.630	262.628	262.603	262.577
	Fuel Planned	262.630	262.630	262.628	262.603	262.577
	Position Error	4.894	4.909	4.978	5.470	6.250
500IT Refine	Fuel Used	0.194	0.194	0.195	0.212	0.234
	Fuel Planned	0.194	0.194	0.195	0.212	0.234
	Position Error	0.067	0.068	0.237	2.016	4.281
500IT Drift	Position Error	0.176	0.158	0.536	4.195	9.298



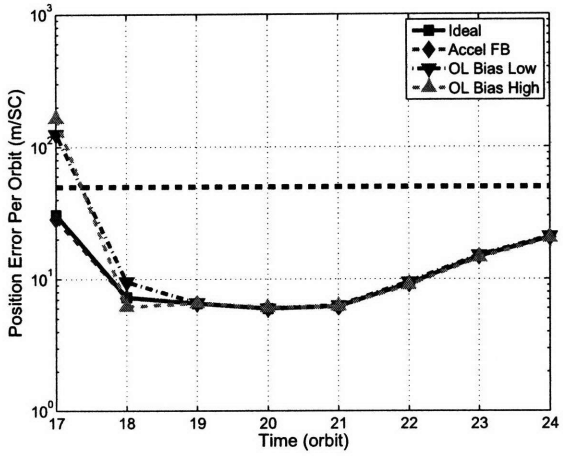
(a) Complete simulation



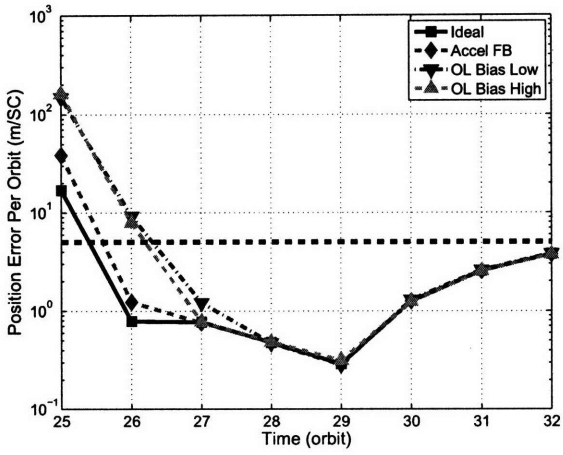
(b) 50 m passive aperture



(c) 500 m passive aperture

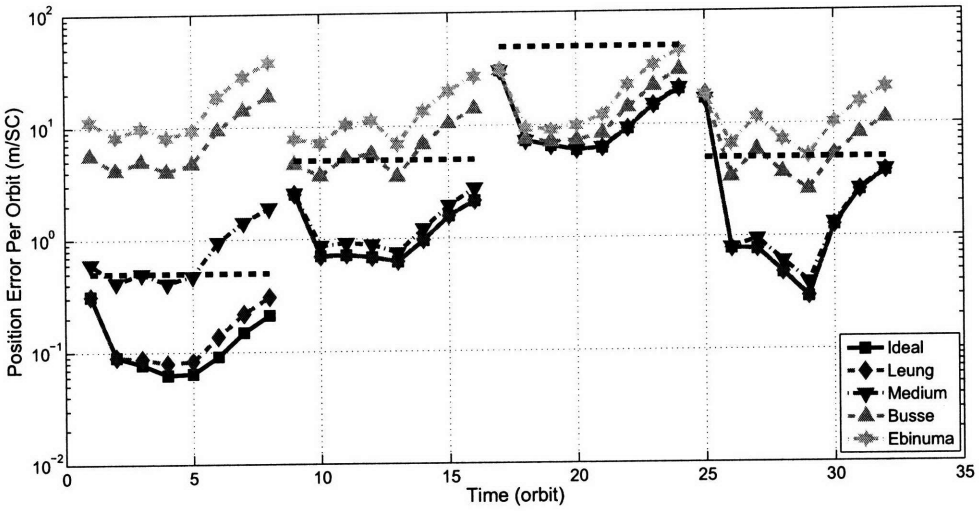


(d) 5 km passive aperture

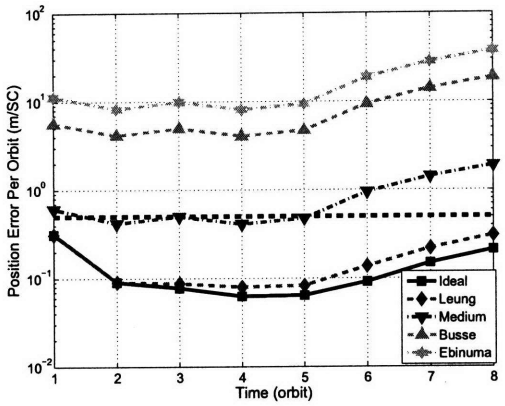


(e) 500 m in-track

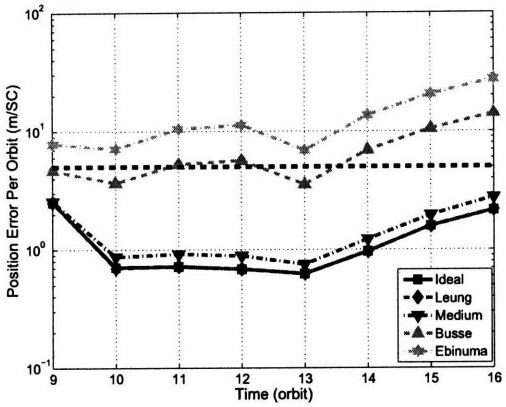
Fig. 5-6: Position error per orbit for the formations in the HEO simulations with implementation error.



