Distributed Estimation and Control Technologies for Formation Flying Spacecraft

by

Philip Andrew Ferguson

Bachelor of Applied Science - Aerospace Engineering
University of Toronto, June 2000

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

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2003

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SEP 1 0 2003
LIBRARIES

© Massachusetts Institute of Technology 2003. All rights reserved.

Author .....
Department of Aeronautics and Astronautics
January 29, 2003

Certified by .....
Jonathan P. How
Associate Professor
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
H.N. Slater Professor
Chairman, Department Committee on Graduate Students

AERO

Distributed Estimation and Control Technologies for Formation Flying Spacecraft

by

Philip Andrew Ferguson

Submitted to the Department of Aeronautics and Astronautics
on January 29, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Many future space missions, such as space-based radar, earth mapping, and interferometry, will require formation flying of multiple spacecraft to achieve their very advanced science objectives. While formation flying offers many performance and operational advantages, there are several challenges that must be addressed, including navigation, control, autonomy, distributed data management, efficient inter-vehicle communication, and robustness. One of the key issues with formation flying of large fleets is selecting the overall system architecture, because it drives the distribution of the various algorithms and the extent to which data must be transmitted.

These challenges are particularly evident with the relative navigation. While carrier-phase differential GPS can be used as a highly accurate sensor for LEO formations, it is not sufficient as the sole sensor for missions beyond LEO. If local ranges and range rates are used to augment or replace the GPS measurements, precise estimation can continue into MEO and beyond. However these new measurements complicate the estimator decentralization by coupling the vehicles' state estimates.

This thesis explores solutions to many of these challenges within the context of the *Orion* microsatellite formation flying mission. It also presents the *Formation Flying Information Technology* testbed, developed to evaluate the communication and computational requirements associated with various system architectures when using augmented GPS. Several architectures and their associated estimation algorithms are also analyzed and compared in terms of performance, computation, and communication requirements. This analysis clearly shows that the decentralized reduced-order filters provide near optimal estimation without excessive communication or computation requirements. Embedding these reduced-order estimators within the hierarchic architecture presented should also permit scaling of the relative navigation to very large fleets.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Acknowledgments

In carrying out the research that went into this Master's thesis, there were several key individuals that played large roles in helping me make it to the end. This was a long and difficult road at times and I thank everyone whole-heartedly for their kindness and support over the past two years.

Firstly, I would like to thank my advisor, Professor Jonathan P. How for directing and guiding me through this research. Professor How's strong will and drive for excellence kept me on track, studying new and exciting topics along the way. Professor How taught me the keys to effective research, and for that, I thank him. I also thank Professor How and the NASA Goddard Spaceflight Center for their continued funding through this research project (funded in part under Air Force grant F49620-99-1-0095, NASA grant NAG5-10719 and NASA grant NAG5-10440).

Next, I would like to thank my research colleagues, among them, Franz Busse, Chanwoo Park, Nick Pohlman, Arthur Richards, John Bellingham, Louis Breger, Ian Garcia, Ellis King and Megan Mitchell. Thanks also to my friend and Professor How's administrative assistant, Margaret Yoon for her support throughout this time. Their help and guidance along the way was completely invaluable to me. They offered everything from editing help, to technical and moral support (sometimes over a civilized pint). Thanks all for making this job easier on me.

Throughout my entire life and especially through this time, my parents and brother and sister have always played an important role in supporting me and keeping my spirits up when things get low. Thank you to Mom, Dad, Matthew and Catherine (LYP).

Finally, I would like to thank my beautiful, loving wife, Ally. Her love and support has been absolutely unending through this and every endeavor I take. She has put up with me in the best of moods when everything is working and in the worst of moods when things work less than perfectly. She is my everything and without her, none of what I do would be possible. Thank you Ally. I love you.

Contents

1 Introduction **15**

1.1 Challenges 15

1.2 Existing Technologies 18

1.3 New Contributions 19

1.4 Thesis Outline 19

2 Orion Hardware and Software Design **21**

2.1 Overview 21

2.2 Mission Description 23

2.2.1 Modes 23

2.2.2 Stages 25

2.2.3 Resource Budgets 31

2.3 The Orion Spacecraft 35

2.3.1 Structure 36

2.3.2 Propulsion 36

2.3.3 Position and Attitude Determination System 36

2.3.4 Attitude Control System 39

2.3.5 Command and Data Handling CPU 40

2.3.6 Communications 40

2.3.7 Data Bus 40

2.3.8 Power Subsystem 41

2.3.9 Science Computer and GPS Interface 42

2.3.10 Software 46

2.3.11 Orion Payload Architecture 48

2.4 Mission Status 50

3 The Formation Flying Information Technology Testbed **51**
3.1 Introduction 51
3.2 Architectures 53
3.3 Algorithms 54
3.3.1 Estimation Algorithms 55
3.3.2 Coordination Algorithms 57
3.3.3 Formation-keeping Control 58
3.3.4 Thrust Mapping and Fault Detection 60
3.3.5 Autonomous Task Allocation 61
3.4 Example Architectures 62
3.4.1 Example Architecture 1 - Centralized Estimator with Distributed Controller 62
3.4.2 Example Architecture 2 - Decentralized Estimator with Distributed Controller 64
3.5 FFIT Testbed 65
3.6 Simulation Results 69
3.7 Conclusions 73

4 Decentralized Estimation Techniques **77**
4.1 Introduction 77
4.2 Centralized Architectures 79
4.2.1 Kalman Filter 79
4.2.2 Information Filter 81
4.3 Decentralized Architectures 82
4.3.1 Full Order Decentralized Filters 82
4.3.2 Reduced-Order Decentralized Filters 86
4.4 Hierarchic Clustering 89
4.5 Analysis 92
4.5.1 Two-Vehicle, Two Dimensional Covariance Analysis 92
4.5.2 Multi-Vehicle Simulation Results 96
4.5.3 Data Flow Performance Validation 102
4.6 Conclusions 108

5 Conclusions **111**
5.1 Orion 111

5.2 Distributed GNC Architectures 112
5.3 Decentralized Estimation Using GPS Augmented with Local Ranging 112
5.4 Future Work 113
5.5 Closing Remarks 114

List of Figures

1-1 Algorithmic Structure for Formation Flying 16

2-1 Orion Mission Timeline 25
2-2 Orion-Emerald Launch Sequence 26
2-3 Orion and Emerald on MSDS 27
2-4 Battery Capacity 33
2-5 Current Input 33
2-6 Orion Spacecraft Propulsion 37
2-7 Orion Spacecraft Avionics 38
2-8 Orion GPS Receiver 39
2-9 Orion Science Computer 41
2-10 Token Bus Electronics 43
2-11 Token Bus Finite State Machine 44
2-12 Polling Finite State Machine 45
2-13 Orion Software Architecture 47
2-14 Orion Payload Architecture 49

3-1 Formation-keeping Control of a Satellite 59
3-2 Example Architecture 1 63
3-3 Example Architecture 2 64
3-4 Physical Architecture of the FFIT Testbed 66
3-5 FFIT Testbed Photo 67
3-6 Relative GPS Position and Velocity Errors. 69
3-7 FLOP Profiles for Basic Estimation 70
3-8 Effect of a Thruster Failure 71
3-9 FLOP History without using Task Allocation 72
3-10 FLOP History using Task Allocation 72

3-11 Typical Slave FLOP Profiles 74
3-12 Bytes Sent from Master 74

4-1 GPS Estimation with Local Ranging Augmentation 78
4-2 Centralized Algorithmic Flow 80
4-3 Information Filter Linearization Effect 84
4-4 Cascade Algorithmic Flow 85
4-5 Hierarchic Clustering Topology 90
4-6 Two Dimensional Problem Geometry 93
4-7 Error Ellipse Analysis 94
4-8 Two-Dimensional Vehicle Motion 98
4-9 Decentralized Methods Position Error Comparison 101
4-10 Estimation Error vs. Computation 102
4-11 Estimation Error vs. Communication 103
4-12 Estimation Error vs. Solution Time 104
4-13 Scaling Trends for $N = 4$ 106
4-14 Scaling Trends for $N \gg 4$ 107

List of Tables

2.1 Orion Power Budget 29
2.2 Orion Mass Budget 35

4.1 Hierarchic Error Scaling 91

Chapter 1

Introduction

The concept of autonomous formation flying of satellite clusters has been identified as an enabling technology for many future NASA and the U.S. Air Force missions [1, 2, 3, 4]. Examples include the Earth Orbiter-1 (EO-1) mission that is currently on-orbit [1, 5], StarLight (ST-3) [6], the Nanosat Constellation Trailblazer mission [7], the Air Force TechSat-21 [8] distributed SAR and the Orion Formation Flying mission [9]. The use of fleets of smaller satellites instead of one monolithic satellite will improve the science return since it provides:

- Coordinated observations from different perspectives.
- Time synchronous observations from spatially distributed instruments.
- Multiple spacecraft to operate as a single instrument.
- Redundancy and reconfigurability in the event of a single vehicle failure.

If the ground operations can also be replaced with autonomous onboard control, this fleet approach should also decrease the mission cost.

1.1 Challenges

There are many challenges to be resolved before the ambitious objectives described above can be achieved, including modeling fleet orbital dynamics, design of fuel efficient controllers, and development of onboard fleet-level autonomy. Relative navigation will also play a crucial role in achieving these mission goals because it will provide the information necessary to perform closed-loop real-time control and the (potentially much more precise) off-line determination of the vehicle relative states for post-processing of the science data.

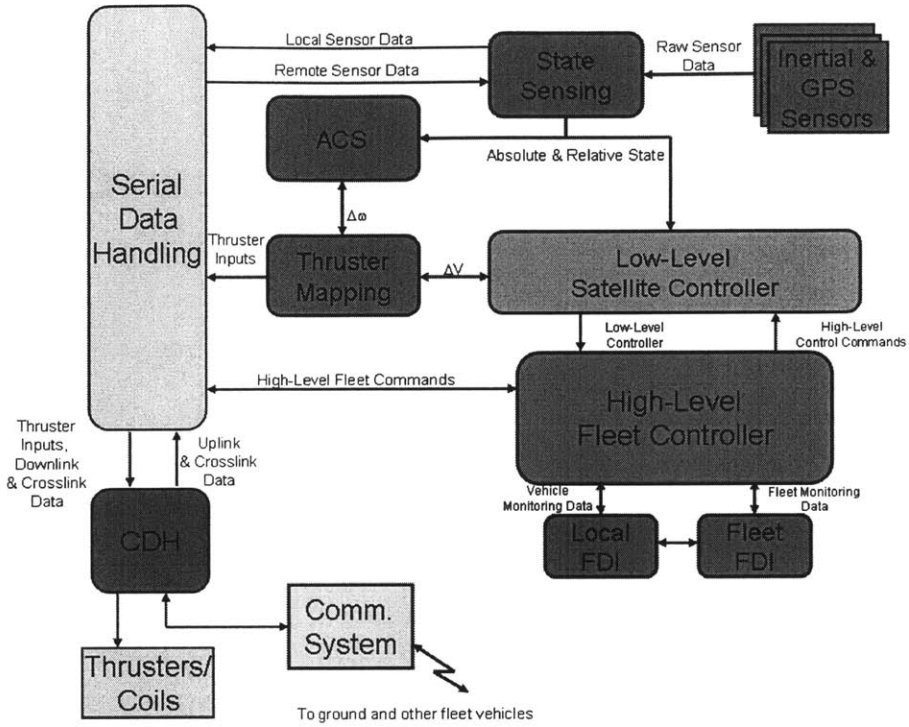


Fig. 1-1: Algorithmic Connectivity Required for Typical Formation Flying Missions.

Future formation flying missions have been proposed at LEO, GEO, in highly elliptical Earth orbits, at L2, and in deep space. While some missions consist of a pair of vehicles (e.g., GRACE and Starlight), others have been proposed with as many as 34 spacecraft (e.g., MAXIM), and some future plans call for missions with as many as 100 spacecraft (e.g., Stellar Imager). Relative navigation for these missions will be challenging because:

1. GPS (a typical "baseline" solution [10]) will not be available at the higher altitude orbits.
2. Carrier-differential GPS (CDGPS) might not provide sufficiently accurate measurements for the science data.
3. The large number of vehicles in the fleet will significantly complicate the estimator design for the relative navigation.

In addition to the the estimation challenges, formation flying is inherently a distributed problem, and achieving the mission goals requires the tight integration of

various algorithms. Figure 1-1 shows the complicated information flow between the various estimation, coordination, and control algorithms for a typical formation flying control system [11]. Several of these algorithms can naturally be *decentralized*, but others require combined or fleet information, and thus must be performed within a *centralized* or *hierarchic* architecture.

These architectures differ by the degree to which the algorithms are *distributed*. The raw measurement data for estimation and control is typically collected in a decentralized manner (*i.e.,* each vehicle takes measurements that pertain only to its own state), strongly suggesting decentralized estimators and/or controllers to handle the data. Dividing estimation, coordination, or control algorithms for distribution across the fleet can provide benefits such as improved robustness, increased flexibility, reduced computation time, and improved autonomy. Parallel processing, if scaled properly, could dramatically reduce the computation time compared to a completely centralized architecture. Also, the modularity inherent in distributed architectures usually lends itself easily to expansion.

These benefits of distributed architectures, however, must be weighed against the disadvantages, such as increased inter-spacecraft communications, possible non-determinacy of solutions (*synchronization*), and higher mission risk stemming from the increase in overall architectural complexity. They key issue here is information management, as significant communication of both raw data (*e.g.,* GPS carrier phase measurements) and solutions (*e.g.,* estimated positions and velocities, coordination requirements) must be shared.

While the issues described above apply to all types of formation flying missions, some of the key challenges to many current and proposed formation flying missions are a result of the mission constraints imposed by the nano- or microsatellites spacecraft designs [9]. The decision to build and fly nanosatellies as part of a formation flying mission is common to many universities, government and military organizations due to their low manufacture cost, ease of operation and small mass (which translates into relatively inexpensive launch costs). However, along with these benefits come the drawbacks of limited power, mass, cost and size restraints. All of these constraints impose restrictions on the bandwidth, communication distance and computational capacity (among others) that can be achieved. While the work presented in this thesis is applicable to control and estimation archictectures for fleets of many different types of spacecraft, the primary focus will be on fleets of nanosatellites, as inspired by the

Orion formation flying mission [9].

1.2 Existing Technologies

The field of formation flying technologies is a very rich one indeed, lending itself to a great many research topics and papers [12, 13]. Work is being done at MIT, Cornell and other universities on navigation and control to prevent collisions, plume impingement and excessive fuel usage. Research is ongoing to invent fault detection and isolation routines that can sense failures and reconfigure systems to regain functionality [14, 15]. In addition, highly accurate GPS estimators have been developed to estimate the relative positions and velocities to within 1 cm and 0.5 mm/s [16, 17, 18]. However, what has yet to be studied is the intricate interactions between all of the different algorithms required to execute a successful formation flying mission. Furthermore, much of the research to date has focussed on missions in LEO with relatively small fleets (*i.e.,* less than 4 vehicles).

Several testbeds have been developed to focus on demonstrations of the basic formation flying concepts [19, 20, 21], testing the implementation of the real-time code [22] and integrating actual flight hardware in the loop [23, 24]. However, none of these testbeds directly address the inter-spacecraft communication expected on future formation flying missions, which could be a key factor in comparing control and estimation architectures due to the cost, power, mass and expandability issues that arise when choosing inter-spacecraft communication systems for small and cheap microsatellites.

Other formation flying navigation research has focused on augmenting GPS measurements with local ranging devices [25, 26, 27, 28, 29]. While useful for extending fleet estimation beyond LEO, the nonlinear measurements, by definition, couple the state estimates of each vehicle, complicating the algorithmic decentralization possible in Ref. [26, 16]. A method for approximating the decentralization was presented in Ref. [26], but its application was limited to ranging devices with accuracy that was similar to CDGPS. Furthermore, the algorithm required multiple iteration steps across the fleet making the method less attractive for large fleets. Other methods of algorithmic decentralization have been explored using information filters [30, 31]. These methods provide solutions identical to the centralized solutions, but require substantial amounts of communication and computation throughout the fleet.

1.3 New Contributions

The work presented in this thesis develops the tools and insight required to verify, test, demonstrate and, in the case of sensing, extend the formation flying technology toolset into a flexible, decentralized framework. This thesis presents the basic requirements of a formation flying mission using Orion as the primary example. A software framework for supporting the autonomous formation flying algorithms is presented along with a description of the key hardware components.

Following a discussion of the required algorithms, a testbed is presented that provides a unique set of capabilities. The *Formation Flying Information Technology* (FFIT) testbed is shown to be a valuable tool for evaluating the communication and computational aspects of various estimation and control architectures.

The last part of this thesis focuses directly on the problem of estimator decentralization in the presence of local ranging augmentation. Several different types of decentralized estimators are compared using the FFIT testbed, including a reduced-order decentralized estimator based on the Schmidt-Kalman filter [32]. Results from the FFIT testbed are presented that indicate the Schmidt-Kalman filter (and its variants) provides the best combination of estimator accuracy, communication and computation. Futhermore, these estimators are shown to be scalable to large fleets using hierarchic architectures.

This research has advanced the field of formation flying by creating an autonomous software framework for the Orion Formation Flying mission, building a testbed to analyze the data flow interactions between the required formation flying algorithms and augmenting the decentralized estimation technologies to make them applicable to future mission scenarios beyond LEO. With the insight gained from the research presented in this thesis, mission designers will be able to better construct control and estimation architectures for large fleets of vehicles in and beyond the range of GPS.

1.4 Thesis Outline

Chapter 2 introduces the Orion Mission in the context of other current and proposed formation flying missions. The Orion timeline and associated activities are presented. Following this discussion, the Orion hardware is presented along with a detailed description of the work done on the software design for the mission.

Chapter 3 presents the FFIT testbed designed and built to analyze formation

flying control and estimation architectures from a data flow point of view. Following a discussion of the motivation for and technical description of the testbed, some results are presented from one particular estimation and control architecture to illustrate the usefulness of the FFIT testbed.

The estimation algorithm used to demonstrate the FFIT testbed in Chapter 3 is extended to include local ranging data to augment the measurements in Chapter 4. Several iterative decentralized estimators are proposed that attempt to account for the coupling effect of the nonlinear local ranging measurements and produce estimates that approach the centralized performance. The FFIT testbed is used to analyze the algorithms from a computation and communication standpoint. Chapter 5 concludes this thesis with a summary of my contributions.

Chapter 2

Orion Hardware and Software Design

The Orion mission was designed to be the first on-orbit demonstration of precise, autonomous, formation flying using Carrier-Phase Differential GPS. Accomplishing this mission using a microsatellite requires a careful design of the hardware, mission plans and associated software. This chapter outlines the status of the Orion spacecraft and mission, as it stood in the summer of 2001, both from a hardware and software design standpoint. However, it should be noted that the Orion mission is still a work in progress and many aspects of the mission are currently undergoing modifications. For the latest developments on the Orion formation flying mission, please consult Ref. [33]. Throughout this chapter, remarks will be inserted to indicate how the latest modifications of the Orion mission may impact the analysis presented here. For the most part, however, much of the analysis is still applicable and only minor changes will be required. The single largest modification to the Orion mission since the writing of this chapter was the replacement of both Emerald spacecraft with a second identical Orion. As will be seen throughout this chapter, this modification primarily impacts the launch sequence and fuel usage.

2.1 Overview

A key step in precise formation flying is developing a sensor and associated estimation algorithms that can be used to accurately measure the **relative** positions and velocities of the vehicles in the fleet. GPS can be used to perform these relative navigation

measurements. In fact, using the Carrier-differential Phase GPS (CDGPS), the relative position and velocity measurements can be determined to within 2-5 cm and 1 cm/s in real-time[1]. One of the primary goals of the Orion mission is to demonstrate this technology on-orbit. Note that several spacecraft formation flying missions have already demonstrated the relative navigation capability using the *code-based* differential GPS. For example, a GPS receiver was installed on the ORFEUS-SPAS [34] satellite that was deployed from the Space Shuttle. A second GPS receiver was mounted on the Shuttle and raw GPS phase measurements were collected by these two receivers. In this mission, 10-50m relative positioning accuracy and meter/sec-level relative velocity accuracy were achieved by post-processing the data. Surrey Space Center [35] has also demonstrated GPS sensing for a formation flying experiment with two spacecraft, SNAP-1 and Tsinghua-1. These two spacecraft carried GPS receivers and demonstrated meter-level positioning capabilities using pseudoranging. However, these two spacecraft computed their absolute positions independently, and relative positioning capability using differential GPS was not demonstrated. NASDA (National Space development Agency of Japan) successfully performed the autonomous rendezvous and docking mission of the ETS-VII [36] using code-based differential GPS. They achieved relative position and velocity errors of less than 10m and 1cm/sec, respectively. However, the code-based differential GPS was only used in the coarse approach phase due to its relatively poor accuracy. A laser radar and a proximity image sensor were used in the final approach and docking phases. While these results are impressive, the goal of the Orion mission is to extend them by demonstrating the use of CDGPS as the sensing system for very precise relative navigation and formation flying control in *real-time*.

To accomplish this goal, we have developed, in conjunction with NASA GSFC, a custom GPS receiver that has been tested extensively in various ground testbeds (blimps, robots, racecars) and GPS simulators. This sensing data is then passed to a series of fleet coordination and control algorithms that determine where the vehicles should be located in the fleet and what control commands should be applied to perform the station-keeping. These control decisions will be performed autonomously onboard the Orion spacecraft. Furthermore, we have designed and built a three-vehicle fleet (one Orion and two Emerald spacecraft) to demonstrate formation flying

[1]These measurements can then be filtered over longer periods of time to obtain better estimates – see Ref. [16]

on-orbit. This mission was originally conceived in 1996 and has been described in several papers since that time [37, 38, 23, 39, 40].

The first part of this chapter gives a detailed account of the Orion-Emerald mission timeline as well as a description of the analysis to support the mission plan. The second part of the chapter describes some of the key hardware being used on the satellites.

2.2 Mission Description

The mission operations must be carefully planned to meet the goals for the Orion-Emerald mission within the power and fuel constraints. This section presents the mission operation plan, including the mission timeline, from launch to de-orbit. Special attention is given to the activity of the spacecraft through the mission life, so that resources (such as power and fuel) can be budgeted for the mission. These budgets are given in the next section.

Throughout this discussion, it is important to distinguish between *stages*, *modes*, and *cycles*. Stages refer to distinct segments of the timeline. Modes describe what the satellite is doing at any given time, and may be repeated several times during a stage, or occur during different stages. Cycles refer specifically to a series of modes that will be repeated in a set order. The following discusses the various spacecraft modes and stages of the Orion-Emerald formation flying mission.

2.2.1 Modes

Various modes of operation determine the activity and power usage of the Orion satellite on-orbit. A power usage number is given at the end of each section. These power values were measured on engineering models of the hardware to be flown, so they are known with a high degree of confidence. The modes of operation are:

- Cruise
- Communicate
- Active Control
- Computing
- Stabilizing

The **Cruise** mode will be employed primarily for recharging the batteries. Many subsystems will be turned off, or set in a low power state. This will allow the solar

cells to trickle charge the batteries. During this time, there will be minimal contact with the Emerald satellites. A single GPS receiver will remain active to provide only absolute state navigation. *Power budget: 340mA, 4130mW*

The **Communicate** mode involves the exchange of information with the ground, so it is only active during periods when there is ground station coverage (generally, this will be a brief interruption of another mode.) During this mode, commands can be sent to the formation and telemetry (vehicle and system status) can be downlinked. Also, during the experiment stage, raw data will be sent to the ground for post-processing. This is critical for validation and verification; performance measurement, and to determine mission success. *Power budget: 720mA, 8590mW*

The **Active Control** mode is entered any time the propulsion system is used for attitude or translational control. This includes large maneuvers and formation keeping. The full GPS suite is required during active control mode since proper thruster usage requires accurate attitude knowledge. Data will continue to be crosslinked between the Emeralds and Orion to allow relative navigation. The Science Computer is also required to perform the computations required for attitude determination, formation planning, position control, and thruster mapping. Significant power will be drawn during this mode for thruster activation. Since the active control mode is the most power intensive mode, ground communications mode will not be entered when operating in this mode. *Power budget: 2740 mA, 32800 mW*

Stabilization mode will occur when the satellite first powers up, or when it is reset for any reason. In this mode the satellite's primary objective is gaining GPS navigation fix, by tracking the required number of GPS signals. This mode will use the magnetometer and magnetic torque coils to de-tumble the satellite (as necessary), thereby enabling the acquisition of GPS signal lock. The ground team will ensure that the GPS fix is acquired before turning the attitude control over to the propulsion subsystem from the magnetic coil subsystem. *Power budget: 640 mA, 7630 mW*

During the **Computing** mode, the Science Computer will carry out most of its computationally intensive tasks. Minimal GPS navigation will be employed (no attitude), and there is no ground communication. During this mode, optimal trajectory planning will be performed as well as other computationally intensive processes such as data analysis. *Power budget: 560 mA, 6680 mW*

> **Remark:** In the new Orion mission, each spacecraft (both Orion's) will have a Science Computer onboard permitting distributed computations during the Computing mode. □

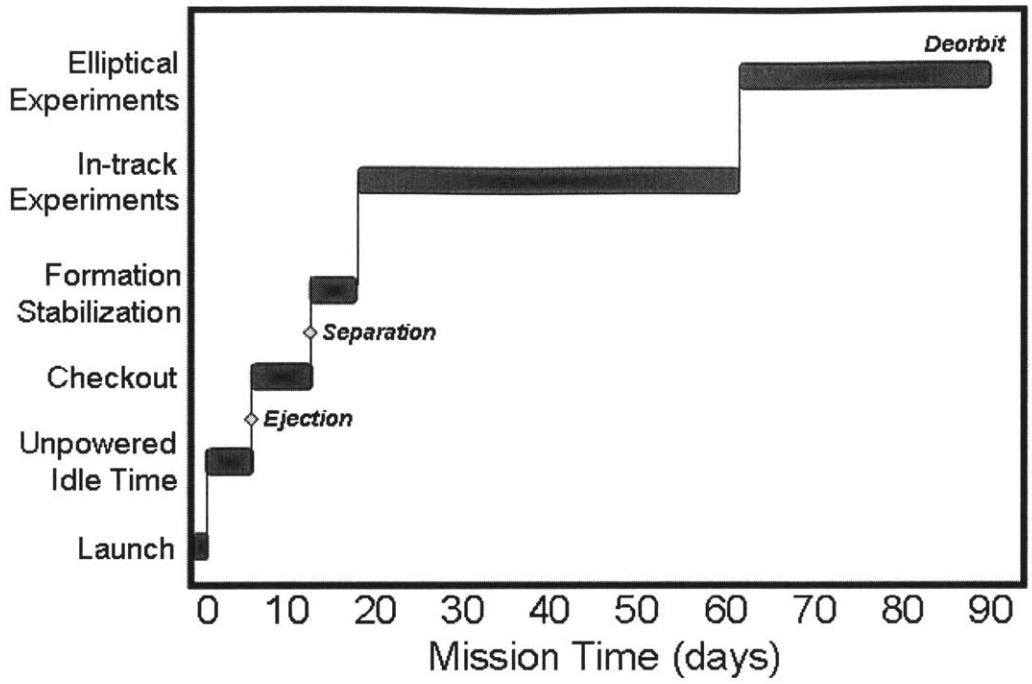


Fig. 2-1: Orion Mission Timeline

2.2.2 Stages

Of course, one of the first stages is the ground preparations which must take place prior to launch. After fabrication, integration, and testing, the integrated Orion-Emerald stack (called *Nanosat-1*) will be delivered to the launch site. A verification process will then be done to ensure that all safety mechanisms are functioning properly. In addition, the team will make final checks and adjustments on the vehicle state. This includes "topping-off" the battery charge, checking/calibrating nominal telemetry values, and downloading the final software revisions.

Figure 2-2 illustrates the launch sequence from the Space Shuttle and Figure 2-1 shows the expected mission timeline. The mission is predicted to last for 50-90 days after deployment from the Shuttle. During this period, there are seven operational stages. The stages are:

1. Launch
2. Ejection
3. Checkout
4. Formation Stabilization
5. In-Track Experiments

Fig. 2-2: Orion-Emerald Launch Sequence

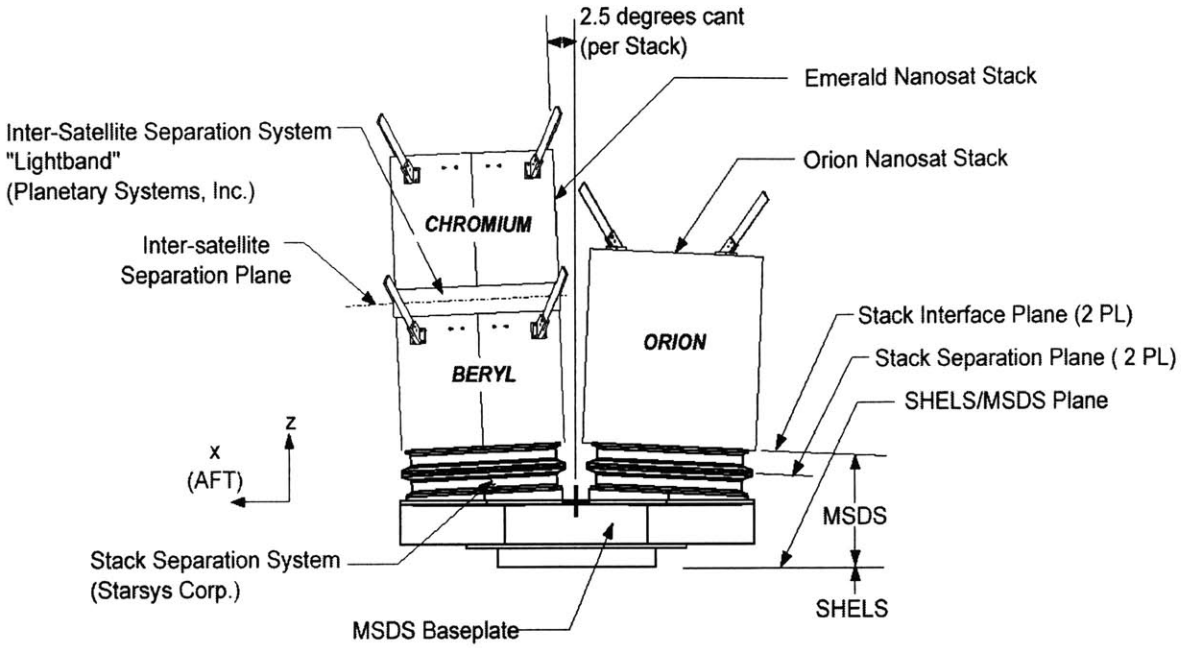


Fig. 2-3: Orion and Emerald on MSDS

6. Elliptical Experiments
7. De-Orbit

The first stage is **Launch**. The Orion satellite will be deployed from the Shuttle with the two Emerald satellites on the Multiple Satellite Deployment System (MSDS) platform, designed and constructed by the Air Force Research Laboratory. Figure 2-3 shows the launch configuration [41]. The entire MSDS-Emerald-Orion package (Nanosat-1) will be ejected from the Shuttle using the SHELS launcher in the payload bay. The target orbit is 325-350 km altitude and 28.5° inclination, with 0.005 eccentricity. However, 400 km and 50° are preferred parameters, since atmospheric drag severely limits mission lifetime at lower altitudes and higher inclinations increase ground contact visibility times. While these are design targets, the exact parameters will, of course, be determined by the Shuttle mission profile.

Extensive work has been done to meet all Shuttle safety requirements, and designing for the physical interface. Due to these requirements, Orion will be powered down while on the Shuttle. During all payload and launch vehicle processing, as well as the actual Shuttle launch, the satellites will remain inert. All circuit paths will be broken by a set of latching relays, controlled from the MSDS platform. They are pre-set to close 20 minutes after deployment from the Shuttle. A second set of inhibits block power from reaching Orion's propulsion system. These inhibits will be closed when

Orion and the Emeralds are released from the MSDS platform (approximately four days after Shuttle deployment). These precautions were developed to prevent any Shuttle re-contact hazards.

The second stage is **Ejection**. The MSDS shells shelf is ejected from the bay of the Shuttle, at a time determined by the Shuttle mission needs. When the shelf is ejected, the Emerald and Orion satellites will remain attached to the platform and to each other. This package will float for twenty minutes away from the Shuttle before a command is given by the MSDS platform to release the first set of inhibits. The satellite subsystems will then be powered on, while still attached to the platform (with the exception of the Orion propulsion subsystem.)

> **Remark:** Much of the Launch and Ejection stages will be changed to accommodate the new mission format. The latest plan as of this writing is to eject one Orion spacecraft from the Space Shuttle cargo bay at a time, separated in time by exactly one orbit of the Space Shuttle, thus removing the need for the MSDS platform. Studies are currently being performed to analyze the feasibility of inserting each Orion spacecraft into similar orbits in this manner without requiring excessive fuel to correct for ejection errors. While this change adds to the complexity of having to match orbit insertions, we gain mission flexibility since each vehicle will have thrusting capabilities (the Emerald spacecraft means of active control is limited to variable drag). □

The third stage is the **Checkout**. At this point the satellites are powered, so their performance can be assessed. Communication with the ground and with each other will be confirmed. Power will be collected by the solar panels, which will be used to charge the batteries. Basic telemetry will be downlinked to confirm that operations are normal. The GPS payload will be turned on for all vehicles, and initial ephemeris data for an aided warm start will be uploaded. While the vehicles are still together, a first navigation fix will be attempted. It is possible that the group will be tumbling uncontrollably at this stage, therefore preventing a GPS lock. In this case the ground crew can either attempt to use the attitude control systems on each vehicle to stabilize the group, or just delay the GPS checkout until the vehicles are separated and stabilized.

This stage will last at least four days, allowing the satellites to drift away from the Space Shuttle. The current mission sequence has the payload deployed at the end of

Table 2.1: Power Budget (all values in mW)

Subsystem	Cruise	Stabilize	Comm	Active	Compute
Payload	1450	1450	1450	6000	4000
CDH	700	700	700	7200	700
Comm	1430	1430	5890	2380	1430
Torquer Coils	250	3750	250	250	250
Propulsion	300	300	300	22400	300
Total	4130	7630	8590	32830	6680

the Shuttle mission. The four days will thus give the Shuttle sufficient time (including contingencies) to de-orbit and land while Nanosat-1 is effectively inert (NASA safety requirements.) After the four-day period, the second inhibit release command is given by the MSDS, and the Orion propulsion subsystem will be powered on.

The fourth stage is **Formation Stabilization**. At this point, the three vehicles are released from the MSDS mounting platform. Using the magnetometer and torquer coils, any initial tumble caused by ejection and release will be damped. Damping any rapid tumbling motion is required to ensure that the GPS sensor can lock on. During this stage, the Orion vehicle will be maneuvered into its first formation position (100 m in-track separation with the target Emerald vehicle.) Note that the Emerald vehicles will also attempt to perform formation flying using the CDGPS sensor and differential drag panels [42, 43]. As such, the modes of operation for the Emerald vehicles will be very similar to the ones described in this paper, but the time-scales will be much longer and the accuracy levels are expected to be much less precise. Future papers will discuss this aspect of the mission in more detail.

This stage will start with the computing and stabilization modes. For the translational maneuver, a short period of active control mode will also be entered. There will also be periods of communication and cruise modes.

The fifth stage is the **In-Track Experiments** stage which consists of various modes:

- Cruise mode to recharge batteries.
- Communicate mode (single uplink to receive "go ahead").
- Computing mode to plan future actions.
- Active control mode, which includes (i) determining the relative navigation estimate (position and attitude); (ii) commanding thrusters to maneuver to the desired formation; and (iii) actively controlling the formation. The active control mode will be continued for the desired length of time (fuel and power consumption will be closely monitored for anomalies).

- At the termination of the active control mode (either as a result of completing the desired experiment length or due to an anomaly), the vehicle returns to cruise mode. It will then alternate in and out of cruise and communication mode, passing the collected data to the ground (when visible), or just coasting otherwise. This period may also require extensive computing to analyze the performance and change planning parameters based on that performance.

The initial objective will be to demonstrate the control of an in-track formation. There can be a tight, or precise formation, where the Orion remains within a 5m×10m×5m *error box* centered at a point 100 m (in-track) from one of the Emerald satellites. A much coarser formation (error box 25m×50m×25m) can also be used during this stage to conserve fuel.

When this first experiment has been successfully repeated several times, a second series of experiments will be performed. In this case Orion will send positioning commands to the Emerald vehicle(s), and using their drag panels, the Emeralds will perform station-keeping maneuvers in cooperation with Orion. This will be repeated until all mission objectives during that stage are completed, or until all budgeted consumable resources are exhausted.

The sixth stage is the **Elliptical Formation Experiments**. This stage will repeat the same experiment cycles as the previous stage, but will demonstrate different formations. These formations are called elliptical because of the shape of the relative motion between the satellites in a local-vertical local-horizontal frame (radial and in-track) attached to the reference vehicle. If sufficient fuel remains, we will also include a slight cross-track component of this relative motion. For operational description purposes, this stage will be the same as the In-Track Experiment stage. Note that this last experimental stage is optional and will be performed to the extent that time (de-orbit) and resources (fuel) allow. However, it could last as long as 60 days if performed at a low intensity level.