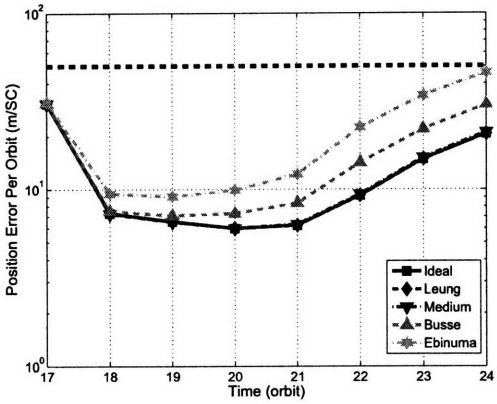
(a) Complete simulation



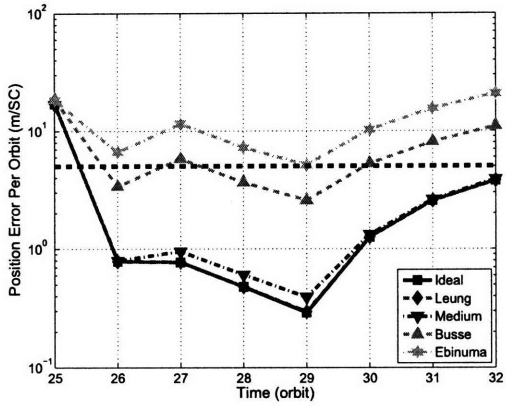
(b) 50 m passive aperture



(c) 500 m passive aperture

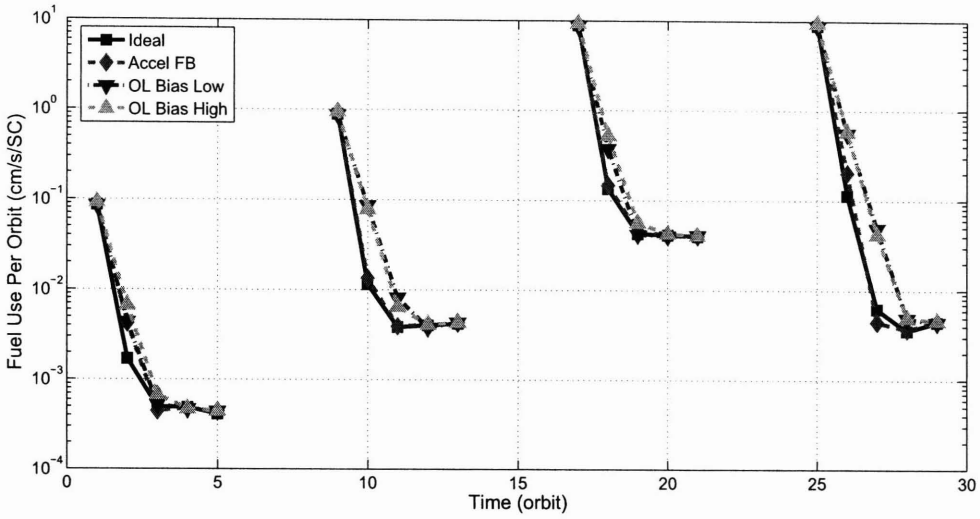


(d) 5 km passive aperture

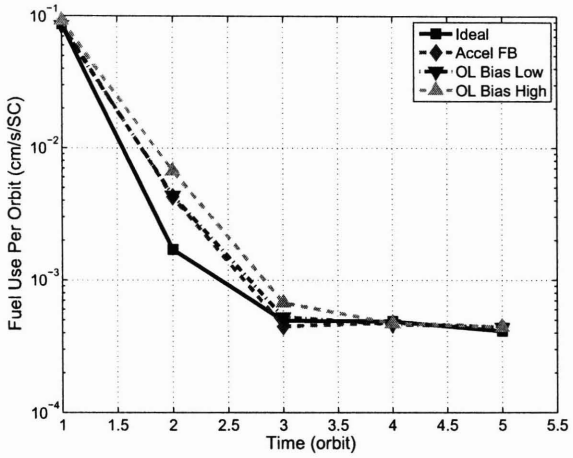


(e) 500 m in-track

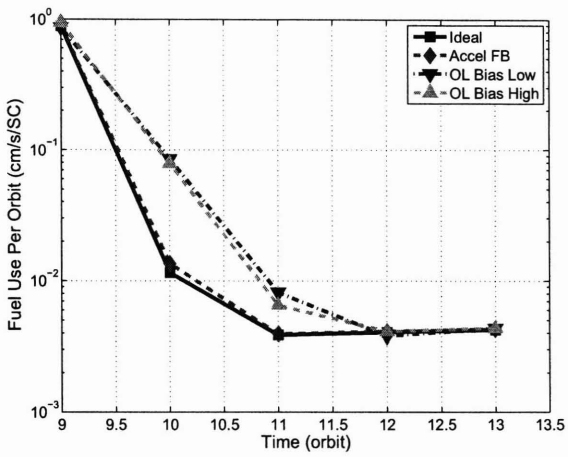
Fig. 5-7: Position error per orbit for the formations in the HEO simulations with sensor error.



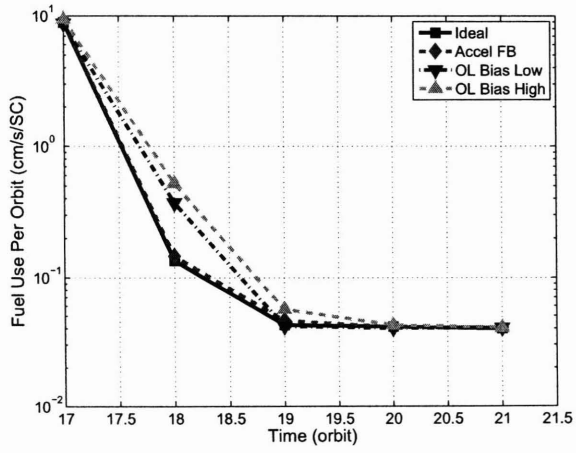
(a) Complete simulation



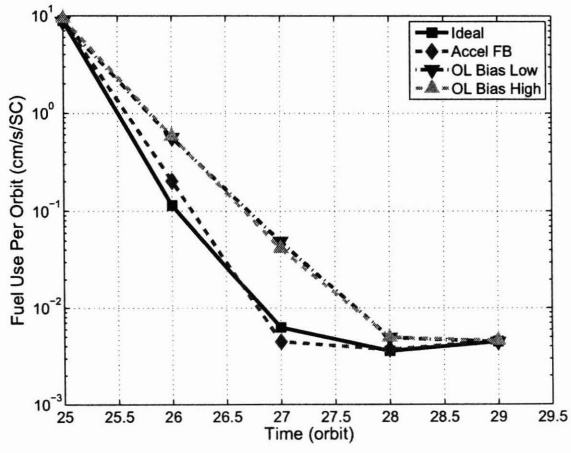
(b) 50 m passive aperture



(c) 500 m passive aperture

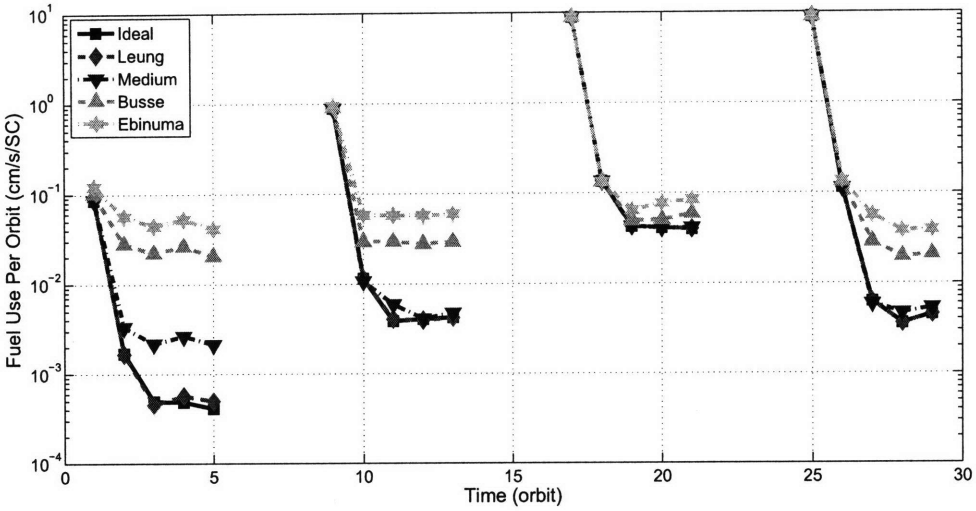


(d) 5 km passive aperture

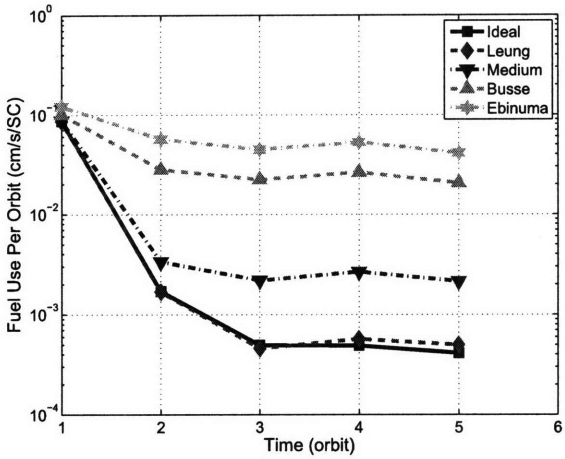


(e) 500 m in-track

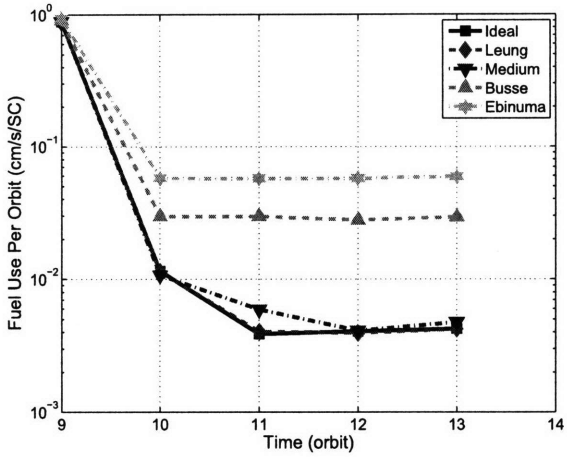
Fig. 5-8: Fuel use per orbit for the formations in the HEO simulations with implementation error.