Remark: Note that with the updated mission format, each of the Orion vehicles will have an approximate ΔV of 25 m/s, which effectively doubles the mission life-time. The current operations plan for the Orion mission will be to run the experiments as described above using one vehicle as the "active control vehicle" and the other as the "passive drifting vehicle". The vehicles will swap roles to distribute the fuel cost. Fully coordinated maneuvers using two active vehicles will also be performed. Also, note that the 4 day period required by the Shuttle safety team to give the Shuttle time to land may have a significant impact on the new mission profile. Our hope is that with an improved design and more safety analyses of the power and propulsion inhibits, the waiting period can be reduced to limit the drift between each Orion vehicle. □

The final stage is **De-orbit** which concludes the mission. Once the experiments are complete, and consumables have been nearly exhausted (with just enough left for de-orbit), the satellite orbits will be decayed to the point where they will re-enter the atmosphere and be entirely destroyed. Due to the small size of the vehicles, no extra precautions are required to ensure debris safety.

2.2.3 Resource Budgets

As with most missions, system resources require careful management and planning. Power, though renewable through the solar cells, still limits the amount of activity on Orion. Fuel, which is not renewable, obviously limits the total mission life. Another important resource is communication bandwidth, and so attention is also given to data transfer budgets.

Power: The power requirements drive the design, frequency, and duration of the modes of operation. Table 2.1 summarizes the predicted power numbers during the various modes of operation for each satellite subsystem. These numbers reflect the power draw as seen by the batteries. The numbers have been determined by testing engineering models of the actual hardware to be used. Many of the components utilized in the active control mode (GPS electronics, flight computers, and propulsion valves) have high power requirements. The power subsystem was carefully designed to be flexible over a wide range of power levels. The required quality of the solar cells was determined based on these estimated needs, balanced against the spacecraft's available surface area. The key point of this design analysis was that the energy

drawn from the power system while in eclipse is the minimum amount that must be returned during "in the sun" operations.

The average power from the solar cells is 2000mA per panel in the sunlight, which generates a time average of 18.6W to the power bus. The batteries have a 10A-h capacity at 12V. Battery power is required not only for periods when the spacecraft is eclipsed, but also during active control operations when the power from the solar cells cannot power all required subsystems. The battery capacity directly controls the length of any given active control period; the rate of battery recharge directly controls the highest frequency (or quickest cycle time) between active control modes.

A total mission simulation package was created in MATLAB (by B. Engberg [9]), and it allows for numerous mission parameters to be adjusted. It was used to assess the expected performance of the power system in the various operational flight modes. This simulation accounts for vehicle dynamics, downlink opportunities, and mission resources (fuel, battery capacity, memory). In addition, a scheduling feature allows a wide variety of mission profiles to be assessed.

Figures 2-4 and 2-5 show a sample of the output from the simulations. Figure 2-4 shows the available battery capacity over a 36-hour period of mission operations, which includes a demonstration of all flight modes. Note that an active control mode experiment begins on orbit 11 and continues until orbit 15, which supports the desire to safely conduct experiments of this length. The "toothing" in the chart shows how power is drawn from the batteries for about one third of each orbit of cruise mode, when the satellite is in the Earth's shadow. However, this power is easily returned while the satellite is in the sunlight; hence, the power system should be suitable for the designed operations. Figure 2-5 shows the current input from the panels. Note that slow satellite rotations result in peaks and troughs, because the sides of the satellite have more solar cell strings than the top/bottom. However, the average value is around 2000 mA, which is the single-panel target value. Note that Figure 2-5 shows a 3 orbit period.

Fuel: Fuel is Orion's most critical non-renewable resource. Orion has a predicted total $\Delta V = 25$ m/sec. During the mission there will be a number of discrete, large maneuvers, as well as periods of active control mode (coarse and precise).

A detailed fuel resource study can be found in Ref. [9]. By examining the spacecraft resource budgets, it became clear that a careful mission operations plan is required. Maintaining an actively controlled formation uses much less fuel than letting

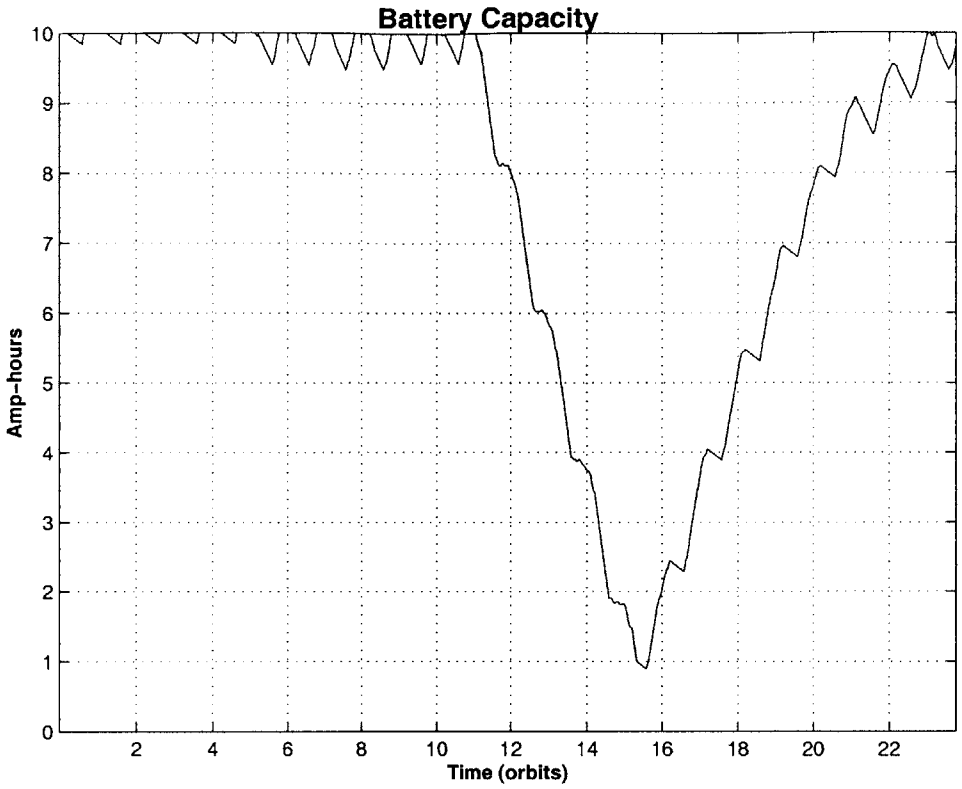


Figure 2-4: Battery Capacity

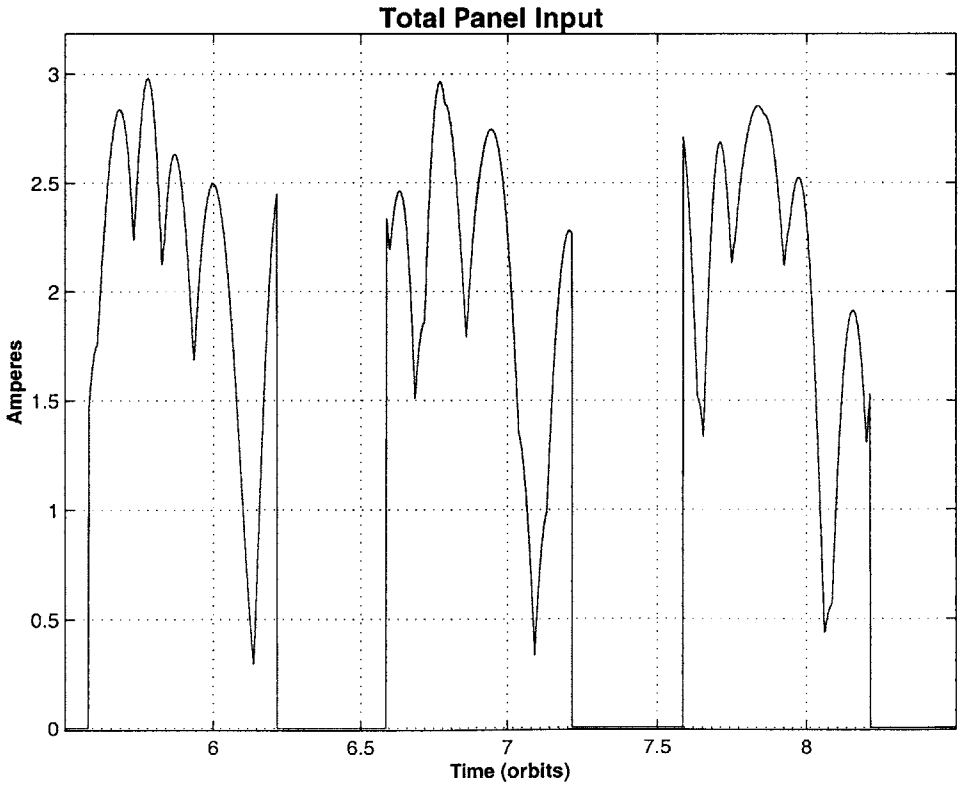


Figure 2-5: Current Input

the vehicles drift apart and then returning the satellites to formation. However, there is not enough power to constantly remain in active control mode. Therefore, the plan is to stay in active control mode as long as possible, which is on the order of 4 orbits. It then requires about 8 orbits to recharge the batteries to full capacity. Assuming 4 orbits of precision flying and attitude control, using ≈24 mm/s each, one active cycle uses ≈100 mm/s. Then, assuming an 8 orbit drift and a 2 orbit repositioning maneuver, which takes ≈250 mm/s, a total of ≈350 mm/s of fuel is used per experiment cycle. Based on the current control implementation, this cycle of control, drift, and maneuver could be repeated ≈55 times. At 14 orbits per cycle, and 16 orbits per day, the mission should be able to last for ≈48 days. Clearly differential drag is a key concern for this mission, and we are investigating ways to reduces its impact.

> **Remark:** The new mission format will contain two identical vehicles, which will greatly reduce the fuel expense due to differential drag. However, additional fuel will be required to correct any orbit insertion errors that may have occurred during the initial ejection stage. □

Communication Bandwidth: Ref. [9] contains a communications bandwidth analysis for the Orion-Emerald mission. Communication bandwidth is an important resource to manage, particularly in the downlink of data to the ground stations. Data is collected during the formation flying experiments which must be transmitted to the ground to assess mission performance. Downlink capacity must account for both mission data and telemetry. However, ground station coverage for this mission is predicted to be very limited and the communication data rate is only 9600 baud.[2] Considering a 28.5°, 325 km altitude orbit, there is only 25 minutes of contact time per day (on average, with each overhead pass lasting approximately 6 minutes). This gives an expected 1500 sec/day of data downlink time. Realistically predicting a downlink rate of 2 Kbits/sec (reduction from 9600 baud accounts for overhead, signal-loss and acquisition time), Orion expects only 3000 Kbits/day (or 375 Kbytes/day). Though the Emeralds will have the same downlink capability as Orion, they will be running other experiments as well as formation flying, and have dedicated the majority of their bandwidth to those other experiments.

Given the analysis in Ref. [9], Orion can downlink about 411 samples of GPS data worth of data per day, which places a very tight constraint on the mission. To enhance

[2]This design was based on communication robustness considering the power levels and transmitter/receiver capability.

Table 2.2: Orion Mass budget

Subsystem	Mass (g)
GPS Payload	1072
Structure	12403
CDH	900
Comm	696
Torquer Coils	2473
Propulsion System	10648
Power System	7140
Total	35332

mission verification, other ground-based approaches will also be used. For just coarse verification, NORAD orbital parameters for Orion and Emerald can be obtained. However, Orion is attempting to demonstrate finer control than the precision of the NORAD orbital elements. By taking concurrent GPS measurements with ground based GPS receivers, the GPS constellation ephemeris can be obtained. This data can then be combined with the data collected on Orion to accurately determine the absolute positions of each vehicle using post-processing techniques (similar to GIPSY-OASIS). This should enable millimeter-level verification.

Remark: Depending on the capabilities of the ground station, it may be possible to downlink data from both Orion vehicles at the same time in the new mission format, thus doubling the downlink capability. While this download capability is sufficient to analyze the mission to determine if the goals have been achieved, help will be solicited from additional ground stations to extract as much data as possible from the Orion vehicles. □

2.3 The Orion Spacecraft

The following subsections discuss each of the primary subsystems of the Orion spacecraft. Emerald spacecraft are similar in the key subsystems associated with the formation flying experiment. The primary difference between the two spacecraft is that the Emeralds do not have a propulsion system, but rely on differential drag to change their relative positions.

2.3.1 Structure

The Orion structure is a 44.5 cm (17.5 in) cube composed of T-6061 aluminum honeycomb. The main load-bearing portion of the microsat consists of a top faceplate, a bottom faceplate, and a set of panels that form an internal "X." These honeycomb plates are each 1.27 cm (0.5") thick and are bound together with aluminum L-brackets and stainless steel bolts. Four 0.64 cm (0.25") honeycomb plates cover the remaining four sides of the cube. These panels are non-load-bearing. The solar cells will be bonded to a Kapton-insulated facesheet, which will then bond to the outside panels. Orion's total mass is approximately 35 kg (see Table 2.2).[3] Recent vibration tests of an engineering model (EM) of the structure have shown that its natural frequency is 119 Hz. The EM structure has also passed static load tests of 33 g's (along the diagonal) in a centrifuge. Figure 2-6 shows some of the propulsion systems of the Orion spacecraft and Figure 2-7 shows the CPU and some of the GPS payload.

2.3.2 Propulsion

The Orion propulsion system uses GN2 stored in three composite wrapped aluminum tanks. The system is design to provide the satellite with the maximum ΔV for the given volume and mass budgets. Most of the parts used are COTS parts in order to simplify the manufacturing process. There are 12 cold gas thrusters clustered in four groups of three to provide full 3-axis attitude and station-keeping control. Each thruster has the capability of providing approximately 60 mN of thrust. The 3 cylindrical fuel tanks, each pressurized to 3500 psi are predicted to provide a total ΔV capability of approximately 25 m/s. The EM propulsion system has been extensively tested to demonstrate that it will meet the Shuttle safety (*e.g.,* verification procedure to show valve closure, and tank certification) and mission performance goals (leak tests of the high and low pressure sides).

2.3.3 Position and Attitude Determination System

The position and attitude determination system for Orion is comprised of two parts: A magnetometer (for coarse attitude determination) and a GPS receiver system (with the capability of determining attitude as well as position). Orion's magnetometer is a

[3]These figures were obtained from actual mass measurements of engineering model hardware.

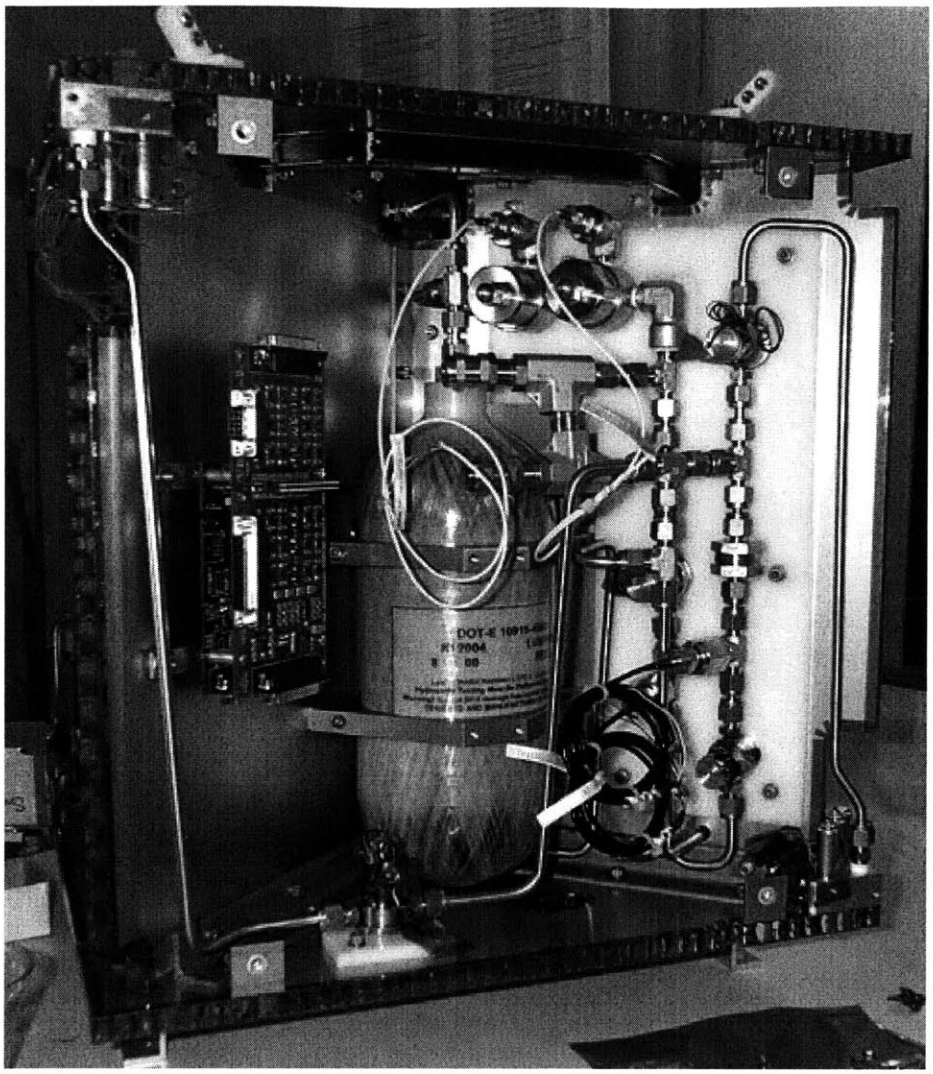


Figure 2-6: Orion spacecraft interior showing some of the propulsion plumbing and control electronics.

Honeywell HMC2003 3-axis magnetometer with 40 μG resolution. This magnetometer will be used early in the mission as feedback for the torquer coils during detumbling.

The GPS receiver for the Orion spacecraft is a modified version of Mitel Semiconductor's Orion^{TM} GP2000 chipset [3, 38, 23, 11]. The receiver operates an ARM60 32-bit processor with a GP2021 Correlator with 12 channels tracking the L_1 band carrier signals. To simplify the attitude determination process, a second RF front end was added to each board. Each RF front end can be programmed to use any of the 12 available correlator channels. Figure 2-8 shows a photo of the modified Orion GPS receiver. There are total of 6 GPS antennas (3 boards) on the Orion vehicle and 2 antennas (1 board) on the Emeralds.