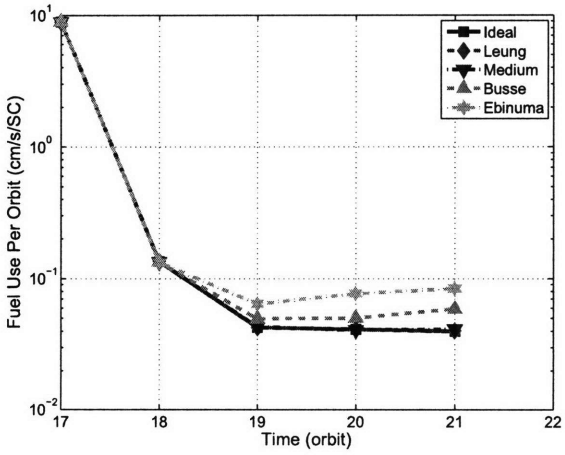
(a) Complete simulation



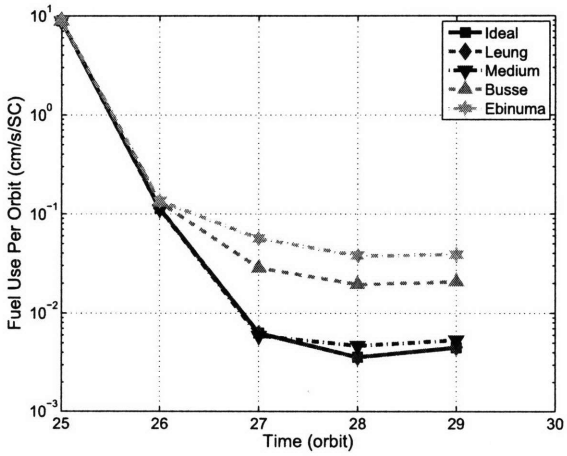
(b) 50 m passive aperture



(c) 500 m passive aperture



(d) 5 km passive aperture



(e) 500 m in-track

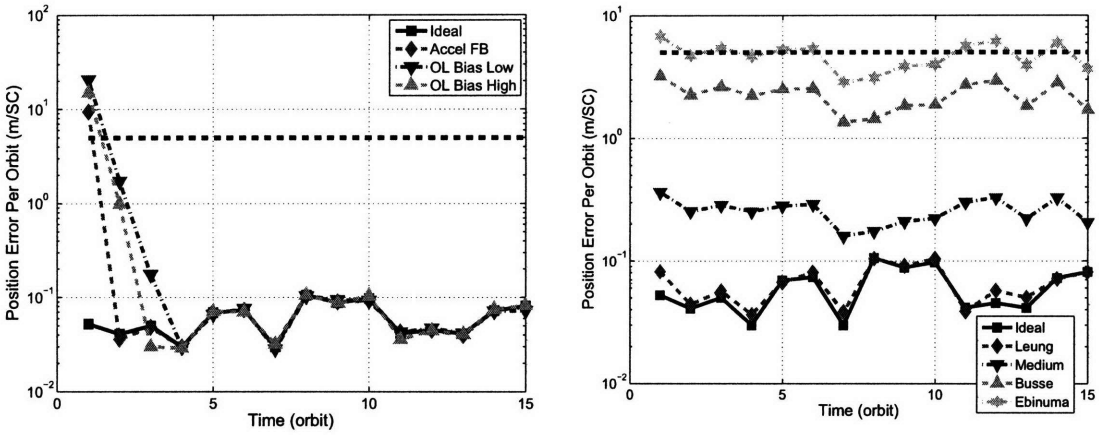
Fig. 5-9: Fuel use per orbit for the formations in the HEO simulations with sensor error.

Table 5.5: Baseline simulation results for the HEO orbit with implementation error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

		Ideal	Accel FB	OL Bias Low	OL Bias High
50IT-50PA	Fuel Used	0.087	0.088	0.085	0.092
	Fuel Planned	0.087	0.087	0.087	0.087
	Position Error	0.316	0.848	1.292	1.920
50PA Refine	Fuel Used	0.001	0.001	0.001	0.002
	Fuel Planned	0.001	0.001	0.002	0.002
	Position Error	0.074	0.078	0.085	0.091
50PA Drift	Position Error	0.151	0.152	0.153	0.149
50PA-500PA	Fuel Used	0.888	0.905	0.870	0.942
	Fuel Planned	0.888	0.888	0.888	0.888
	Position Error	2.516	4.445	18.605	16.833
500PA Refine	Fuel Used	0.006	0.007	0.026	0.023
	Fuel Planned	0.006	0.006	0.026	0.022
	Position Error	0.684	0.695	0.930	0.679
500PA Drift	Position Error	1.568	1.566	1.586	1.550
500PA-5kPA	Fuel Used	8.899	9.030	8.723	9.427
	Fuel Planned	8.899	8.899	8.900	8.899
	Position Error	30.691	28.465	125.422	165.689
5kPA Refine	Fuel Used	0.065	0.068	0.124	0.165
	Fuel Planned	0.065	0.067	0.126	0.157
	Position Error	6.516	6.497	7.125	6.272
5kPA Drift	Position Error	14.917	14.843	15.216	14.626
5kPA-500IT	Fuel Used	8.924	9.073	8.732	9.467
	Fuel Planned	8.924	8.923	8.926	8.923
	Position Error	16.950	38.509	149.296	162.362
500IT Refine	Fuel Used	0.032	0.054	0.154	0.158
	Fuel Planned	0.032	0.053	0.161	0.150
	Position Error	0.581	0.689	2.787	2.359
500IT Drift	Position Error	2.528	2.530	2.566	2.512

Table 5.6: Baseline simulation results for the HEO orbit with sensor error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

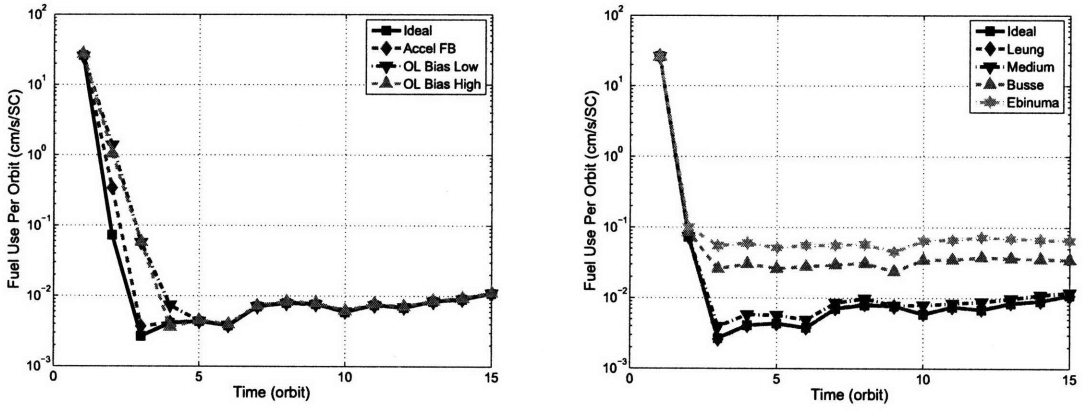
		Ideal	Leung	Medium	Busse	Ebinuma
50IT-50PA	Fuel Used	0.087	0.087	0.087	0.101	0.121
	Fuel Planned	0.087	0.087	0.087	0.101	0.121
	Position Error	0.316	0.313	0.611	5.625	11.276
50PA Refine	Fuel Used	0.001	0.001	0.003	0.024	0.049
	Fuel Planned	0.001	0.001	0.003	0.024	0.049
	Position Error	0.074	0.085	0.452	4.461	8.923
50PA Drift	Position Error	0.151	0.221	1.425	14.076	28.181
50PA-500PA	Fuel Used	0.888	0.888	0.888	0.910	0.939
	Fuel Planned	0.888	0.888	0.888	0.910	0.939
	Position Error	2.516	2.517	2.565	4.652	7.825
500PA Refine	Fuel Used	0.006	0.006	0.006	0.029	0.058
	Fuel Planned	0.006	0.006	0.006	0.029	0.058
	Position Error	0.684	0.682	0.856	4.542	8.931
500PA Drift	Position Error	1.568	1.570	1.984	10.498	20.464
500PA-5kPA	Fuel Used	8.899	8.900	8.901	8.914	8.928
	Fuel Planned	8.899	8.900	8.901	8.914	8.928
	Position Error	30.691	30.691	30.696	30.854	31.268
5kPA Refine	Fuel Used	0.065	0.065	0.065	0.073	0.090
	Fuel Planned	0.065	0.065	0.065	0.073	0.090
	Position Error	6.516	6.514	6.514	7.550	10.147
5kPA Drift	Position Error	14.917	14.936	15.184	22.118	34.170
5kPA-500IT	Fuel Used	8.924	8.924	8.925	8.946	8.980
	Fuel Planned	8.924	8.924	8.925	8.946	8.980
	Position Error	16.950	16.932	16.789	18.551	18.251
500IT Refine	Fuel Used	0.032	0.032	0.032	0.050	0.067
	Fuel Planned	0.032	0.032	0.032	0.050	0.067
	Position Error	0.581	0.583	0.685	3.842	7.603
500IT Drift	Position Error	2.528	2.529	2.602	8.098	15.503