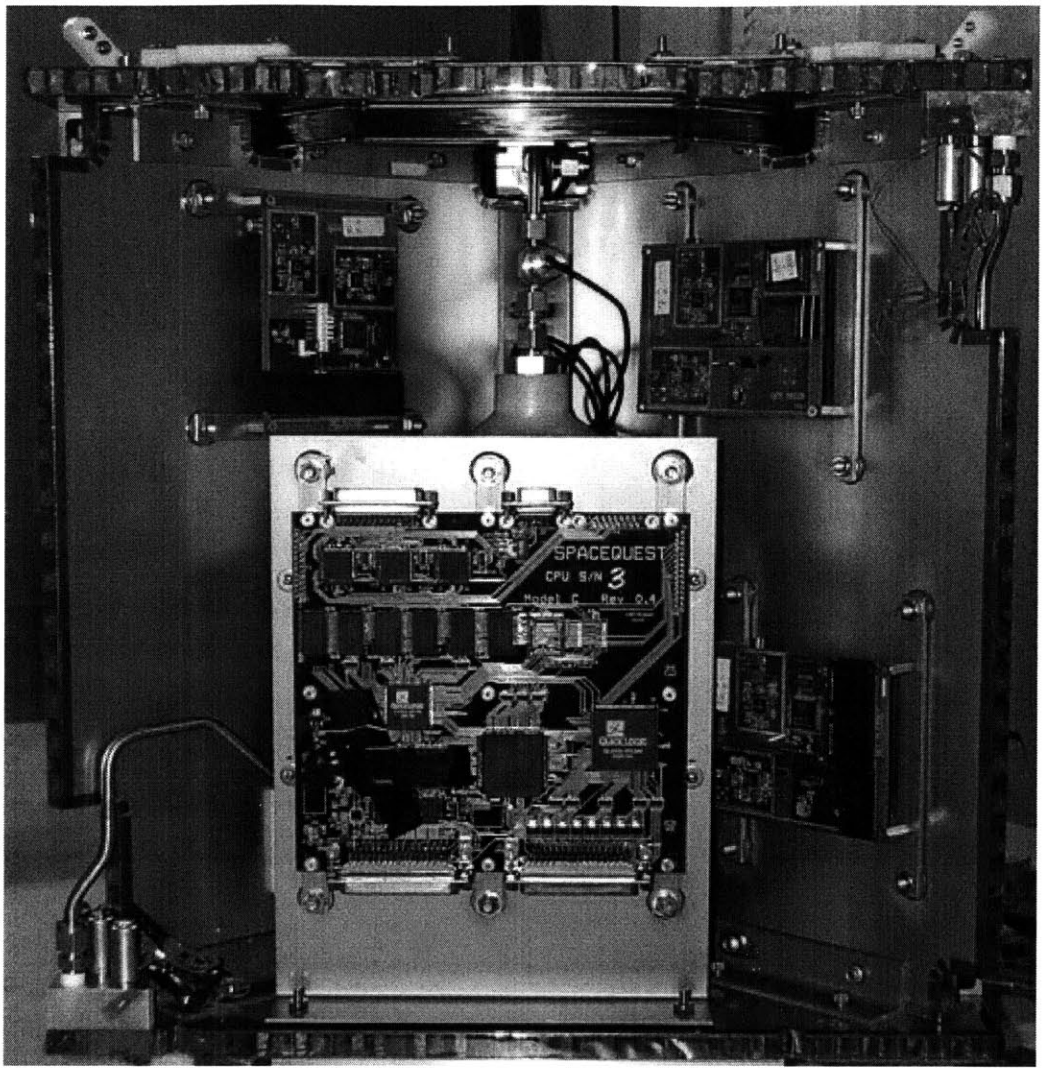


Figure 2-7: Orion internal structure showing CDH CPU (middle bottom), GPS (top left, right and bottom right), HP prop (middle back) and LP Prop (lower left and upper right).

Differential Carrier phase measurements should provide very precise relative position estimates between the Emerald and Orion satellites (expected errors on the order of 2-5 cm depending on the geometry). Velocity estimates on Orion will employ Doppler measurements used in the GPS phase lock loop. Doppler measurements can be used as an additional measurement for the estimator rather than a state which is purely an estimated value.

Orion will employ three Trimble Miniature OEM Antennas on the top face, a single antenna on the bottom, and two on opposite sides. Once the satellite detumbles, the "top face" should nominally point towards the NAVSTAR constellation

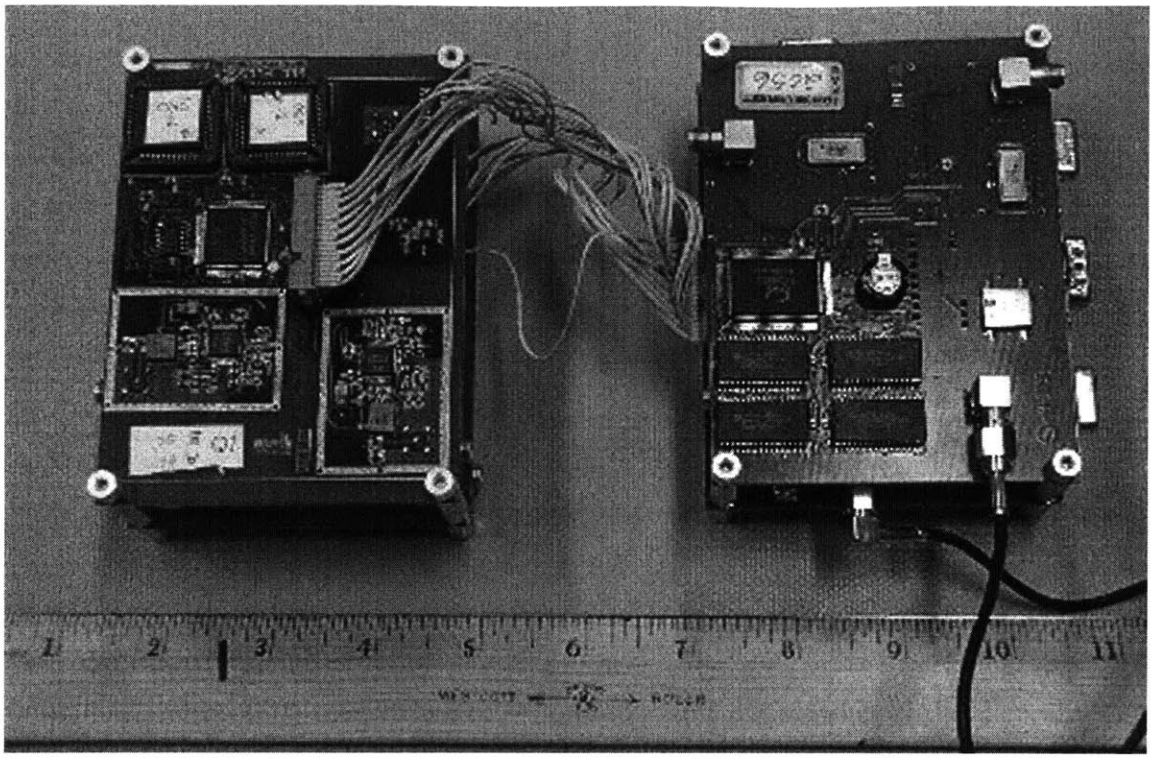


Figure 2-8: Orion GPS Receiver Stack.

and the remaining antennas will provide GPS visibility during maneuvers. The three on the top face form a triangle, which should provide a three dimensional orientation solution. The remaining three antennas were placed to improve the average sky coverage and attitude dilution of precision "ADOP", assuming that a *non-aligned antenna array* is used for attitude determination [44, 45]. These attitude and relative position solutions will be used in the formation and attitude control algorithms.

2.3.4 Attitude Control System

The Orion Attitude Control System (ACS) consists of two distinct subsystems. A magnetic damping controller is included to slow the spacecraft rotation sufficiently for GPS signal to be acquired. This damping might be required at the start of the mission, in the event of loss of GPS, or at the start of a GPS experiment after a period in power-down mode. Dedicated hardware, consisting of a three-axis magnetometer and torquer coils, allows the detumbling to be performed without GPS information. The torquer coils can be seen at the top of Figure 2-7. The control law $\vec{\mathbf{M}} = -k\dot{\vec{\mathbf{B}}}$ guarantees energy damping [46] where $\vec{\mathbf{M}}$ is the moment vector, $\dot{\vec{\mathbf{B}}}$ is the time rate of change of the magnetic field vector calculated in the body frame of the spacecraft,

and k is the gain.

The second attitude controller uses the GPS attitude solution and the thrusters. It is designed to keep the top face (with three GPS antennas) pointing "up" for the best GPS sky coverage. A Kalman filter is used to estimate the full attitude state (including rates.) These estimates are compared with a reference state generated from the approximate absolute position knowledge to find the error for each axis. Thruster switching rules are then applied to stabilize about the reference attitude. The current controller uses $\Delta V \approx 4\,\text{mm/s}$ per orbit to maintain pointing within $\pm 45°$. Both systems have been extensively tested using full nonlinear attitude models.

2.3.5 Command and Data Handling CPU

The Command and Data Handling (C&DH) system on Orion is responsible for all low-level tasks onboard the spacecraft. These tasks include decoding ground and inter-satellite communication, forwarding commands to distributed subsystems, controlling power switching for subsystems and experiments and gathering health and telemetry data. The CPU will be a SpaceQuest NEC V53 with a 10MHz processor and 1MB of EDAC Ram. The CPU will be running the Space Craft Operating System (SCOS) by BekTek. Both the CPU hardware and operating system have spaceflight heritage and are known to perform well. Figure 2-7 shows the C&DH integrated into Orion.

2.3.6 Communications

The communications subsystem on Orion is responsible for handing all data transfers between the ground and Orion as well as between Orion and either of the two Emerald spacecraft in the fleet. Crosslink between spacecraft will operate in half-duplex while the down/uplink will operate in full-duplex mode. Both crosslink and up/downlink communications will be conducted at 9600 baud. The modems for Orion are manufactured by SpaceQuest. These modems have been used successfully on past spaceflights. The communications system makes use of an omni directional antenna pattern with circular polarization.

2.3.7 Data Bus

The data bus for Orion is the COTS standard by Dallas Semiconductor. The I2C data bus provides a 100 kbps signal rate with multi-master arbitration. This bus

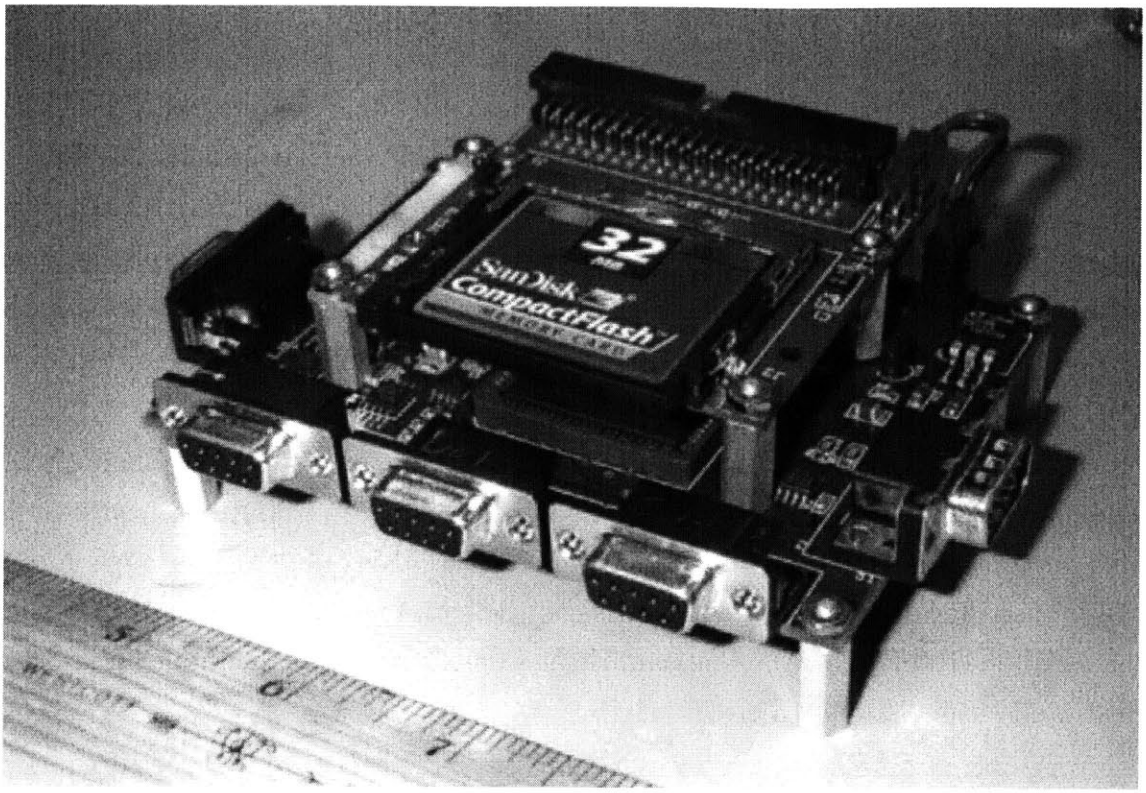


Figure 2-9: Orion Science Computer

will be used to send commands and data to all subsystems on the spacecraft. Most subsystems on Orion will use a PIC board as a bus controller. A bus monitor is also included on the Orion data bus to ensure that all bus activity is operating nominally.

2.3.8 Power Subsystem

Orion will make use of solar energy to charge batteries for power for all subsystems. The power subsystem consists of 6 body-mounted gallium arsenide solar cells (20.6% average efficiency), one 10-cell battery, power safety inhibits as well as electronics for power regulation, radiation latch-up protection, digital power switching, and voltage, temperature and current monitoring. Orion has 15 strings of solar cells with 8 cells per string. Each string generates 9.3 W on average. The storage cells are Sanyo CADNICA 5 AH KR-5000DEK cells. A battery box has been designed and built that meets NASA's stringent safety requirements. The mass of the entire battery box is 1.1 kg. The family of cells selected for Orion have spaceflight heritage on manned missions.

2.3.9 Science Computer and GPS Interface

The Orion satellite will lead most of the formation flying experiments in the Orion-Emerald mission. As such, it will carry out most of the navigation computation and fleet coordination. Note that the GPS receivers each contain an ARM60 microprocessor, but these are too limited to be used to perform sophisticated planning and navigation algorithms. Thus another computer (known as the *Science Computer*) was added to the Orion spacecraft that is dedicated to science objectives.

The Science Computer (SC) is a 200 MHz StrongARM 1110 RISC based microprocessor called the "nanoEngine" (designed and built by Brightstar Engineering.) The nanoEngine has three RS-232 communication ports and 20 general purpose IO pins for controlling a wide range of hardware. A 96 MB CompactFlash memory disk will be used for data storage. At full usage, the entire nanoEngine board draws less than 2W ($\approx$1700 MIPS/W). The nanoEngine weighs only 76 g (without an interface card) and is smaller than a credit card.

A significant amount of work was required to integrate the GPS receiver stack to the SC to create a unified payload for the Orion mission. The Orion GPS receivers (built by Mitel/Plessey/Zarlink) communicate data via simple RS-232 serial ports. To enable the SC to carry out the relative navigation algorithms, the raw GPS data from each GPS receiver must be transmitted to the SC at a frequency of 1 Hz (to permit timely relative navigation updates). Since the nanoEngine has only 3 serial ports available and two are already taken up by the C&DH and the debug port, there is only one remaining port to communicate with the GPS receiver stack. Thus, all three GPS receivers needed to share one serial port on the nanoEngine.

To permit this type of communication, a "token-bus" architecture was developed. A token bus permits multiple nodes to share a common communication link by tightly controlling which entity has authority to use the communication link. When a node has authority to communicate over the common communication link, that particular node is said to have the "token", hence the name token-bus. A token bus is similar to a token ring with the exception being that in a token ring, each node is limited to communicating only with the next node in line and not as a broadcast to all nodes, as in the case of a token bus. In the case of the Orion Payload, 4 nodes share the token bus (one Science Computer and 3 GPS receivers).

Electrically, the token bus is controlled by simple TTL logic gates. Figure 2-10 illustrates the required electronics to permit RS-232 communication link sharing.

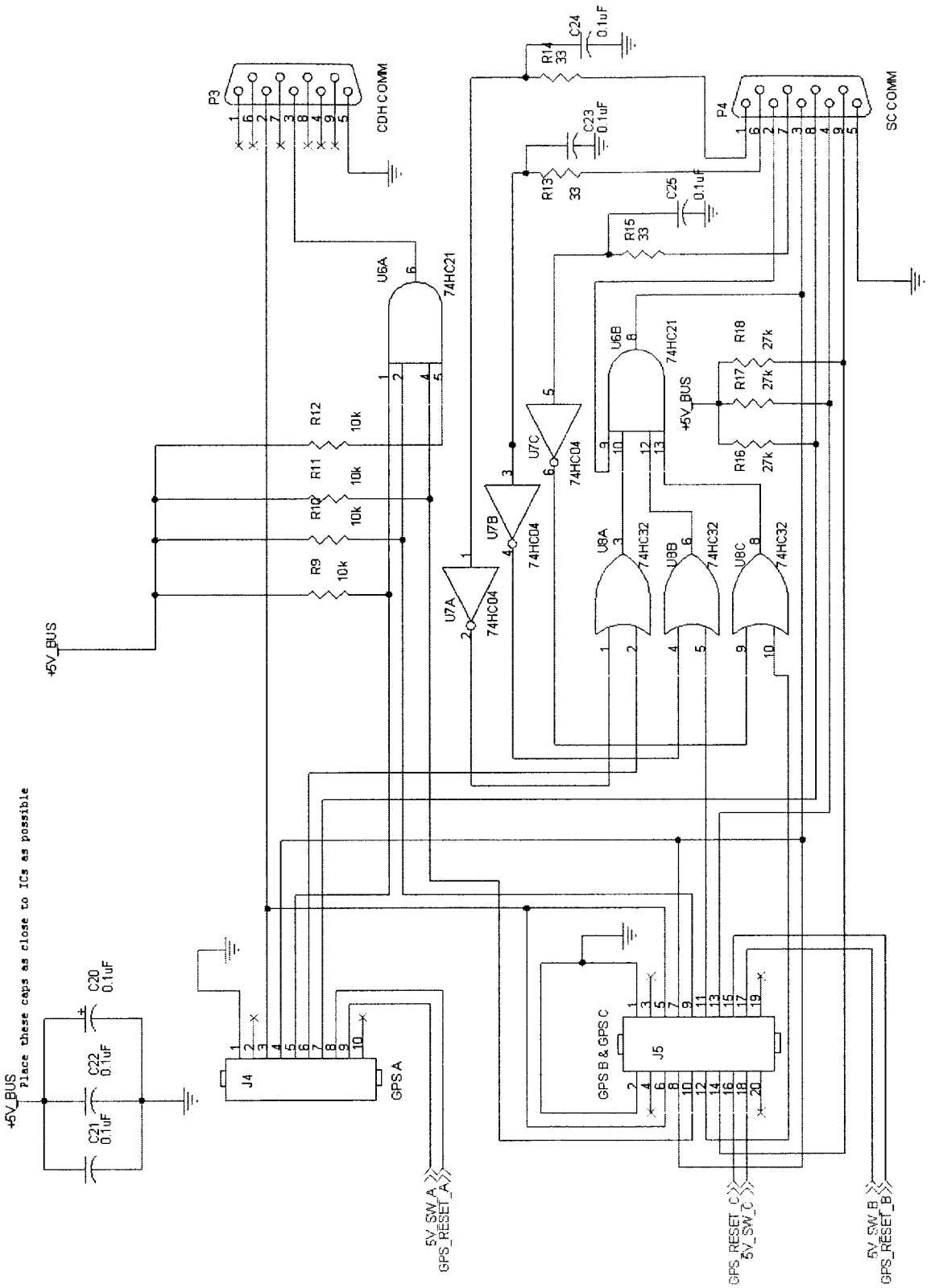


Figure 2-10: Token Bus Electronic Schematics

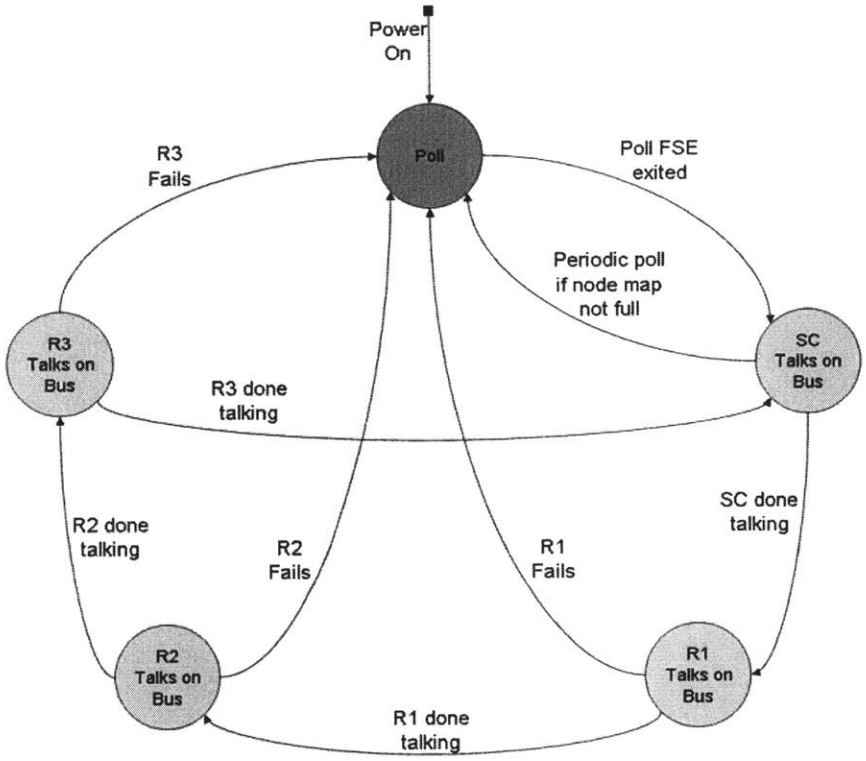


Figure 2-11: Token Bus Finite State Machine

Maxim 232A chips were required to translate the RS232 level signals (±10 V) down to TTL level signals (0-5 V). Once at TTL, regular AND, OR and NOT logic gates are used to multiplex all signals onto one line as well as to provide enable/disable capability from the Science Computer. Note that while the GPS receivers may be enabled or disabled to communicate over the token bus, every entity can always listen to data on the token bus.

From a software perspective, the token bus architecture describes a strict order in which each receiver (or the Science Computer) can communicate at a time. Figures 2-11 and 2-12 show the finite-state machine that describes the operation of the Orion Payload Token Bus. Upon startup, the Science Computer disables each receiver from being able to access the token bus by asserting a zero voltage on the GPIO pins associated with the enable/disable lines. Before activation, the GPS receivers must not be permitted to access the token bus because power-up transients on the RS232 port have the potential to confuse the other entities listening to the token bus. The SC begins the initialization process by polling the first GPS receiver. Immediately

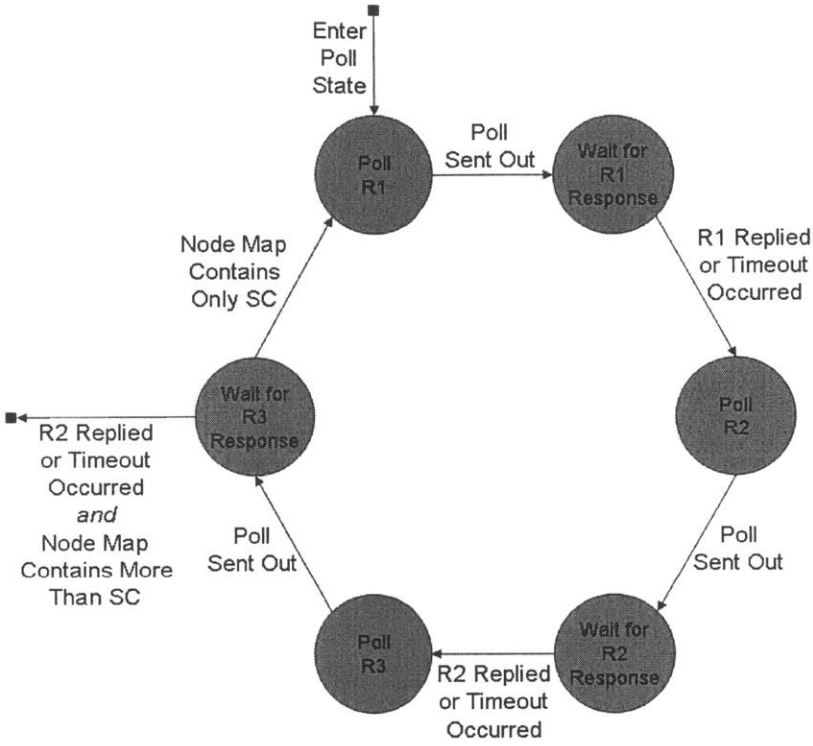


Figure 2-12: Polling Finite State Machine

after the poll is sent out, the SC enables the first receiver and waits for a response. If a response is received, the SC polls the next receiver in line and so on until all receivers have been polled and responded. If a receiver fails to respond over a specified time interval, the SC assumes that the receiver is in a failed state, disables the receiver and polls the next one in line. Whenever a receiver responds to a poll, every node on the token bus hears it and adds that receiver to the node's "node map". A node map informs each node in the token bus which nodes are active.

After the third receiver has been polled and either responded or determined failed, the SC enables all active nodes (not including the failed ones) and sends out a message on the token bus indicating that normal communications may commence. Embedded in this message is the next node that is permitted to communicate over the token bus. This next node is determined by the locally maintained node map. If the SC had multiple commands or data to send to the GPS receivers, the SC would repeatedly set the next node to be itself until it had communicated all of the data it needed to. When the first active receiver gets the token, it treats it in the same manner as the

SC did by setting the next node to itself until all data has been transmitted.

Periodically, the SC checks to see if the node map is full (*i.e.,* contains the SC and all three receivers). If it is full, normal communications simply continue. If the node map is not full, it indicates that one or more of the receivers has failed, possibly temporarily. To attempt to bring the failed receiver back online, the SC waits for the token, disables all nodes and repeats the initialization process. If the receiver failure was due to a temporary glitch causing a re-start of the receiver, this procedure will bring the once failed receiver back into normal operations on the token bus. The initialization process is also repeated in the event of the token bus being silent for a specified period of time, as this could also be an indication of a failed receiver.

The token bus control software was demonstrated to reliably communicate with all 3 GPS receivers simultaneously. Long tests were run for several days at a time to verify the robustness of this system. Furthermore, several tests were conducted that simulated a failure on one or more of the GPS receivers. The Science Computer software was able to diagnose the failure, remove the particular receiver from the token bus and continue communications with the remaining functioning receivers. Once the failures were cleared, the Science Computer was able to detect the correction and resume communication with the previously failed receivers.

2.3.10 Software

The operating system chosen for the Science Computer is Arm Linux (a form of Embedded Linux). The Linux operating system was chosen over a true real-time operating system because most computations required on Orion do not have hard real-time deadlines. Most tasks that will run on the Science Computer will have a window within which it is desirable that the task be completed but, with the possible exception of the GPS navigation solution, it is not critical that the tasks *always* complete on time.

While being a totally cost-free operating system, embedded Linux also provides many high- and low-level commands and structures that greatly simplify the coding process. File I/O, serial port reading and writing and thread management are just a few of the many utilities available to the Orion software provided by Linux. Through the use of multi-threading, longer, more computationally intensive tasks (such as formation planning) can be carried out in the background while shorter, more time critical tasks (such as communication tasks) can be completed in the foreground.

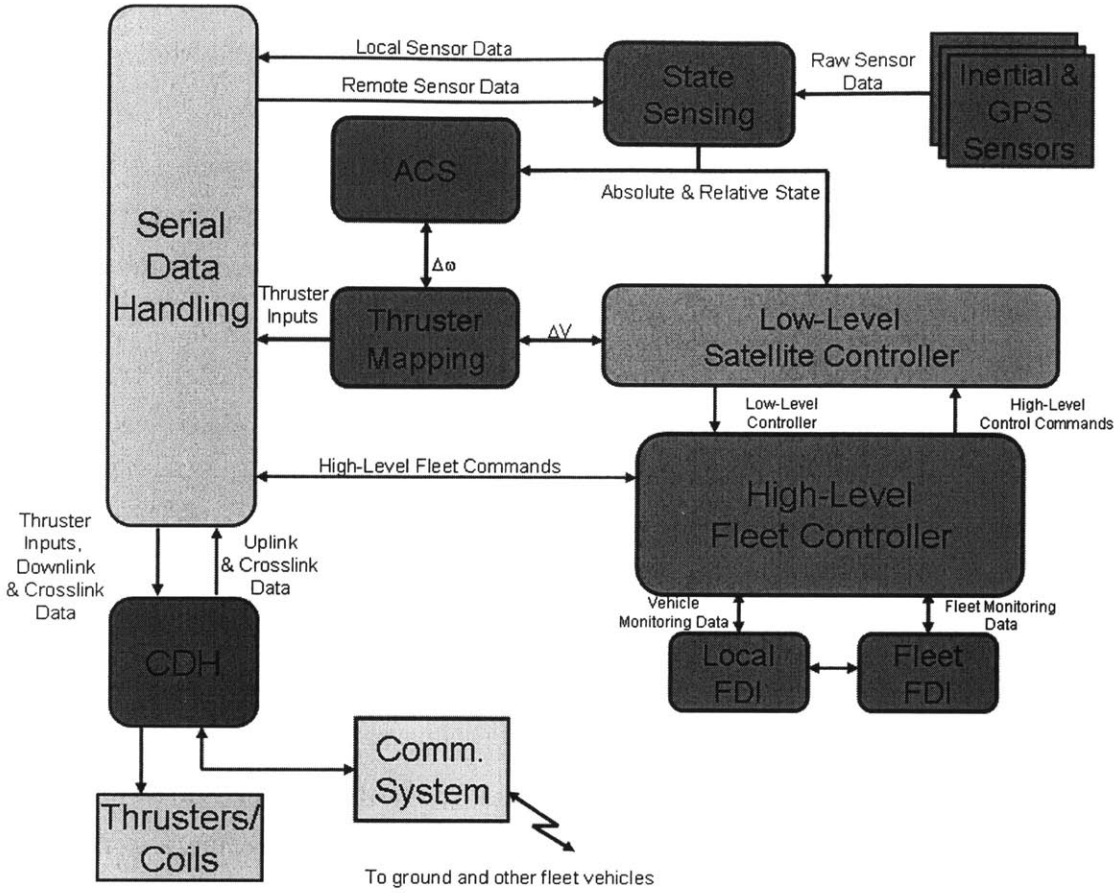


Figure 2-13: Orion Software Architecture

Inter-thread communication is done through mutex-protected global memory. The POSIX standard mutex structure of Linux is used to ensure that only one thread accesses a particular variable at any time, while preventing deadlock (deadlock is a situation where two threads wait forever on each other to stop accessing some variable).

Figure 2-13 illustrates the different software modules that make up the Orion software subsystem. Each module runs in its own thread(s) and has a set of well-defined inputs, outputs and update rate. The following briefly describes each module:

- **Task Dispatcher** - Handles all threads and mutexes. The task dispatcher spawns all system threads and wakes them up at the appropriate update rates (thus controls the sequencing of events on Orion).
- **State Sensing** - Uses 6 GPS antennas & 3 receivers to determine relative/absolute

position and velocity & spacecraft attitude using Carrier-Phase Differential GPS (CDGPS).

- **Attitude Control System** - Controls the spacecraft attitude to enable adequate GPS satellite visibility and ground communications.
- **Thruster Mapping** - Oversees the thrust commands sent to the cold-gas thrusters on the spacecraft. It feeds back the actual response given the requested ΔV and uses that data to estimate the actual performance of the thrusters on orbit.
- **Low-Level Satellite Controller** - Computes optimal control trajectories as commanded by the High-Level Fleet Controller. Also, applies thrust commands based on these detailed plans approved by the High-Level Fleet Controller. If disturbances cause the spacecraft to deviate from its plan, it is the responsibility of the Low-Level Satellite Controller to replan its trajectory and continue with the maneuver.
- **High-Level Fleet Controller** - Runs linear programming planners for coordination to reconfigure the fleet for different experiment runs.
- **Local and Fleet Fault Detection and Isolation (FDI)** - Monitors the thrusters, fleet state, vehicle states and other health parameters to determine spacecraft faults.
- **Serial Data Handling** - Queues up, sends, receives and interprets serial data from the debug port, C&DH and all three GPS receivers via the token bus.

As part of this research, the task dispatcher was demonstrated to successfully spawn each process at the correct time while passing it the required data. This demonstrated that the chosen architecture for the Science Computer can handle the onboard operations for the Orion formation flying mission.

2.3.11 Orion Payload Architecture

The Orion payload architecture is shown in Figure 2-14. For normal operations, raw GPS data will be sent to the Science Computer from the GPS receiver cluster for processing the navigation solutions. With this information, the Science Computer will run attitude control code and planning computations to determine optimum trajectories and formations. All hardware commands (*i.e.*, commands to the thrusters or

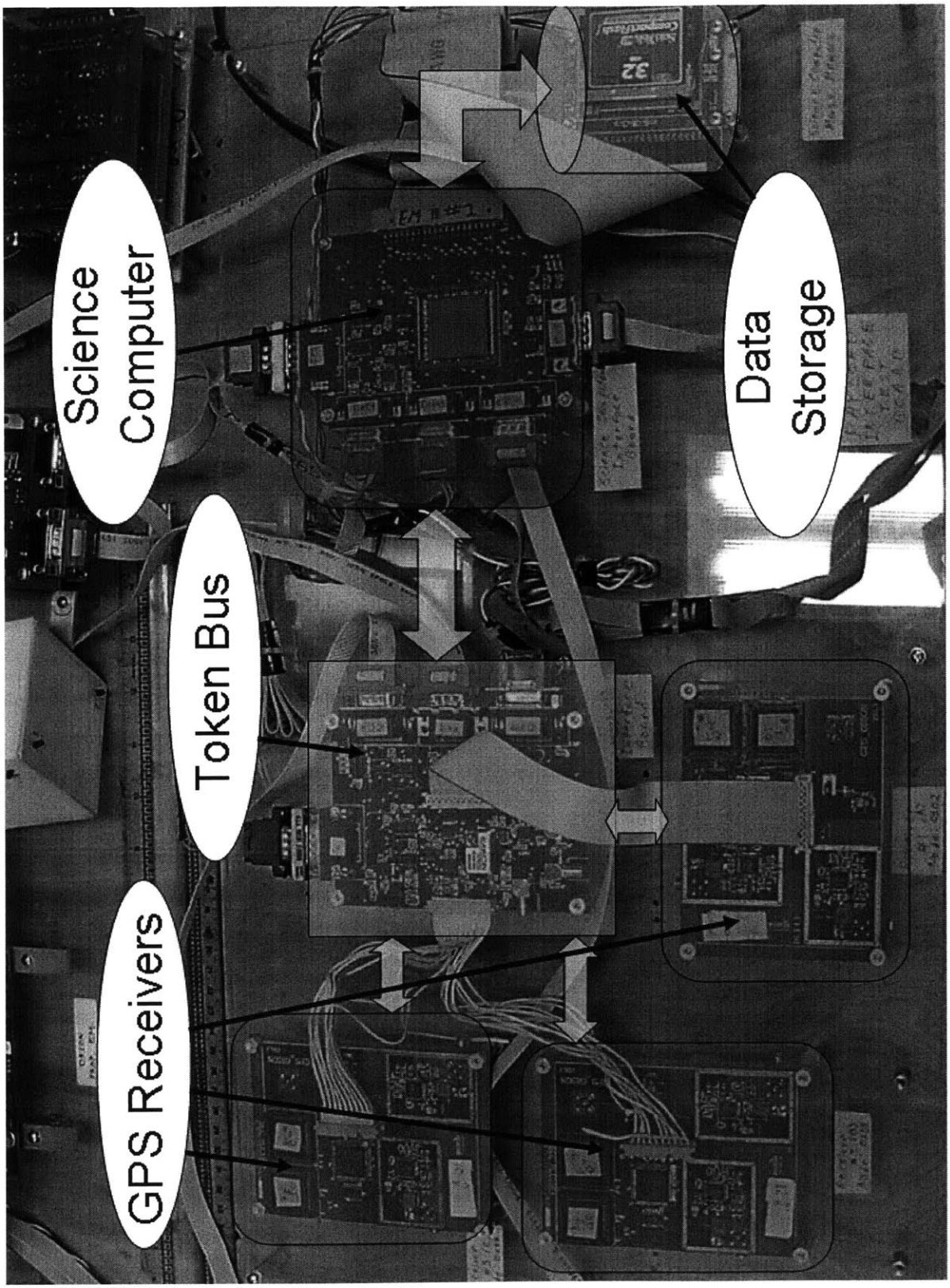


Figure 2-14: Orion Payload Architecture

torquer coils) or ground telemetry are sent through the C&DH computer for processing and relaying. In the event of a Science Computer failure, there is a secondary communications path from the C&DH computer to the GPS receiver cluster. This will permit the Orion spacecraft to continue to operate through a Science Computer failure in a degraded state.