(a) With implementation error

(b) With sensor error

Fig. 5-10: Position error per orbit for the formation-keeping simulations.



(a) With implementation error

(b) With sensor error

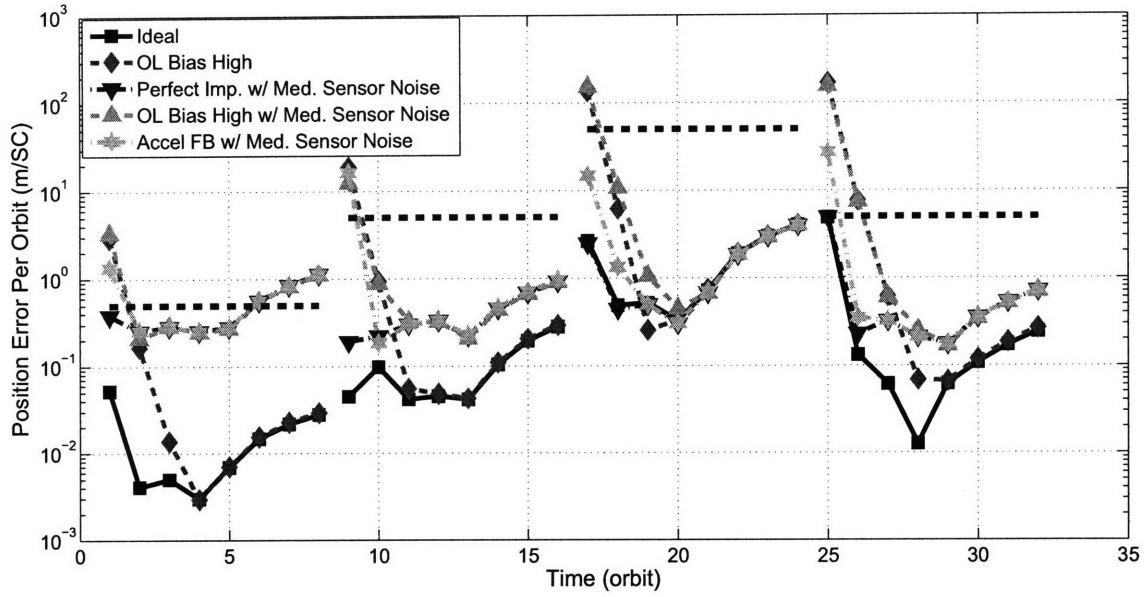
Fig. 5-11: Fuel use per orbit for the formation-keeping simulations.

Table 5.7: Formation-keeping simulation results for the LEO orbit with implementation error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

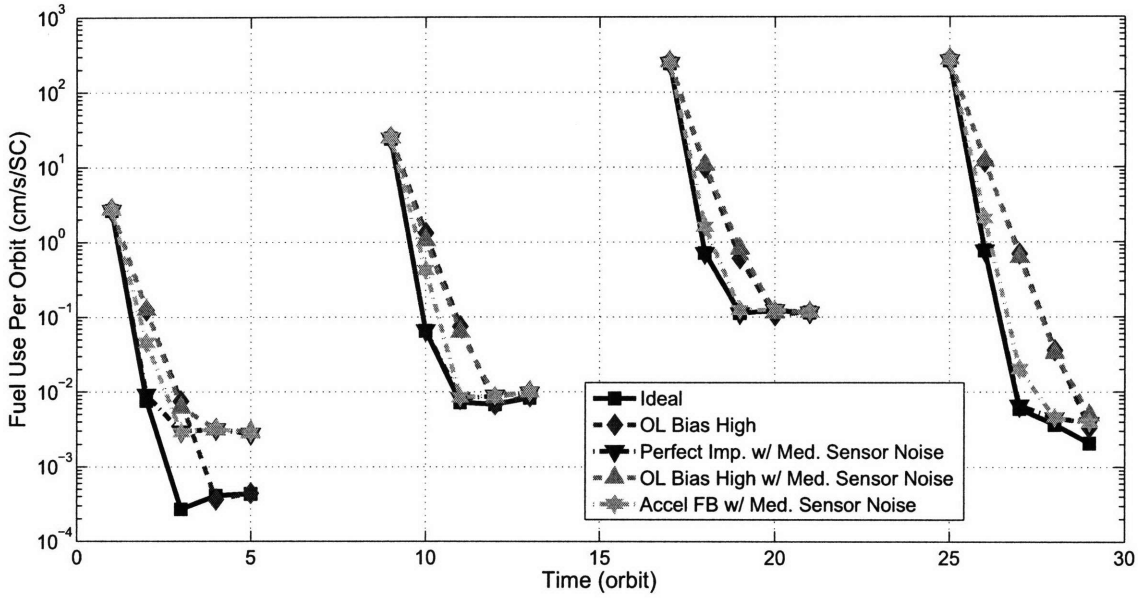
		Ideal	Accel FB	OL Bias Low	OL Bias High
Reconfiguration	Fuel Used	26.134	26.556	25.320	27.445
	Fuel Planned	26.134	26.134	26.134	26.134
	Position Error	0.053	9.386	20.702	15.034
Refine	Fuel Used	0.021	0.089	0.361	0.278
	Fuel Planned	0.021	0.087	0.366	0.262
	Position Error	0.048	0.046	0.498	0.280
Formation Keep	Fuel Used	0.008	0.008	0.008	0.008
	Fuel Planned	0.008	0.008	0.008	0.007
	Position Error	0.068	0.067	0.067	0.068

Table 5.8: Formation-keeping simulation results for the LEO orbit with sensor error. Fuel is in cm/s per spacecraft per orbit. Position errors are in meters per spacecraft per orbit.

		Ideal	Leung	Medium	Busse	Ebinuma
Reconfiguration	Fuel Used	26.134	26.134	26.134	26.135	26.136
	Fuel Planned	26.134	26.134	26.134	26.135	26.136
	Position Error	0.053	0.082	0.365	3.224	6.804
Refine	Fuel Used	0.021	0.021	0.022	0.042	0.067
	Fuel Planned	0.021	0.021	0.022	0.042	0.067
	Position Error	0.048	0.051	0.268	2.407	5.023
Formation Keep	Fuel Used	0.008	0.008	0.009	0.033	0.062
	Fuel Planned	0.008	0.008	0.009	0.033	0.062
	Position Error	0.068	0.072	0.244	2.112	4.473