2.4 Mission Status

The project passed Phase 0/1 safety in August 2000 and passed its CDR in November 2000. As of July, 2001, an engineering model of Orion had been built and extensively tested. In particular, the Orion structure had successfully passed its vibration/static load tests and the propulsion system had been successfully leak tested. The token bus hardware and software for the Science Computer/GPS cluster interface had been developed and testing was ongoing. C&DH and modem hardware interfaces had been finalized, built and tested - software development was ongoing. The GPS hardware and estimation approach had been extensively tested on the NASA GSFC formation flying testbed [16, 17, 18].

Remark: Currently, the first Orion vehicle is in its final assembly and checkout phase. Construction of the second vehicle has started, but has not progressed as far as the first. Scheduling problems have eliminated the original MSDS launch option, and we are currently investigating several other launch possibilities. In particular, work is ongoing to develop a new launch platform to carry the two microsatellites in the Space Shuttle payload bay, and analyses are being performed to determine changes that will be required to the satellite structure and power systems to fly in this configuration. We are also studying the orbit insertion options to determine approaches that comply with Shuttle safety protocols and reduce the fuel costs of the initial rendezvous. Software writing and validation is ongoing, with no major modifications required with the change in mission format. □

Chapter 3

The Formation Flying Information Technology Testbed

3.1 Introduction

Chapters 1 and 2 discussed the importance of formation flying to future NASA missions. However, to reduce the risk associated with these new formation flying technologies, testbeds are required that will enable comprehensive simulation, experimentation, and validation [47]. As such, the objective of this chapter is to present a new formation flying testbed developed to perform a comprehensive investigation of both distributed and centralized relative navigation, coordination, and control approaches.

A key aspect of autonomous formation flying vehicles is the selection of an appropriate estimation and control architecture and determining how the chosen architecture impacts the overall performance of the fleet. However, the architecture selection process is very complicated, and involves several trade-offs that include communication, computation, flexibility, and expansibility [48]. These issues arise because the computational and communication requirements of a centralized estimator / controller grow rapidly with the size of the fleet. However, many of these difficulties could be overcome by developing decentralized approaches for the system. Standard advantages of decentralized systems include modularity, robustness, flexibility, and extensibility [49]. Note that these advantages are typically achieved at the expense of degraded performance (due to constraints imposed on the solution algorithms) and an increase in the communication requirements because the processing units must exchange information [49].

In choosing which architecture or hybrid is appropriate for a particular fleet estimation and control scheme, one needs to look closely at not only the data flow between the vehicles in the fleet, but also to the timing involved in the computation and data transfer. The basic data rates and computational demands of each algorithm can be analyzed for a selected architecture, but this analysis would be very complicated when all aspects of the estimation, coordination, and control are implemented. Thus it is also important to develop testbeds that can be used to perform a detailed analysis of the distributed informational and computational flow. Testbeds that focus on high-level data and computational flow rather than low-level, operating system specific implementations can achieve this goal.

Several hardware testbeds have already been developed to focus on demonstrations of the basic concepts [19, 20, 21], testing the implementation of the real-time code [22] and integrating actual flight hardware in the loop [23, 24]. However, none of these testbeds directly address the inter-spacecraft communication expected on future formation flying missions, which could be a key factor in comparing control architectures due to the cost, power, mass and expansibility issues that arise when choosing inter-spacecraft communication systems for small and cheap microsatellites. Furthermore, formation flying explicitly involves distributed information (measurements and solutions) that must be processed using algorithms on distributed computers (onboard each vehicle), so it is important that a testbed be available that can be used to analyze the performance (*e.g.,* navigation and control accuracy), efficiency (*e.g.,* relative computational load of the various processors), and robustness (*e.g.,* flexibility to account for changes in the fleet) of the various alternative implementations. Finally, in stressing the real-time implementation of the software, existing testbeds require that algorithms be coded in "C" for a new operating environment. While this step is consistent with the ultimate objectives, it tends to greatly increase the complexity of modifying the control/estimation architectures, making the testbeds unsuitable for analyzing various alternatives and combinations.

With these thoughts in mind, this chapter presents an innovative hardware-in-the-loop testbed for developing and testing estimation and control architectures for formation flying spacecraft. The testbed consists of multiple computers that each emulate a spacecraft in the fleet. These computers are restricted to communicate via serial cables to emulate the actual inter-spacecraft communications expected on-orbit. A unique feature of this testbed is that all estimation and control algorithms are imple-

mented in MATLAB, which greatly enhances its flexibility and reconfigurability and provides an excellent environment for rapidly comparing numerous algorithms and architectures. Several instances of MATLAB run simultaneously on each computer, which can be used to emulate the multi-tasking/multi-thread environment typically planned for spacecraft software. Of course, apart from the benefits described above, the testbed also enables the estimation and control to be performed in parallel thereby permitting execution of the code on a realistic time-scale. This is essential because it provides the correct amount of time for representative inter-spacecraft communication and computation to take place without having to simulate it in software.

3.2 Architectures

As discussed in Ref. [11], a typical formation flying control system includes a complex interaction between various estimation, coordination, and control algorithms. Some of these algorithms can naturally be decentralized or distributed, but others require combined information and thus must be performed within a centralized or hierarchic architecture. Typically, the decision to be made with regards to architecture design is one of *distribution*. Splitting up estimation, coordination, or control algorithms for distribution across the fleet can provide benefits such as robustness, flexibility, speed, and improved autonomy. Furthermore, the modularity inherent in distributed architectures usually lends itself easily to expansion. Also, splitting up the algorithms for execution across the fleet allows for parallel processing which, if scaled properly, could dramatically reduce the computation time compared to a completely centralized architecture. Of course, these benefits of distributed architectures must be weighed against the apparent disadvantages, which include increased inter-spacecraft communications, possible non-determinacy of solutions and higher mission risk stemming from the increase in overall architecture complexity. When analyzing the degree to which algorithms can be distributed, it is convenient to identify the following architecture categories:

- **Centralized:** One entity performs the computation for the fleet. In this type of architecture, each spacecraft sends its measurements and other data to one location for processing. The end results of the centralized operation (estimation results and/or control commands) are then broadcast back out to the fleet vehicles.

- **Distributed:** Large parts of the computation are allocated to other computational nodes in the fleet for parallel processing. The distinguishing characteristic of these architectures is that the intermediate results at each node are typically not meaningful on their own; the information must be integrated at a central location.
- **Decentralized:** This type of architecture is similar to distributed architectures in that the overall algorithm is executed in parallel across the fleet. However, in this case the results computed by the individual nodes are meaningful and often represent a component of the overall solution. In this case, the final solution is already distributed across the fleet.
- **Hierarchic:** These architectures involve hybrids of the above three architectures.

The distinction between the different types of architectures is of paramount importance when attempting to integrate several algorithms together. For instance, decentralized architectures might appear superior as a stand-alone algorithm because it is executed in parallel. However, if the final result needs to be used in another algorithm that cannot be effectively distributed, decentralized architectures could lose some of their advantage because the solution ends up distributed across the entire fleet.

These architectures all involve distributed computation and significant communication of both raw data (*e.g.,* GPS carrier phase measurements) and solutions (*e.g.,* estimated positions and velocities, coordination solutions). As such, a sophisticated testbed is required that can accurately assess the feasibility/performance of the proposed estimation and control algorithms with correct computational and communication limitations in place.

3.3 Algorithms

One complication when analyzing various information architectures is that estimation, coordination, and control algorithms must be developed for each configuration to correctly establish the computation and communication requirements. In particular, in order to be able to make specific statements regarding the benefits and/or disadvantages of certain architectures, it is necessary to perform an in-depth analysis of several estimation and control approaches. Fortunately, much work has been done on the navigation and control for formation flying, and these techniques can be

used in this analysis. This section presents centralized and decentralized versions of estimation, coordination, and control algorithms that have been used on the testbed described in this chapter. The algorithms presented here are part of a larger effort focused on the Orion formation flying mission [39, 9].

All aspects of the Orion software have been implemented except the Attitude Control System. To simplify the simulations for this thesis, it is assumed that an attitude controller is regulating the attitude so that the spacecraft body frame remains closely aligned with the local-vertical local-horizontal frame. This enables the vehicle to maintain a GPS lock on the satellites. Note that the vehicle attitude motion significantly complicates the GPS relative navigation, but has a small impact on the initial architectural investigations in this thesis.

The following sections briefly describe the set of algorithms used to evaluate the testbed presented in this chapter.

3.3.1 Estimation Algorithms

For the formation flying applications of interest in this chapter, estimation of relative position and velocity is performed using measurements from the NAVSTAR satellites (see Refs. [50, 16] for details). Attaching the formation frame to a *master* vehicle (designated as vehicle m), the measurements from the NAVSTAR constellation can then be written in vector form as

$$\Delta\phi^s{}_{mi} = H_m \begin{bmatrix} X_i \\ \tau_i \end{bmatrix} + \beta^s{}_{mi} + \nu^s{}_{mi} \tag{3.1}$$

where

$$H_i = \begin{bmatrix} los_i^1 & 1 \\ los_i^2 & 1 \\ \vdots & \vdots \\ los_i^n & 1 \end{bmatrix}$$

$\Delta\phi^s{}_{mi}$ = differential carrier phase between vehicles m and i

X_i = position of vehicle i relative to vehicle m

τ_i = relative clock bias between receivers on vehicles m and i

β^s_{mi} = carrier − phase biases for single − differences

$$\nu^s_{mi} \;=\; \text{carrier} - \text{phase noise}$$

H_i is the traditional geometry matrix. The components los^k_i are the line-of-sights from the ith user vehicle to the kth NAVSTAR satellite in the formation coordinate frame. For an N-vehicle formation, these measurements are combined into one equation

$$\begin{aligned}\Delta\Phi^s &= \begin{bmatrix} H_m & & 0 \\ & \ddots & \\ 0 & & H_m \end{bmatrix} \begin{bmatrix} X_1 \\ \tau_1 \\ \vdots \\ X_{N-1} \\ \tau_{N-1} \end{bmatrix} + \begin{bmatrix} {\beta^s}_{m1} \\ \vdots \\ {\beta^s}_{mN-1} \end{bmatrix} + \nu^s \\ &= HX + \beta^s + \nu^s \end{aligned} \tag{3.2}$$

where

$$\Delta\Phi^s = \begin{bmatrix} {\Delta\phi^s}_{m1} \\ {\Delta\phi^s}_{m2} \\ \vdots \\ {\Delta\phi^s}_{mN-1} \end{bmatrix}$$

and it is assumed that the *master* vehicle m has visibility to all available satellites and all vehicles track the same set of satellites [50]. This assumption is consistent with the formation flying missions of interest that have relatively short baselines, and thus all the vehicles can see the same set of NAVSTAR signals.

In general this may not be the case, and H may have off-diagonal terms corresponding to the single differences that can be formed between vehicles using NAVSTAR signals not available on the *master* vehicle. In addition, not all of the block diagonal entries will be H_m as shown in Eq. 3.2. For example, if the kth GPS satellite was not visible on vehicle m, but was visible on vehicles i and j, then the following single difference could be formed

$$\Delta\phi_{ji} + R^k_j(1 - los^k_i \cdot los^k_j) = los^k_i(X_j - X_i) + \tau_j - \tau_i + \beta^s_{ij} + \nu^s_{ij} \tag{3.3}$$

where R^k_j is the range from the jth vehicle to the kth GPS satellite. This measurement would be added to those using the *master* vehicle, and would appear in the off-diagonal elements of H. However, note that since we are interested in missions with a clear sky view (formation flying missions) and relatively short baselines, we are mainly

interested in scenarios wherein all the vehicles can see the same set of NAVSTAR signals.

It is also assumed that the coupling between the measurements (due to effects such as differential ionosphere) are negligible, resulting in a block-diagonal measurement matrix H. It has been shown in [16] that for small fleet separations, this is an appropriate assumption.

A similar process can be done for the Doppler measurements to form a second set of measurement equations. The two velocity and position equations can then be combined into a single matrix equation

$$\begin{bmatrix} \Delta\Phi^s \\ \Delta\dot{\Phi}^s \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & H_v \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \end{bmatrix} + \begin{bmatrix} \beta^s \\ 0 \end{bmatrix} + \begin{bmatrix} \nu^s \\ \dot{\nu}^s \end{bmatrix} \tag{3.4}$$

$$\triangleq H \begin{bmatrix} X \\ \dot{X} \end{bmatrix} + \begin{bmatrix} \beta^s \\ 0 \end{bmatrix} + \begin{bmatrix} \nu^s \\ \dot{\nu}^s \end{bmatrix} \tag{3.5}$$

With these measurements, a decentralized Extended Kalman filter is derived in [26] that solves for the vehicles' states making use of the block-diagonal form of H. Later, in Chapter 4, the topic of estimator decentralization is explored further when the measurements are augmented with local ranging devices. However, for the purposes of testbed evaluation, this relatively simple estimation scenario will suffice.

3.3.2 Coordination Algorithms

With a large number of vehicles, the computational aspects of the fleet trajectory planning are complicated by the large information flow and the amount of processing required. Typical problems that must be addressed by the high-level fleet controller are: (i) to ensure that the vehicle maneuvers are designed to avoid collisions and plume impingement; and (ii) design fuel/time-efficient ways to move each spacecraft in the fleet to their locations in a new configuration that provides a different science viewing mode. These are challenging optimization problems that can require significant computational effort to solve by a centralized algorithm [51]. However, this computational load can be balanced by distributing the effort over the fleet [52]. The result is a list of predicted fuel costs for every possible final location (called a ΔV map), which are used to generate the fuel cost to move the fleet to each global configuration. These costs are based on fuel usage, but they could include other factors,

such as the vehicle health. The ΔV maps are given to a centralized coordinator to assign the final locations, which can be done by solving a simple assignment problem. The resulting hierarchic architecture uses each vehicle to estimate the cost to perform various alternative maneuvers using linear programming. These predicted fuel costs are then used in a centralized assignment problem to allocate the locations in the new configuration to each vehicle [52]. Collision avoidance and plume impingement must then be verified for the selected configurations.

Many alternative formation flying control strategies have been recently proposed. As discussed in the following section, work by Tillerson [53] has focused on using a model-predictive control approach based on linear programming (LP). A key advantage of this approach is that it can directly include state constraints (*e.g.,* errorbox limits) and input constraints (*e.g.,* actuator limitations) in the *formation-keeping* trajectory optimization. This formation-keeping can easily be decentralized given local measurements of the relative position/velocity of the satellite with respect to the time-varying *desired state* in the current configuration. But this requires distributed knowledge of the desired locations, which can be obtained by propagating the states associated with a "template" of the desired passive aperture[6] about a virtual center [54, 55, 53]. By allowing extensive cooperation in determining the set of desired points for the formation, the use of a virtual center extends the formation-keeping control to the full *formation flying control problem* [53].

As this discussion illustrates, the full formation flying control system involves a combination of centralized (template initialization, propagation, and monitoring) and decentralized calculations (LP trajectory optimization).

3.3.3 Formation-keeping Control

Disturbances such as differential drag, J_2, and errors in the linearized dynamics will cause the satellites to drift from the designed periodic motion associated with the *passive apertures.* So control effort is required to maintain a state that results in the periodic motion. Linear programming (LP) can be used to develop fuel-optimal control inputs to move the satellite from the disturbed state back to the desired state, or to maintain the satellite within some tolerance of the desired state.

The formation-keeping problem is comprised of two issues. The first issue is what

[6]Typically short baseline periodic formation configurations that provide good, distributed, Earth imaging while reducing the tendency of the vehicles to drift apart.

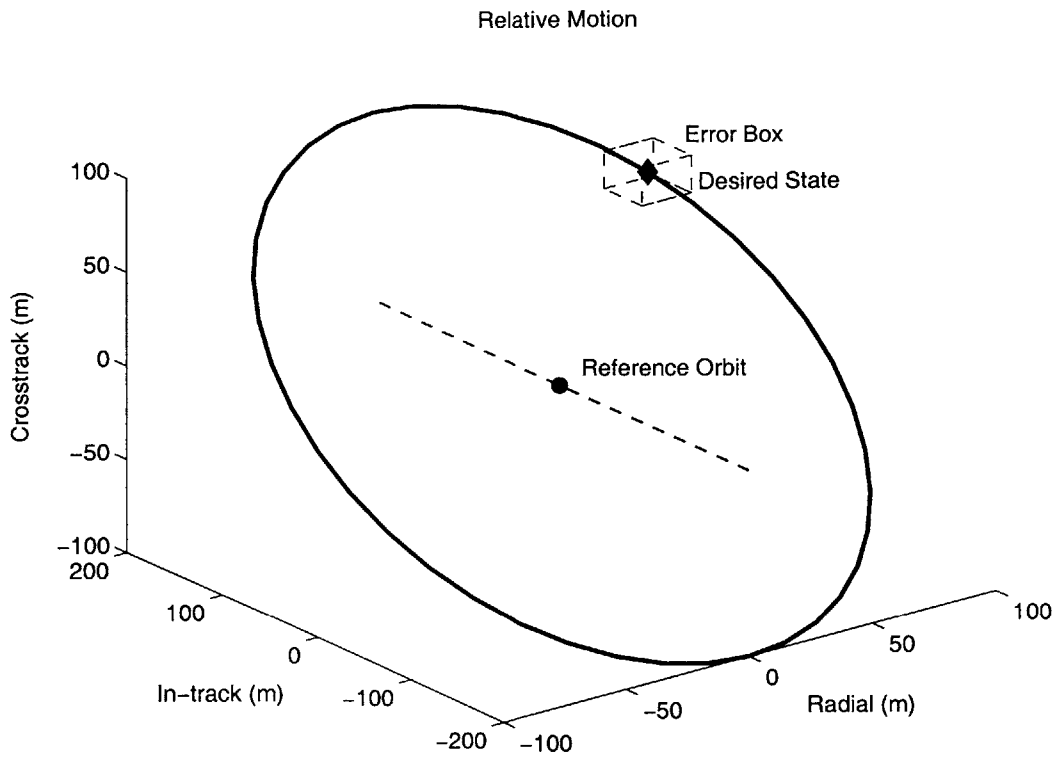


Figure 3-1: Motion of satellite relative to a reference orbit. Current position of reference orbit denoted with •, current desired relative position is ♦, which is the center for the error box.

relative dynamics and initialization procedure should be used to specify the desired state to maintain the passive aperture formation. The desired state is shown in Figure 3-1 as ♦ and the reference orbit position as •. The periodic motion followed in the absence of disturbances is also shown. The desired state can be determined from the closed-form solutions of the linearized dynamics and the initial conditions [56]. These initial conditions are then used in the corresponding closed-form solutions to determine the desired state at any other time. Ref. [56] analyzes the use of various models to perform these initializations and predictions.

The second issue for formation-keeping is which relative dynamics to use in the actual LP problem. The error box is fixed to the desired state as in Figure 3-1. The desired state is centered in the error box, but the true state of the satellite will be disturbed from the desired state by differential drag, J_2, or other disturbances. The error state is then the difference between the current state and desired state relative to the reference orbit. The dynamics used in the LP are the dynamics relative to the desired state.

The basic form of the formation-keeping LP problem can be written as

$$\min \|u\|_1 \text{ subject to } \mathbf{A}u \leq \mathbf{b} \tag{3.6}$$

where u is the vector of fuel inputs (ΔV) at each time-step and $\mathbf{A}, \mathbf{b}$ are functions of the linearized spacecraft dynamics, initial conditions, and all constraints. Constraints to the problem can include: state constraints such as remaining within some tolerance (error box) of a specified point, maximum input values (actuator saturation), and terminal constraints. The LP determines the control inputs for a specified time interval that minimizes the fuel cost while satisfying the constraints on the trajectory. This approach can also include differential disturbances such as drag and linearized forms of the differential J_2 effects. To complete the low-level control design, the LP is embedded within a real-time optimization control approach that monitors the spacecraft relative positions and velocities, and then redesigns the control input sequence if the vehicle approaches the edge of the error box [52].

The formation-keeping described above can easily be decentralized given local measurements of the relative position/velocity of the satellite with respect to the desired state on the passive aperture. However, this requires knowledge of the desired states, which, as discussed previously, can be obtained by propagating the states of the desired passive aperture [53]. The template can be initialized using the GPS measurements (absolute and relative) from all vehicles in the fleet.

3.3.4 Thrust Mapping and Fault Detection

The LP formation control algorithms generate a set of desired ΔV changes for the vehicle that are aligned with a local-vertical local-horizontal frame. It is the job of the thruster mapper to convert these to a set of thruster commands using the current attitude measurements and knowledge of the thruster capabilities. If all thrusters have linear effects on the spacecraft movement, the thrust mapping problem can be solved via a simple LP, which can be solved quickly and can accommodate various types of actuators. The LP formulation selects the on-times, U, of the available thrusters that satisfies the desired maneuvers, T, at the minimum fuel cost

$$\min\{C_{\text{map}}U \mid A_{\text{map}}U = T, \quad 0 \leq U \leq U_{\max}\} \tag{3.7}$$

The cost vector, C_{map}, is a $1 \times n$ vector containing the cost of using the n thrusters. Each column of the thrust mapper matrix, A_{map} $(m \times n)$ corresponds to the acceleration vector of a particular actuator.

Work by Yang [14] developed a static Kalman filter along with a Generalized Likelihood Ratio (GLR) test to monitor the long-term performance of the thrusters and provide updates to the thrust mapper matrix, A_{map}, when degradation and/or failures occur. While the Kalman filter provides an optimal filter for characterizing the long-term trends in the actuator performance, it is not a particularly good technique for fast detections of unexpected and sudden actuator failures. The GLR failure detection scheme not only provides a fast detection system but also has a very low false-alarm ratio. The GLR test isolates the failed actuator by using the knowledge of how different thruster failures affect the spacecraft. In the algorithms implemented on the testbed, failures are assumed to be random events and only one actuator can fail at each step. Once a failure is detected a model-comparison (MC) estimation process is used to identify the exact nature of the failure. The MC estimation applies different failure models to the identified actuator. Failure models include full-on, full-off and partial degradation failures. A $\mathcal{X}^2$-test is applied to the different models and the one with the lowest $\mathcal{X}^2$ value is the estimated failure type. The new thruster model is then updated to the thrust mapper.

3.3.5 Autonomous Task Allocation

After an algorithm has been broken up into small tasks that can be computed in parallel, each task must be assigned to a spacecraft for processing. Due to the differing processor loading levels across the fleet, it may not be efficient to divide the tasks evenly between the spacecraft. The purpose of the Task Allocation algorithm is to determine an appropriate distribution of computational effort that minimizes both the time required to complete the overall computation and the amount of inter-spacecraft communication required to define the computational parameters for each task (*i.e.,* initial conditions or intermediate results). The relative computational loading of each spacecraft is periodically estimated by measuring the time required to complete a well-defined computation. Using these estimates, a central spacecraft determines the ideal task distribution by assigning the least loaded spacecraft the most work.

Typically, task distribution results in smaller computational units that cannot be

further distributed. Thus, the next job of the Task Allocation algorithm is to assign computational units to the spacecraft such that the ideal distribution is approximately met while minimizing the required inter-spacecraft communication. This allocation is currently performed using simple heuristics, but more sophisticated assignment algorithms could be investigated as future work.

3.4 Example Architectures

To illustrate the complexity associated with combining estimators and controllers on a fleet of vehicles, two examples of possible architectures will be presented here. The first architecture will combine a centralized GPS estimator with the distributed coordination controller described above. The second architecture will combine a decentralized GPS estimator with the same distributed coordination controller. In both cases, the information and computational flow requirements will be analyzed at the instant that the new coordination maneuver is planned. Also, for these examples, it will be assumed that the fleet consists of 3 slave vehicles and 1 master vehicle[1].

3.4.1 Example Architecture 1 - Centralized Estimator with Distributed Controller

The algorithmic flow for the first example is illustrated in Figure 3-2.

1. Each slave vehicle sends their raw GPS phase measurements to the master vehicle.
2. The master differences the slaves' measurements with the measurements taken on the master vehicle and performs a measurement update on the entire fleet state.
3. The master sends each slave vehicle the navigation solution pertaining to their specific state.
4. Each slave computes the ΔV maps to arrive at each location on the new ellipse proposed by the master.
5. Each slave transmits the resulting ΔV maps to the master.
6. The master receives all ΔV maps and computes the optimal assignment.

[1]For the purposes of these examples, the controller master and estimator master are taken to be the same vehicle, but in practice, this need not be the case.

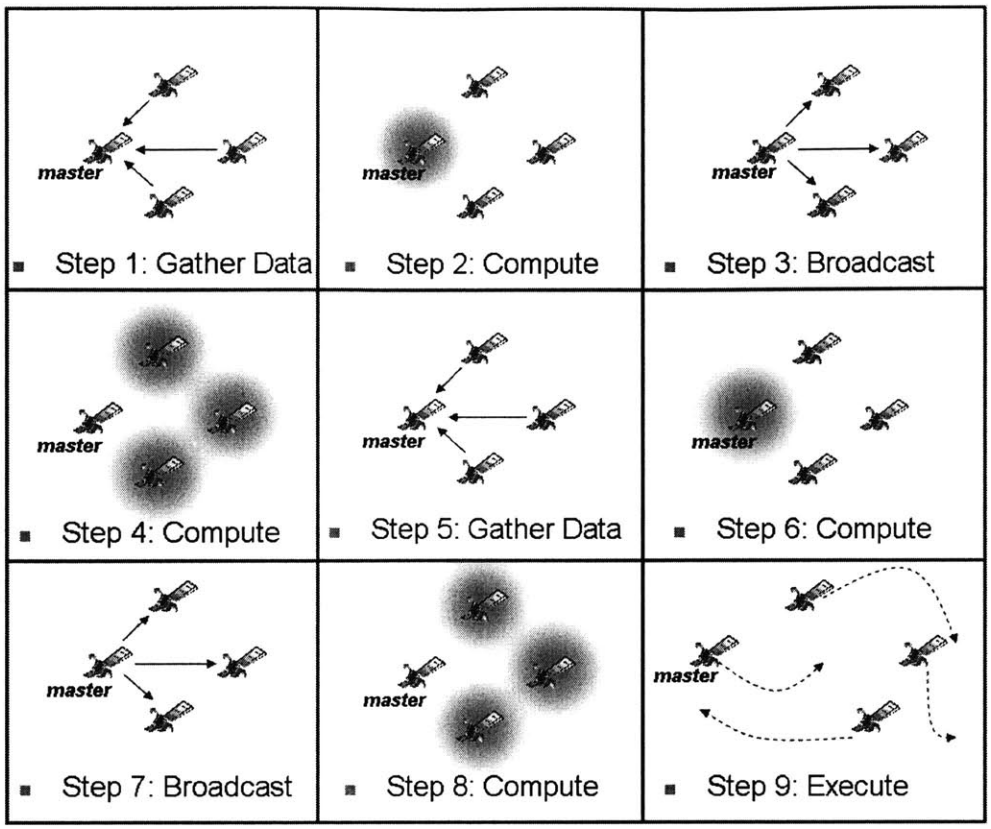


Figure 3-2: Example Architecture 1 - Centralized Estimator, Distributed Controller.

7. The master sends the optimal assignment out to each slave.
8. Each slave recomputes the trajectory corresponding to its optimal assignment.
9. The coordination maneuver begins.

Since the data is collected in a distributed fashion, it must be gathered at a central location prior to processing, which can be a time-consuming task, especially if the master vehicle can only process one incoming message at a time (which is typically the case for nanosatellites [9]). The situation is further complicated when the distributed control phase begins because each vehicle must now obtain the centrally computed navigation solution, which requires another communication step.

A further complication with this architecture is the degree of synchronization that is required, which is typical of centralized algorithms. For instance, step 2 cannot take place until all slaves have sent their data to the master. Furthermore, the slaves must wait until the master has sent out the navigation solution before computing their ΔV maps. This synchronization requirement can result in substantial idle time for some spacecraft in the fleet, possibly leading to an inefficient design.

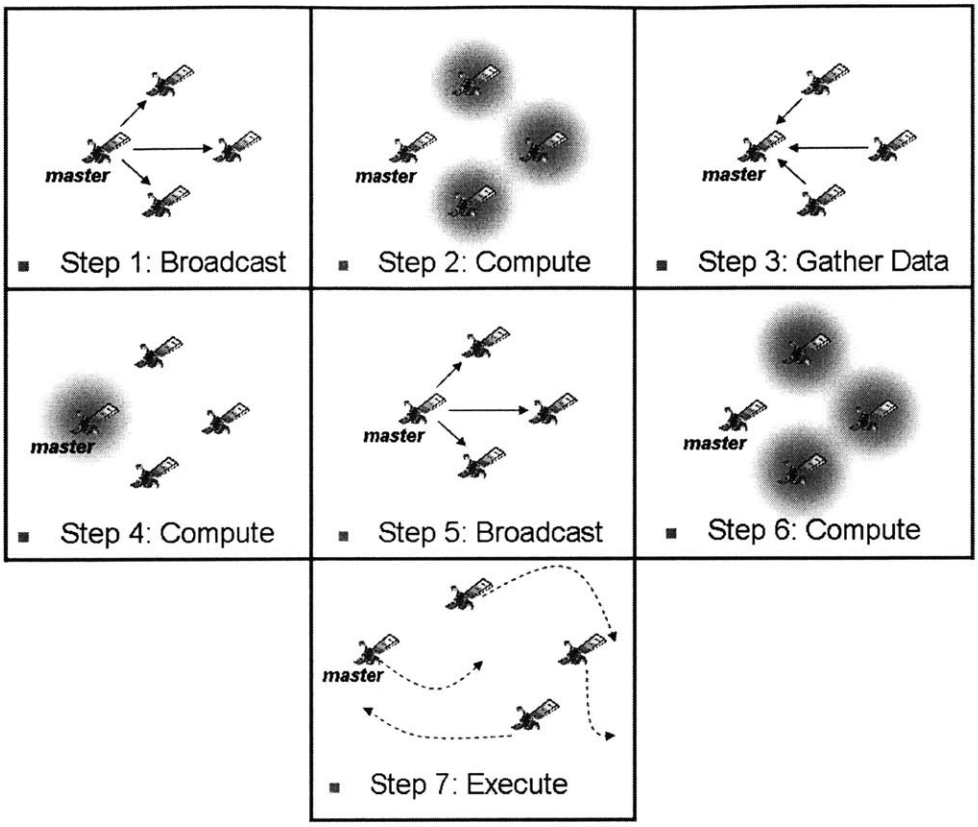


Figure 3-3: Example Architecture 2 - Decentralized Estimator, Distributed Controller.

3.4.2 Example Architecture 2 - Decentralized Estimator with Distributed Controller

The algorithmic flow for the second example (illustrated in Figure 3-3) is described in detail below.

1. The master broadcasts its raw GPS phase measurements to all slaves.
2. Each slave differences its own phase measurements with the master's upon receipt and performs a measurement update for only its state. Each slave then computes the ΔV maps to arrive at each location on the new ellipse proposed by the master. Note that all computations in this step occur on each slave independently of one another.
3. Each slave transmits the resulting ΔV maps to the master.
4. The master receives all ΔV maps and computes the optimal assignment.
5. The master sends the optimal assignment out to each slave.
6. Each slave recomputes the trajectory corresponding to its optimal assignment.

7. The coordination maneuver begins.

Decentralizing both the estimator and a portion of the controller has removed some of the complexity of the previous example architecture. In this case, the master broadcasts its phase measurements for each slave to difference independently instead of one vehicle having to process multiple messages from around the fleet. Also, since each vehicle performs their own estimation locally, the solution is already in the correct location for the distributed control to start.

As will be explored further in Chapter 4, decentralizing the estimator also results in a more balanced computational load. In these examples, the algorithmic decentralization was relatively simple due to the assumption of decorrelated measurements. If measurements such as local ranges were also introduced (as described in Chapter 4), the assumption of decorrelated measurements no longer holds and the decentralization becomes much more challenging. In the case of correlated measurements, estimator decentralization is only approximate and one must trade some estimator accuracy for decentralization.

The previous two examples have briefly explored the intricate interactions of two fleet estimation and control architectures. In an actual fleet architecture, algorithms such as fault detection, attitude control and basic fleet maintenance must also be integrated into the overall architecture, thus imposing more synchronization, computation, communication and timing constraints. Clearly, without adequate planning and testing of the architectural setup, fleets could suffer from communication bottlenecks, computational imbalances, or simply inefficient operations. The next section introduces the testbed designed as part of this research to compare, contrast and evaluate different formation flying control and estimation architectures.

3.5 FFIT Testbed

The Formation Flying Information Technology (FFIT) testbed described in this thesis (see Figures 3-4 and 3-5) is used to simulate the computational and data flow of the control system for a fleet of LEO formation flying microsatellites. The primary goal of the testbed is to provide an environment wherein the distributed algorithms can easily be developed and executed in scaled real-time over real communication links in a way that minimizes the impact of the simulation on the actual algorithmic performance. The testbed has the following four defining features.

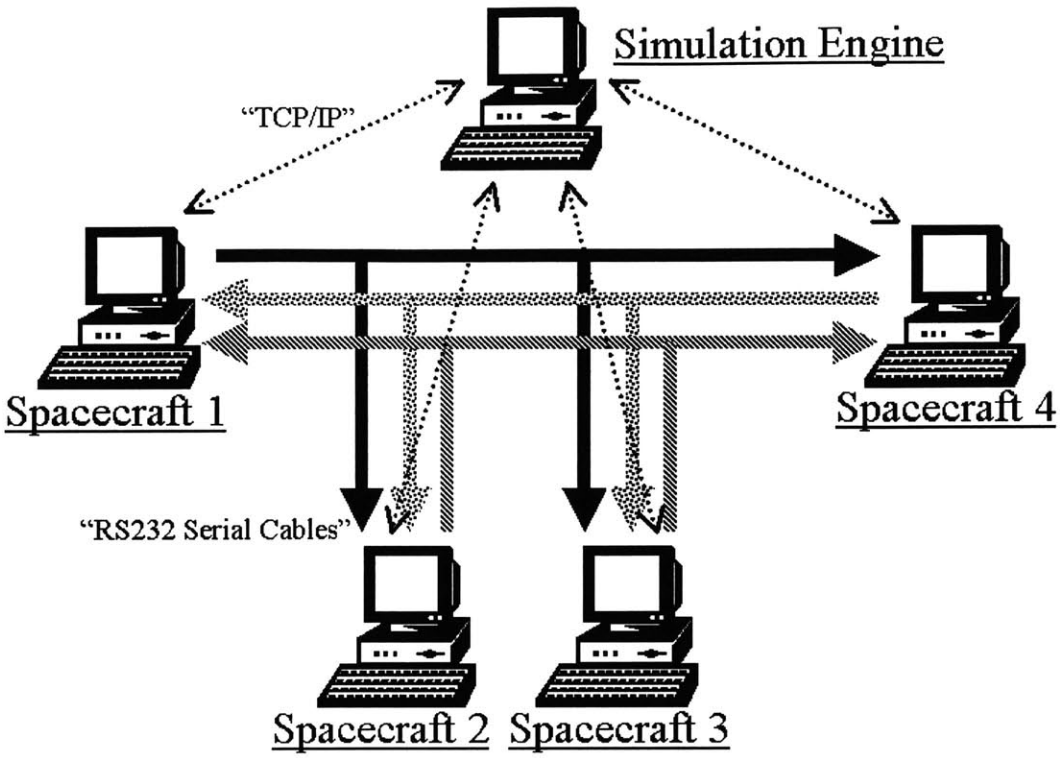


Figure 3-4: Physical Architecture of the Testbed

MATLAB: MATLAB was chosen as the programming language for the testbed because of its ease of algorithm implementation and because of the numerous toolboxes that already exist. Having to re-write the functions in these toolboxes significantly complicates the code development and hinders the rapid architecture development desired for this testbed. Thus, working in MATLAB enables a seamless transition of new control and estimation approaches from various investigators to the testbed. Working in MATLAB also provides a detailed window into the algorithms, which is excellent for debugging. Of course, an additional hardware and OS specific analysis of the software must be done prior to architecture acceptance to ensure it is within the capabilities of the chosen computer. A further benefit of MATLAB is that it provides a very clean interface to Java, which simplifies the implementation of sockets and other external communications methods. This Java extension permits low- and high-level data manipulation and transmission to be carried out using a MATLAB function. FLOP counts are available in MATLAB using the third-party software suite called PAPI [57]. The PAPI suite replaces the MATLAB *flops.m* function and

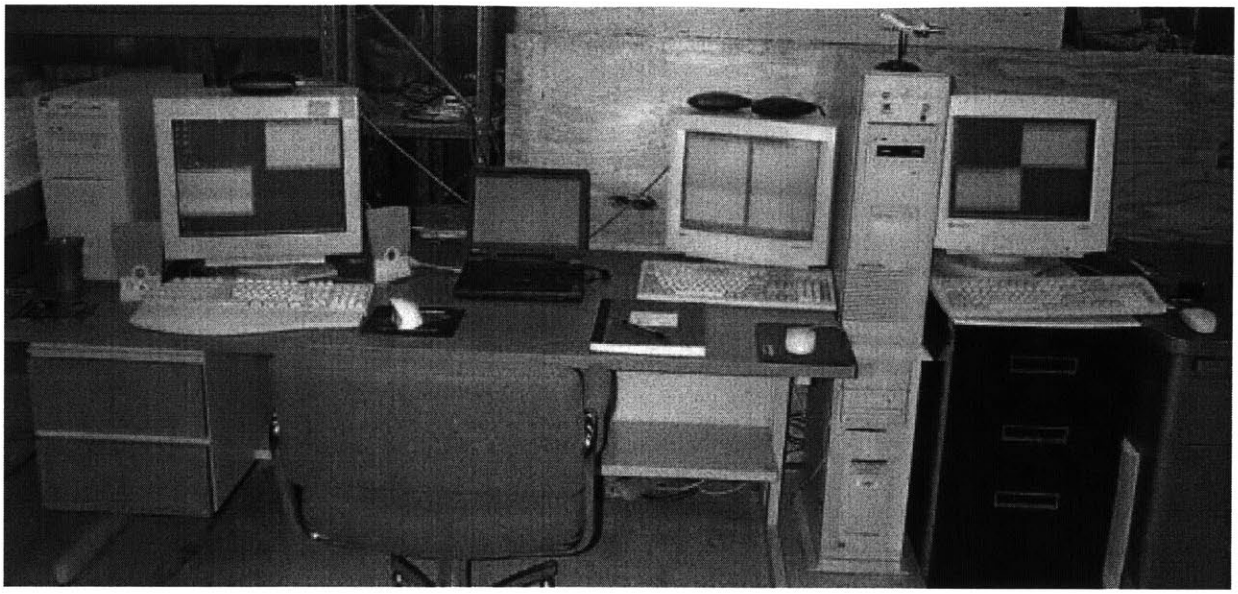


Figure 3-5: Photo of the FFIT Testbed showing 4 of the 5 computers. Notice the USB serial adapters sitting atop two of the monitors.

provides an independent count of the number of arithmetic operations being carried out at the hardware level.