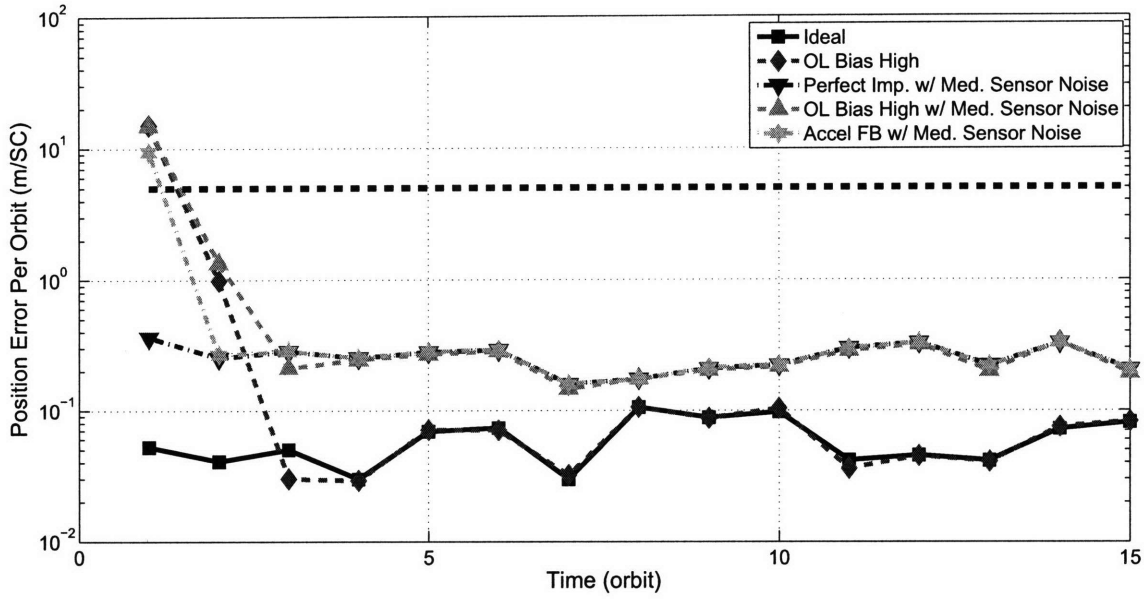


(a) Position error per orbit

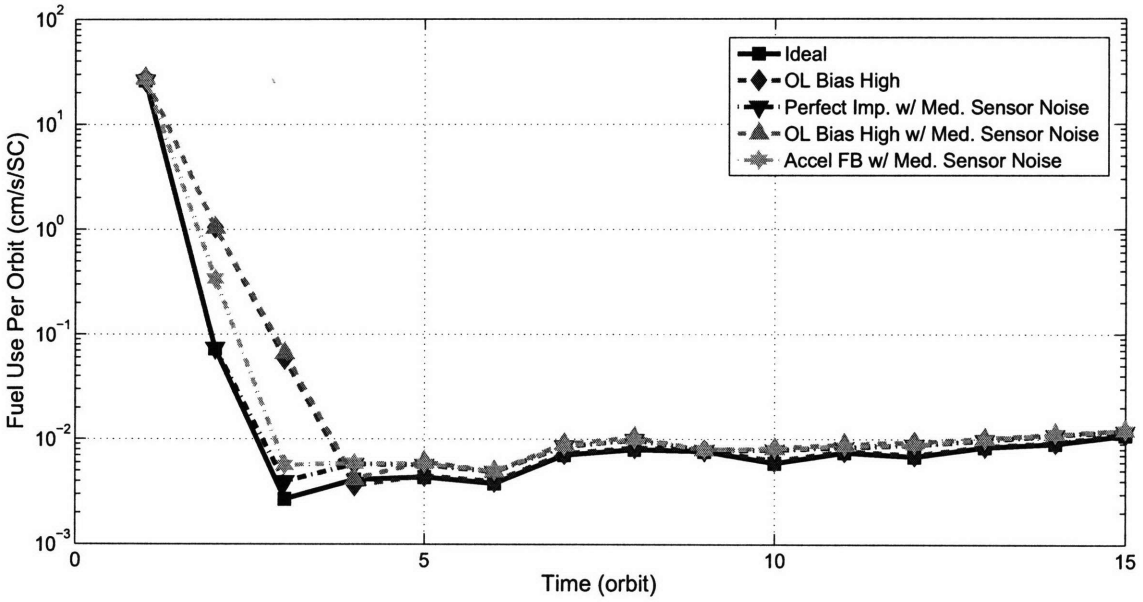


(b) Fuel use per orbit

Fig. 5-12: The LEO baseline simulation with both implementation and sensor error.



(a) Position error per orbit



(b) Fuel use per orbit

Fig. 5-13: The formation-keeping simulation with both implementation and sensor error.

Chapter 6

Conclusions

In the near future, formation flying missions will lead to exciting scientific discoveries as engineers, scientists, and researchers worldwide continue to make strides in the field. Within the next decade, missions like LISA, TPF, Darwin, and more will make observations of unprecedented precision and clarity. However, before this can happen, there are still a number of problems to solve and questions to answer. Control of a spacecraft formation requires both a method to generate plans and a way to implement those plans. This thesis expanded online planning capabilities for Earth-orbiting spacecraft formations by treating the fleet as a single entity, allowing the optimization of fleet-wide objective functions and enabling true formation flying. Recent improvements in CDGPS navigation filters have significantly reduced the uncertainty in the relative state estimation between two satellites that, until a few years ago, was the dominant source of error. This is no longer the case; the performance of the thruster subsystem and its ability to accurately implement ΔV commands is becoming increasingly important, particularly for reconfiguration maneuvers and missions with stringent inter-spacecraft position requirements.

6.1 Thesis Contributions

Fleet-wide Multi-Objective Planning

Previous work by Breger and How enabled the multi-objective online initialization of spacecraft formations, but the approach presented in [29] was limited to formation-keeping maneuvers, as the initial positions of the spacecraft were assumed to be their desired positions. Moreover, the optimizations were formulated for a single spacecraft only, and formations were initialized and maintained by solving separate optimizations for each spacecraft in the fleet. While this provided feasible solutions, there was no guarantee that combining a set of optimum plans for single spacecraft would yield a fleet-wide optimal plan. As discussed in Chapter 2, some objectives, such as fuel balancing, tend to force one spacecraft to sacrifice optimality so that another can benefit. This thesis expanded Breger's planner to solve a single optimization for the entire formation that explicitly optimizes fleet-wide objectives. Although this idea had been pursued in [28], that work utilized the Hill-Clohessy-Wiltshire equations, limiting its usefulness to formations in circular orbits with short baselines. The dynamics model contained in the fleet-wide planner developed in this thesis uses GVEs that support formations with longer baselines. Furthermore, the dynamics incorporate J_2 effects and use state transition matrices that are valid for any eccentricity [33]. In short, this planner is valid for a much broader class of formations than those that had been previously developed.

Along with the conversion to a fleet-wide planner, a number of new capabilities were introduced. The leader-follower architecture was implemented via relative position constraints, allowing the formation center to move away from the reference orbit when it is advantageous to do so. The extra degrees of freedom, can, in some cases, improve fuel use. Fuel balancing constraints were added and demonstrated to be worthwhile in certain situations, but they must be used carefully; designing intelligent formation configurations is a precursor to successful fuel balancing.

Importance of Accurate ΔV Implementation

The increasing complexity and stricter separation requirements associated with future formation flying and autonomous rendezvous and docking missions raises questions on the importance of accurately implementing planned ΔV commands. Chapter 3 explored these scenarios and showed that implementation errors of just a few percent can lead to dangerous errors for proximity operations. Two methods for improving the performance were discussed: better implementation of the plans, and replanning. The proactive solution is to implement the plans better because this prevents errors from ever happening. Replanning is a reactive solution, and only fixes problems after they happen. Moreover, replanning is not always easy nor feasible. Therefore, the proactive solution is preferred, and a simple closed-loop control system utilizing accelerometers was proposed that detects and corrects deviations from expected thruster performance. For properly calibrated accelerometers, the closed-loop system was shown to offer 100 times the performance of an open-loop system when the delivered thrust was only 2% off-nominal.

Closed-Loop Thrust Controller for SPHERES

Experimental data collected on the accuracy of the existing open-loop mixing algorithm for SPHERES uncovered the tendency of the satellites to provide less than the desired ΔV. Consistent under-performance of the actuator has undesired side effects and degrades performance. Extensive work was done for the development of a low-level closed-loop thrust controller for the SPHERES satellites. Numerous challenges were encountered and met through a variety of means. Some of these challenges, such as handling very short burns, are similar to what future missions, like TPF, are expected to face [52]. In the end, two different algorithms were proposed; the second was an improvement of the first, based on experimental data obtained from the ISS. A thruster ringing phenomenon limited the usefulness of the accelerometers for short burns, making them impossible under the first closed-loop control algorithm. The innovative second algorithm incorporated an least-squares estimation technique that

offered a continuously improving estimate of actual ΔV. Because the values from each burn are stored afterwards in separate locations (corresponding to the combination of thrusters used), the control system develops and maintains a map of expected thruster performance. This map is more accurate than the simple open-loop thruster model and can even be used for short burns when the accelerometer data is not reliable.

6.2 Recommendations for Future Work

Several formation flying missions that are proposed for the near future will fly in non-Earth orbits. The TPF spacecraft will be configured in formations similar to those studied in this thesis, with separations from 75 m to 1000 m, but they will orbit at L2. The Darwin mission will also be deployed at L2 [5]. The planner developed in this thesis only applies to spacecraft orbiting the Earth, but could be modified to work with other dynamics models, such as L2 orbits. The extension would not require too much modification as the dynamics only enter into a relatively small portion of the planner code. Much of the analysis on sensor noise and implementation error could be repeated for the L2 case and the results compared with those in this thesis.

Although fuel balancing constraints improved fuel management for specific cases, they were also shown to carry the risk of wasting fuel if ϵ was selected too aggressively. Since selection of ϵ is somewhat arbitrary, a possible improvement is to convert ϵ into a decision variable in the optimization. If the cost function were then modified to include a very small penalty on ϵ (relative to the other cost considerations), then the planner will select the smallest window that does not sacrifice performance in the other objectives. When given the choice between two plans with identical overall fuel use and geometry performance, the planner will then automatically choose the one with the smallest ϵ and hence, the most balanced fuel consumption. This sort of possibility was observed in Chapter 2, where there was a region that the window could be tightened without degrading performance in other areas.

Linear Programs easily incorporate upper bounds on actuator performance, and these constraints were used for the simulations in this thesis. However, minimum

impulse bits were not considered and are expected to have a notable influence on ΔV implementation. Since a minimum impulse constraint is nonconvex, it cannot be implemented in a pure LP, but the planner could be expanded to use binary variables to model it.

The SPHERES algorithm for improved ΔV implementation could also be used as a starting point for improved velocity estimation. Currently, the global estimator differentiates the position estimate to compute the satellite's velocity. When burns are executed, a thruster integration model based on the open-loop case is used for the initial velocity update. Chapter 4 already showed that the open-loop thruster model is suspect and does not necessarily provide a very accurate solution. Moreover, inaccurate velocity estimates are a well known driver of poor trajectory tracking and high fuel costs. Utilizing the IMU to measure burn performance could potentially improve the velocity estimate, trajectory tracking and fuel use of the satellites. The recursive least-squares algorithm derived in this thesis is easy to convert to a batch processing routine that could process burn IMU data in a single pass during the estimator update. This estimation process would be independent from the actual thrust controller itself, enabling its use with any type of thrust controller.

Appendix A

Digital Filter Overview

The naming convention for Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters stems from the behavior of the impulse response of each type. An IIR filter has an impulse response that is nonzero over an infinite length of time. An FIR filter has an impulse response that goes to zero after a finite duration.

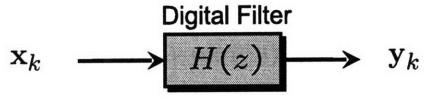


Figure A-1: A basic digital filter.

Figure A-1 illustrates a basic digital filter. At time k, a discrete measurement, in this case $\mathbf{x}_k$, is input into the filter, and the filter outputs a corresponding filtered value $\mathbf{y}_k$. The filter receives a new input at every timestep. The transfer function for a digital filter may be written as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \tag{A.1}$$

The first term in the denominator is typically scaled to be 1, as it turns out to be the multiplier for the filter output, $\mathbf{y}_k$. The frequency and phase responses of a digital filter depend on the selection of the remaining b_n and a_m coefficients, as well as the order of the filter, N. Higher order filters are able to obtain greater attenuation in the stop band, as well as a narrower transition band. The cost is an increase in the

required computation. Higher order digital filters also depend on a larger number of past measurements, and therefore might require more memory (depending on how the filter is implemented). To use a digital filter in practice, the transfer function is converted to a linear, constant-coefficient difference equation via the following z-transform.

$$\mathcal{Z}\{\delta_{k-k_0}\} = z^{-k_0} \tag{A.2}$$

The signal is sampled at specific instants, and each sample is represented by a coefficient (the value of the measurement) times the digital delta function. This makes it very easy to apply the relationship in (A.2); when the inverse transform is applied to (A.1), it yields the following result for the filter output at the k^{th} timestep.

$$\mathbf{y}_k = b_0\mathbf{x}_k + b_1\mathbf{x}_{k-1} + b_2\mathbf{x}_{k-2} + \cdots + b_N\mathbf{x}_{k-N} - a_1\mathbf{y}_{k-1} - a_2\mathbf{y}_{k-2} - \cdots - a_M\mathbf{y}_{k-M} \tag{A.3}$$

That the digital filter output is simply a combination of previous measurements and outputs makes it ideal for use with a computer. The output of an FIR filter depends only on the previous measurements, not on any previous filter outputs. In other words, all of the a coefficients in (A.1) are 0. In an IIR filter, the output depends on both the previous measurements, as well as the previous outputs. This added complexity allows lower order IIR filters to deliver performance that is comparable to higher order FIR filters. This is an attractive quality as it is possible to get better filtering for each CPU cycle, and any embedded filter must run fast enough to operate in real time. Also, infinite impulse response filters derive from analog counterparts, such as Chebyshev or Butterworth filters. These filters may be converted to to the digital domain via a bilinear transformation, and the result is a digital IIR filter.

However, IIR filters have several disadvantages when compared with FIR filters. The phase response is nonlinear, which can lead to difficulties when the output of several filtered signals must be synchronized. Conversely, FIR filters are linear phase filters, so this is not a problem for them. Because an IIR filter uses feedback (through the a coefficients), they also have the potential to go unstable. An FIR filter depends only on the last N inputs, and therefore does not go unstable (it has no poles). A

related issue is the sensitivity of an IIR filter to numerical quantization. For IIR filters higher than about 2nd or 3rd order, rounding errors in the coefficients can drastically alter filter performance, in some cases even leading to instability [89]. This type of instability was actually observed while evaluating filter code on the SPHERES hardware. The instability problem can be handled by constructing high-order IIR filters as cascading combinations of 1st and 2nd order IIR filters, but this creates a more complicated structure.

Bibliography

[1] Daniel P. Scharf, Fred Y. Hadaegh, and Scott R. Ploen. A survey of spacecraft formation flying guidance and control (Part II): Control. In *Proceedings of the American Control Conference*, pages 2976–2985, Boston, Massachusetts, June–July 2004.

[2] David Folta. Results of NASA's first autonomous formation flying experiment: Earth Observing-1 (EO-1). In *AIAA/AAS Astrodynamics Specialist Conference*, Monterey, California, August 2002. AIAA Paper 2002-4743.

[3] Candace Carlisle and Evan H. Webb. Space Technology 5 - a successful micro-satellite constellation mission. In *21st Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, August 2007.

[4] Graeme L. Stephens, Deborah G. Vane, Ronald J. Boain, Gerald G. Mace, Kenneth Sassen, Zhien Wang, Anthony J. Illingworth, Ewan J. O'Connor, William B. Rossow, Stephen L. Durden, Steven D. Miller, Richard T. Austin, Angela Benedetti, and Cristian Mitrescu. The Cloudsat mission and the A-Train: A new dimension of space-based observations of clouds and precipitation. *Bulletin of the American Metereological Society*, 83(12):1771–1790, December 2002.

[5] C. V. Fridlund and Philippe Gondoin. Darwin mission. In *Proceedings of SPIE*, volume 4852, pages 394–404, 2003.

[6] P.R. Lawson. The Terrestrial Planet Finder. In *Proceedings of the IEEE Aerospace Conference*, volume 4, pages 4–2005–4–2011, March 2001.

[7] W.M Folkner. The Laser Interferometer Space Antenna mission. In *AIAA Space 2000 Conference & Exposition*, Long Beach, CA, September 2000. AIAA Paper 2000-5129.

[8] T. Chabot and B. Udrea. XEUS mission guidance navigation and control. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, Colorado, August 2006. AIAA Paper 2006-6587.

[9] K.C. Gendreau, W.C. Cash, A.F Shipley, and N. White. The MAXIM pathfinder X-ray interferometry mission. In *Proceedings of SPIE*, volume 4851, pages 353–364, March, 2003.

[10] J. Borde, F. Teston, S. Santandrea, and S. Boulade. Feasibility of the PROBA 3 formation flying demonstration mission as a pair of microsats in GTO. In *55th International Astronautical Congress*, Vancouver, Canada, 2004.

[11] Staffan Persson, Bjorn Jacobsson, and Eberhard Gill. PRISMA demonstration mission for advanced rendezvous and formation flying technologies and sensors. In *56th International Astronautical Congress*, Fukuoka, Japan, 2005. IAC-05-B5.6.B.07.

[12] Michael E. Polites. Technology of automated rendezvous and capture in space. *Journal of Spacecraft and Rockets*, 36(2):280–291, March–April 1999.

[13] Isao Kawano, Masaaki Mokuno, Toru Kasai, and Takashi Suzuki. Result of autonomous rendezvous docking experiment of engineering Test Satellite-VII. *Journal of Spacecraft and Rockets*, 38(1):105–111, January–February 2001.

[14] Lionel Baize, Martial Vanhove, Pascale Flagel, and Alberto Novelli. The ATV Jules Verne supplies the ISS. In *SpaceOps 2008 Conference*, May 2008. AIAA Paper 2008-3537.

[15] Alan Lindenmoyer. Commercial orbital transportation services (COTS) demonstrations. In *Space 2006*, San Jose, California, September 2006. AIAA Conference presentation.

[16] Timothy E. Rumford. Demonstration of Autonomous Rendezvous Technology (DART) project summary. In *Proceedings of SPIE*, volume 5088, pages 10–19, 2003.

[17] Richard T. Howard and Thomas C. Bryan. DART AVGS performance. In *SPIE Defense and Security Symposium*, 2007.

[18] Geoffrey Hintze, Keith G. Cornett, Michael H. Rahmatipour, Andrew F. Heaton, Larry E. Newman, Kevin D. Fleischmann, and Byron J. Hamby. AVGS, AR&D

for satellites, ISS, the moon, mars and beyond. In *Infotech@Aerospace*, Rohnert Park, California, May 2007. AIAA Paper 2007-2883.

[19] G. W. Hill. Researches in the lunar theory. *American Journal of Mathematics*, 1(1):5–26, 129–147, 245–260, 1878.

[20] W.H. Clohessy and P.S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of Aerospace Sciences*, 27:653–658, 674, September 1960.

[21] Derek F. Lawden. *Optimal Trajectories for Space Navigation.* Butterworths, 1963.

[22] Samuel A. Schweighart and Raymond J. Sedwick. High-fidelity linearized J2 model for satellite formation flight. *Journal of Guidance, Control and Dynamics*, 25(6):1073–1080, November–December 2002.

[23] Andrew Robertson, Gokhan Inalhan, and Jonathan P. How. Spacecraft formation flying control design for the ORION mission. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1562–1575, 1999. AIAA Paper 99-4266.

[24] Mehran Mesbahi and Fred Y. Hadaegh. Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching. *Journal of Guidance, Control and Dynamics*, 24(2):369–377, March–April 2001.

[25] Donald T. Stansbery and James R. Cloutier. Nonlinear control of satellite formation flight. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Denver, Colorado, August 2000. AIAA Paper 2000-4436.

[26] Hanspeter Schaub and Kyle T. Alfriend. Impulsive feedback control to establish specific mean orbit elements of spacecraft formations. *Journal of Guidance, Control, and Dynamics*, 24(4):739–745, July–August 2001.

[27] Daniel Chavez Clemente and Ella M. Atkins. Optimization of a tetrahedral satellite formation. *Journal of Spacecraft and Rockets*, 42(4):699–710, July–August 2005.

[28] Michael Tillerson, Gokhan Inalhan, and Jonathan P. How. Co-ordination and control of distributed spacecraft systems using convex optimization techniques. *International Journal of Robust and Nonlinear Control*, 12:207–242, 2002.

[29] Louis Breger and Jonathan P. How. Gauss's variational equation-based dynamics and control for formation flying spacecraft. *Journal of Guidance, Control, and Dynamics*, 30(2):437–448, March–April 2007.

[30] S. Curtis. The Magnetospheric Multiscale Mission...resolving fundamental processes in space plasmas. Technical report, NASA Goddard Space Flight Center, Greenbelt, MD, December 1999. NASA/TM2000-209883.

[31] John L. Junkins, M.R. Akella, and Kyle T. Alfriend. Non-gaussian error propagation in orbital mechanics. *Journal of the Astronautical Sciences*, 44(4):541–563, October–December 1996.

[32] Dimitris Bertsimas and John N. Tsitsiklas. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[33] Dong-Woo Gim and Kyle T. Alfriend. State transition matrix of relative motion for the perturbed noncircular reference orbit. *Journal of Guidance, Control, and Dynamics*, 26(6):956–971, November–December 2003.

[34] Pini Gurfil. Control-theoretic analysis of low-thrust orbital transfer using orbital elements. *Journal of Guidance, Control, and Dynamics*, 26(6):979–983, November–December 2003.

[35] Simone D'Amico, Eberhard Gill, and Oliver Montenbruck. Relative orbit control design for the PRISMA formation flying mission. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, Colorado, August 2006. AIAA Paper 2006-6067.

[36] Eberhard Gill, Simone D'Amico, and Oliver Montenbruck. Autonomous formation flying for the PRISMA mission. *Journal of Spacecraft and Rockets*, 44(3):671–681, May–June 2007.

[37] Jonathan P. How and Michael Tillerson. Analysis of the impact of sensor noise on formation flying control. In *Proceedings of the American Control Conference*, pages 3986–3991, Arlington, Virginia, June 2001.

[38] Michael Tillerson, Louis Breger, and Jonathan P. How. Distributed coordination and control of formation flying spacecraft. In *Proceedings of the American Control Conference*, pages 1740–1745, Denver, Colorado, June 2003.

[39] Michael Tillerson. Coordination and control of multiple spacecraft using convex optimization techniques. Master's thesis, Massachusetts Institute of Technology, June 2002.

[40] Louis Breger. Model predictive control for formation flying spacecraft. Master's thesis, Massachusetts Institute of Technology, June 2004.

[41] Louis Breger, Jonathan P. How, and Kyle T. Alfriend. Partial J2-invariance for spacecraft formations. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2006. AIAA Paper 2006-6585.

[42] Jonathan P. How, Robert Twiggs, D. Weidow, K. Harman, and F. Bauer. Orion - a low-cost demonstration of formation flying in space using GPS. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pages 276–286, Boston, Massachusetts, August 1998. Collection of Technical Papers (A98-37348 10-13), Reston, VA, American Institute of Aeronautics and Astronautics, 1998, p. 276-286.

[43] Hanspeter Schaub and Kyle T. Alfriend. J2 invariant relative orbits for spacecraft formations. In *Flight Mechanics Symposium*, Greenbelt, Maryland, May 1999.

[44] S. R. Vadali, Kyle T. Alfriend, and S. S. Vaddi. Hills equations, mean orbital elements, and formation flying of satellites. In *American Astronautical Society*, March 2000. AAS Paper 00-258.

[45] Louis Breger, Jonathan P. How, and Arthur G. Richards. Model predictive control of spacecraft formations with sensing noise. In *Proceedings of the American Control Conference*, pages 2385–2390, Portland, Oregon, June 2005.

[46] Franz Busse, Jonathan P. How, and James Simpson. Demonstration of adaptive extended kalman filter for low earth orbit formation estimation using CDGPS. In *Institute of Navigation GPS Meeting*, Portland, Oregon, September 2002.

[47] Suny Leung and Oliver Montenbruck. Real-time navigation of formation-flying spacecraft using global-positioning-system measurements. *Journal of Guidance, Control, and Dynamics*, 28(2):226–235, March–April 2005.

[48] David Berthelier, Xavier Clerc, Mathieu Chaize, and Patrick Delpy. Qualification of the automated tansfer vehicle (ATV) flight control. In *Space 2006*, September 2006. AIAA Paper 2006-7266.

[49] T.J. Barber and R.T. Cowley. Initial Cassini propulsion system in-flight characterization. In *AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, July 2002. AIAA Paper 2002-4152.

[50] Danny C. Lam, Kenneth H. Friberg, Jay M. Brown, Siamak Sarani, and Allan Y. Lee. An energy burn algorithm for Cassini saturn orbit insertion. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, August 2005. AIAA Paper 2005-5994.

[51] Raymond B. Frauenholz, Ramachandra S. Bhat, Steven R. Chesley, Nickolaos Mastrodemos, William M. Owen Jr., and Mark S. Ryne. Deep Impact navigation system performance. *Journal of Spacecraft and Rockets*, 45(1):39–56, January–February 2008.

[52] D. Scharf, Fred Y. Hadaegh, Zahidul H. Rahman, Joel F. Shields, and Gurkipal Singh. An overview of the formation and attitude control system for the Terrestrial Planet Finder formation flying interferometer. In *International Symposium on Formation Flying Missions and Technologies*, Washington, D. C., September 2004.

[53] J. Leitner, F. Bauer, D. Folta, M. Moreau, R. Carpenter, and Jonathan P. How. Distributed spacecraft systems develop new GPS capabilities. *GPS World: Formation Flight in Space.*, February 2002.

[54] Dean C. Tsai. The effects of thrust uncertainty and attitude knowledge errors on the MMS formation maintenance maneuver. Technical report, NASA Goddard Space Flight Center, 2005. Document ID: 20050243598.

[55] Chris Sabol, Rich Burns, and Craig A. McLaughlin. Satellite formation flying design and evolution. *Journal of Spacecraft and Rockets*, 38(2):270–278, March–April 2001.

[56] R.H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics.* AIAA Education Series. AIAA, New York, 1987.

[57] Louis Breger. *Control of Spacecraft in Proximity Orbits.* PhD thesis, Massachusetts Institute of Technology, 2007.

[58] Gary Blackwood, Curt Henry, Eugene Serabyn, Serge Dubovitsky, MiMi Aung, and Steven M. Gunter. Technology and design of an infrared interferometer for

the Terrestrial Planet Finder. In *Space 2003*, Long Beach, California, September 2003. AIAA Paper 2003-6329.

[59] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems.* AIAA Education Series. AIAA, Reston, VA, 2003.

[60] Randal M. Beard, Timothy W. McLain, and Fred Y. Hadaegh. Fuel optimization for constrained rotation of spacecraft formations. *Journal of Guidance, Control, and Dynamics*, 23(2):339–346, March–April 2000.

[61] Amirreza Rahmani, Mehran Mesbahi, and Fred Y. Hadaegh. Optimal balanced-energy formation flying maneuvers. *Journal of Guidance, Control, and Dynamics*, 29(6):1395–1403, November–December 2006.

[62] S. R. Vadali, S. S. Vaddi, and Kyle T. Alfriend. An intelligent control concept for formation flying satellites. *International Journal of Robust and Nonlinear Control*, 12(2):97–115, 2002.

[63] Wei Ren and Randal M. Beard. Decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82, January–February 2004.

[64] GLPK (GNU Linear Programming Kit). Online at `http://www.gnu.org/software/glpk`, last accessed May 2008.

[65] GLPKMEX. Online at `http://glpkmex.sourceforge.net`, last accessed May 2008.

[66] COIN-OR Linear Program Solver. Online at `http://www.coin-or.org/Clp`, last accessed May 2008.

[67] Robin Lougee-Heimer. The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47(1):57–66, January 2003.

[68] mexclp. Online at `http://control.ee.ethz.ch/~joloef/clp.php`, last accessed May 2008.

[69] MOSEK. Online at `http://www.mosek.com`, last accessed May 2008.

[70] ILOG CPLEX. Online at `http://www.ilog.com/products/cplex`, last accessed May 2008.

[71] S. S. Vaddi, Kyle T. Alfriend, S. R. Vadali, and P. Sengupta. Formation establishment and reconfiguration using impulsive control. *Journal of Guidance, Control, and Dynamics*, 28(2):262–268, March–April 2005.

[72] Robert F. Stengel. *Optimal Control and Estimation.* Dover Publications, 1994.

[73] R.H. Tolson, G.M Keating, R.W. Zurek, S.W. Bougher, C.G. Justus, and D.C. Fritts. Application of accelerometer data to atmospheric modeling during Mars aerobraking operations. *Journal of Spacecraft and Rockets*, 44(6):1172–1179, November–December 2007.

[74] Gene F. Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, 3rd edition, 1998.

[75] Marshall H. Kaplan. *Modern Spacecraft Dynamics and Control.* Wiley, 1976.

[76] Allen Chen, Alvar Saenz-Otero, Mark O. Hilstad, and David Miller. Development of formation flight and docking algorithms using the SPHERES testbed. In *15th Annual USU Conference on Small Satellites.* AIAA/USU, August 2001.

[77] Simon Nolet, Edmund Kong, and David Miller. Autonomous docking algorithm development and experimentation using the SPHERES testbed. In *Space Platforms and Infrastructure*, volume 5419, pages 1–15. SPIE, August 2004.

[78] Simon Nolet. The SPHERES navigation system: from early development to on-orbit testing. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007. AIAA Paper 2007-6354.

[79] Simon Nolet, Alvar Saenz-Otero, David Miller, and Amer Fejzic. SPHERES operations aboard the ISS: Maturation of GN&C algorithms in microgravity. In *30th Annual AAS Guidance and Control Conference*, Breckenridge, Colorado, February 2007.

[80] Simon Nolet. *Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite.* PhD thesis, Massachusetts Institute of Technology, June 2007.

[81] Allen Chen. Propulsion system characterization for the SPHERES formation flight and docking testbed. Master's thesis, Massachusetts Institute of Technology, June 2002.

[82] Mark O. Hilstad. A multi-vehicle testbed and interface framework for the development and verification of separated spacecraft control algorithms. Master's thesis, Massachusetts Institute of Technology, June 2002.

[83] QRS14 (GyroChip II) Micromachined Angular Rate Sensor. Online at `http://www.systron.com/PDFS/datasheets/QRS14.pdf`, last accessed May 2008.

[84] QA-750 Q-Flex Accelerometer. Online at `http://www.inertialsensor.com/qa750.shtml`, last accessed May 2008.

[85] Mark O. Hilstad, John P. Enright, and Arthur G. Richards. *The SPHERES Guest Scientist Program.* Space Systems Laboratory, October 2003.

[86] Bong Wie. *Space Vehicle Dynamics and Control.* AIAA Education Series. American Institute of Aeronautics and Astronautics, Inc., 1801 Alexander Bell Drive, Reston, VA, 20191, 1998.

[87] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, January 1986.

[88] Texas Instruments. *TMS320C6000 Programmers Guide*, 2006.

[89] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing.* Signal Processing. Prentice-Hall, 2nd edition, 1999.

[90] A. I. Solutions. *FreeFlyer User's Guide.* A. I. Solutions, March 1999.

[91] J.R Carpenter, J. Leitner, D. Folta, and Richard D. Burns. Benchmark problems for spacecraft formation flying missions. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003. AIAA Paper 2003-5364.

[92] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control.* John Wiley & Sons, Inc., 2006.

[93] T. Ebinuma. *Precision Spacecraft Rendezvous Using Global Positioning System: An Integrated Hardware Approach.* PhD thesis, University of Texas, Austin, TX., August 2001.