RS232 Serial Connectivity: A key aspect of the testbed is that, to retain as much realism as possible, **all** inter-spacecraft communication is carried out through RS232 serial cables. The RS232 serial protocol is very representative of inter-spacecraft communications modems planned for most future microsatellite and Nanosat missions (*e.g.* [39, 9]). Through the use of Java, the baud rate of the serial connections can be altered for simulation scaling. An important aspect of inter-spacecraft communications of Nanosats is the method by which multiple spacecraft can communicate with one another. Cost, power and mass typically limit Nanosats to having very simple communications systems and thus require sophisticated multiplexing algorithms to permit multiple spacecraft to use the same communications link. In order to accurately model communication systems such as these, serial splitters have been used to force each spacecraft to broadcast every outgoing message to each spacecraft in the fleet.

To facilitate 2-way communication amongst spacecraft, a "token bus" architecture is used to ensure no data collisions occur on the "bus", thus emulating a TDMA approach to inter-spacecraft communications. Although TDMA was chosen as the original communications architecture for the studies done in this chapter, the FFIT testbed has the capability of operating in other communication modes (such as CDMA

or FDMA) with a simple flag change.

Multi-Threaded Applications: Many spacecraft software systems have several different requirements that drive the need for multi-threaded applications. For example, large optimization algorithms may take upwards of several minutes to compute. It would be impractical for all other spacecraft functions to have to wait for this optimization to complete before addressing low-level tasks such as communications and state sensing. Thus, it is desirable to implement some tasks in "background" while others run in the "foreground". While MATLAB does not have built-in support for multi-threaded applications, such programs can be implemented on the testbed using several instances of MATLAB on each computer. Using this technique, the "MATLAB Threads" communicate to each other on one spacecraft through TCP/IP sockets. Socket communication provides a fast means of interprocess communication with minimal impact to the rest of the system. The testbed uses the Windows 2000 operating system, which permits the user to set the priority of each process (*i.e.,* the different MATLAB threads).

Simulation Engine: A separate computer is used as the simulation engine to propagate the states of each spacecraft in the fleet as well as the states of each GPS satellite for navigational purposes. At each time step, the simulation computer transmits (via TCP/IP) the current simulation time as well as simulated GPS signals that would be received by the spacecraft's GPS antennas. The data sent to each spacecraft computer from the simulation computer is an exact replica of what would be received from the GPS receivers onboard the actual spacecraft. Using a GPS simulator in the simulation engine forces each spacecraft to perform its own navigation exactly as it would on-orbit. Since this data would be available virtually instantaneously onboard each spacecraft, (independent of the inter-spacecraft data traffic) using TCP/IP as the communications medium for this link alone (as well as for inter-process communication as stated earlier) does not impact the architecture performance or analysis. Since TCP/IP is an entirely separate data bus from the serial cables used for inter-spacecraft communications, the simulation data in no way interferes with the inter-vehicle communications, thus retaining representative data rates between spacecraft.

Utilizing a separate computer for simulation purposes increases the realism of the simulation because it removes any code from the spacecraft computers that would not be run in the actual flight system. Future versions of the testbed could even add a

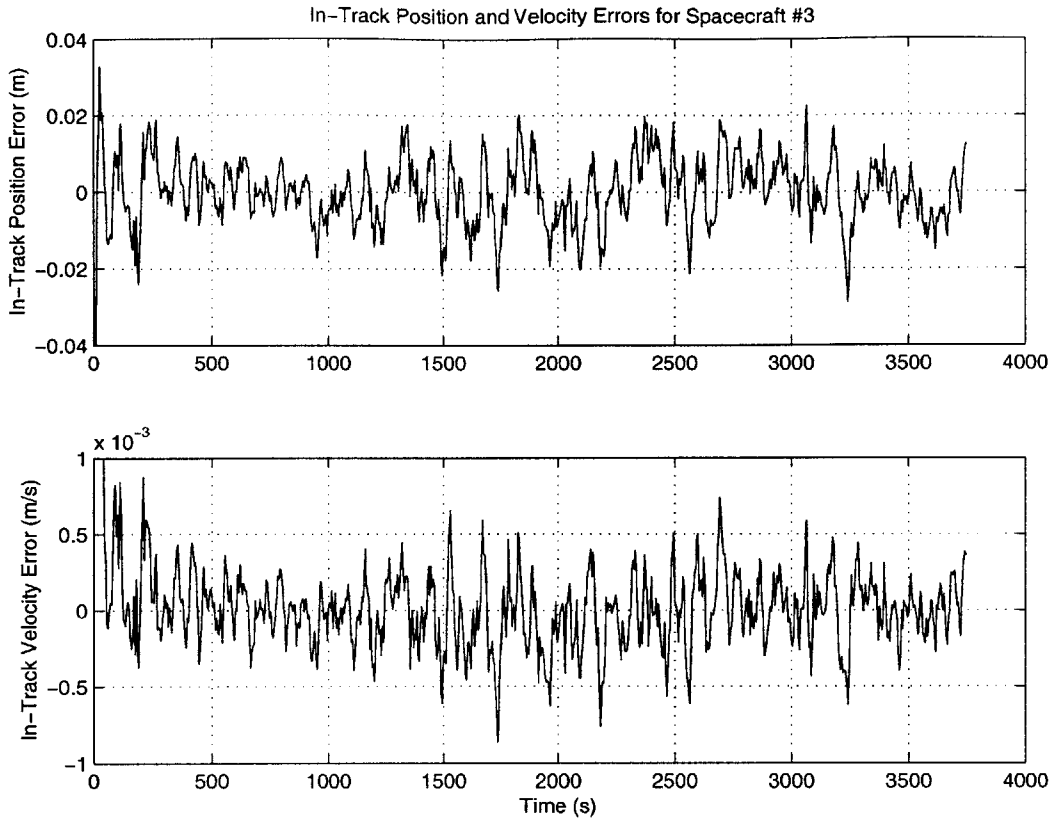


Figure 3-6: Relative Position and Velocity Errors.

second computer to the simulation engine for spacecraft visualization purposes. This computer would communicate to the simulation propagator computer to receive the latest state vector of each spacecraft and plot their relative positions in real-time.

3.6 Simulation Results

Several simulations were run to demonstrate the features of the FFIT testbed and illustrate how they can be used to evaluate fleet estimation and control architectures. The architecture chosen for the demonstration has centralized (fleet coordinator, task allocation engine), decentralized (EKF GPS estimator), and distributed (spacecraft controller, thruster fault detection and recovery system) computational components.

A primary concern when evaluating architectures is the performance of the estimation and control algorithms. The FFIT testbed records all spacecraft data throughout every real-time simulation for post analysis. Figure 3-6 illustrates the estimation accuracy achieved using the decentralized EKF GPS estimator (accuracy $\approx$1 cm in position and $\approx$0.5 mm/s in velocity). These values are comparable to the hardware-

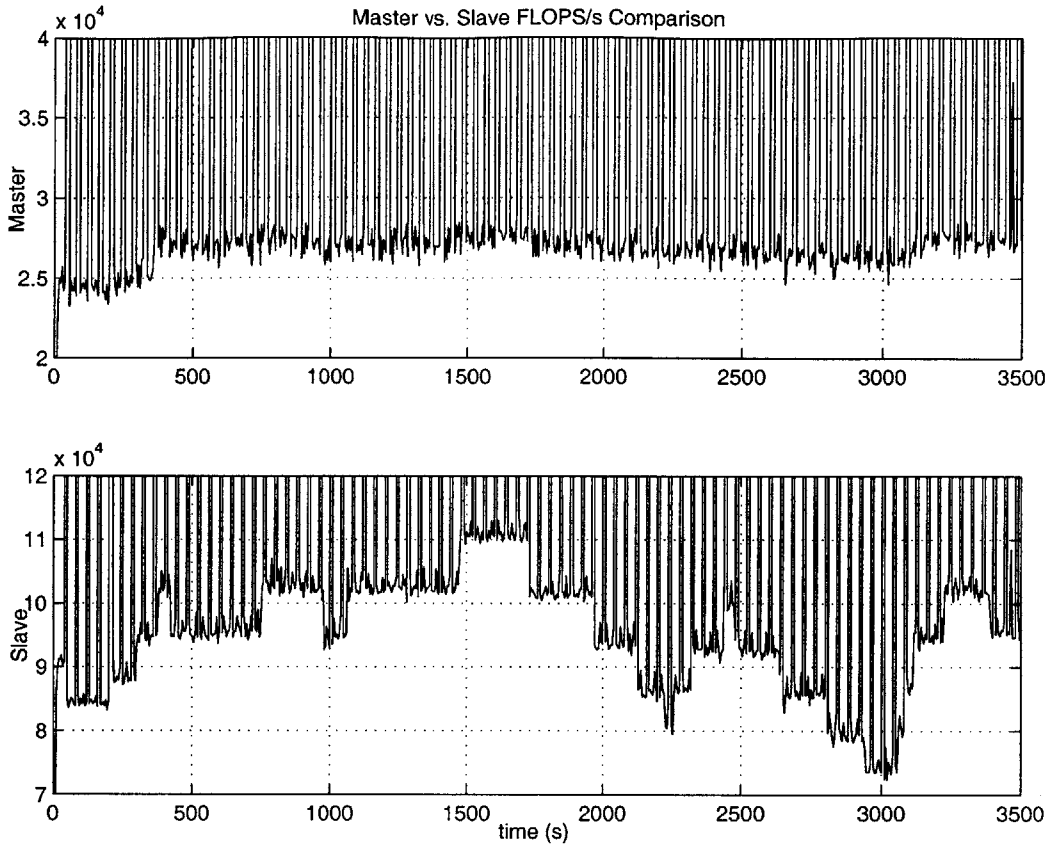


Figure 3-7: Comparison of FLOP rate baseline during decentralized estimation.

in-the-loop results in Ref. [17].

Figure 3-7 compares the amount of computational effort expended between the master and a slave during a long period of GPS estimation. Due to the decentralized architecture, the master spacecraft uses about 1/3 the amount of FLOPs as the slave. It is interesting to note the relative randomness of the slave plot versus the master plot. This is a result of the extra thread that the slaves use to run their estimation. This inter-process communication takes both time and FLOPs. The TCP/IP inter-process communication speed can vary as a result of on the processor loading. This could cause more or less FLOPs to be completed per second as the inter-process communication fluctuates. The FFIT testbed proved useful in observing this unexpected result.

Controller performance can also be evaluated on the FFIT testbed. Figure 3-8 illustrates the effectiveness of the thruster fault detection and recovery algorithms during a thruster failure. The top plot shows the spacecraft going off course when a

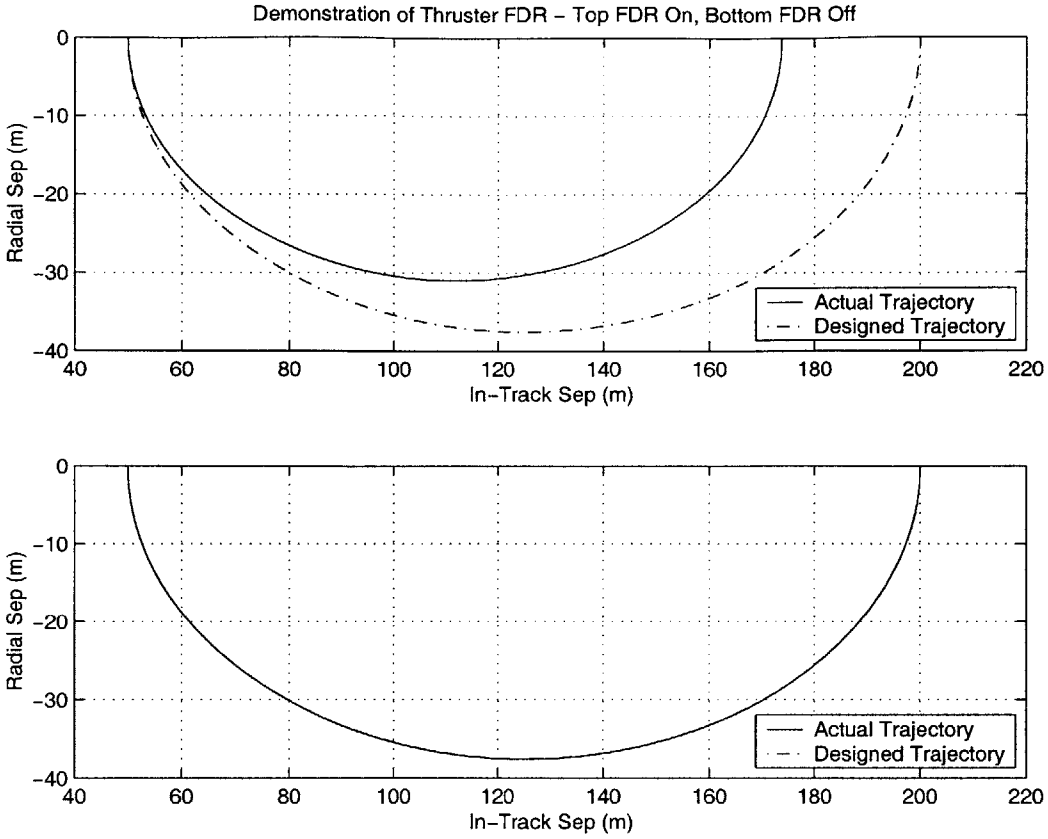


Figure 3-8: Effect of thruster failure. Top is without FDR routine, bottom is with FDR.

thruster fails "off" during a transfer maneuver. The bottom plot shows the spacecraft following the trajectory almost perfectly during the same thruster failure.

One of the key reasons for testing distributed algorithms on the FFIT testbed is to understand the impact of indeterministic effects such as communication delays and processor loading. These effects are best observed by monitoring the time required for computations to complete using a time-history of the FLOPs used. Using this technique, it is possible to evaluate the effectiveness of the Task Allocation algorithm in the *FFIT Testbed* section. Figures 3-9 and 3-10 compare the computation rate for each spacecraft during a distributed fleet coordination calculation with the Task Allocation algorithm on and off.

For the case where the Task Allocation algorithm is off (Figure 3-9), the computational units are simply divided evenly across the fleet. Without using the task allocation algorithm an equal amount of ΔV map computation takes 15 sec to complete on SC3, but nearly 55 sec on SC4. Once all ΔV maps are complete, each spacecraft must re-compute the single optimal trajectory chosen by the coordinator. This is

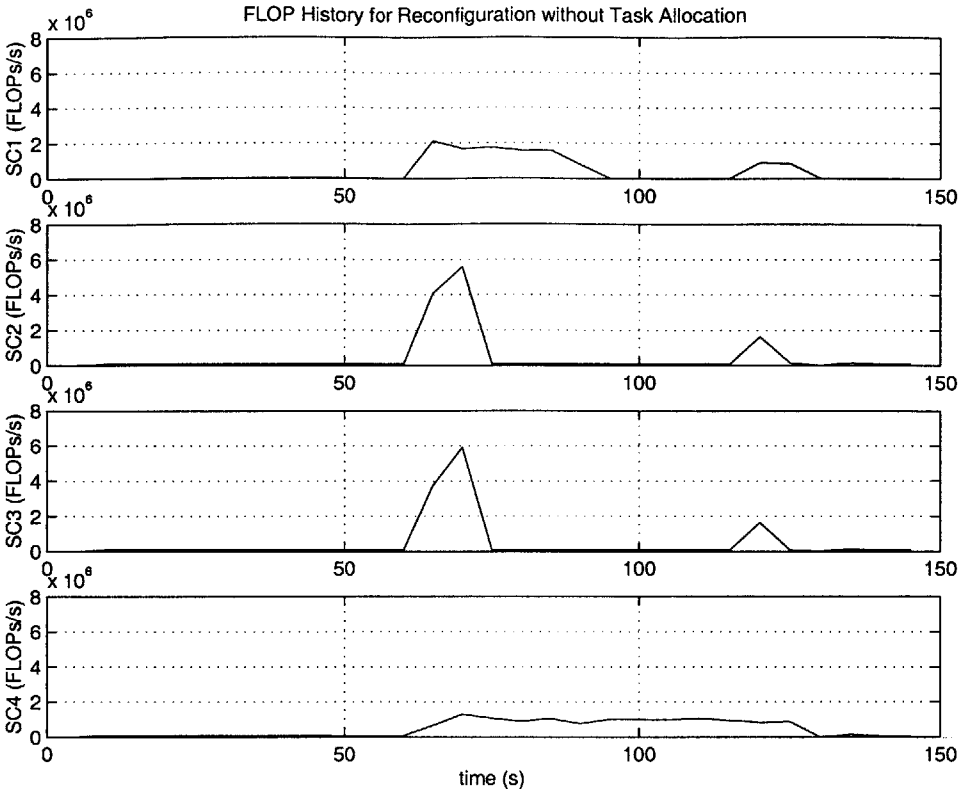


Figure 3-9: FLOP History without using Task Allocation.

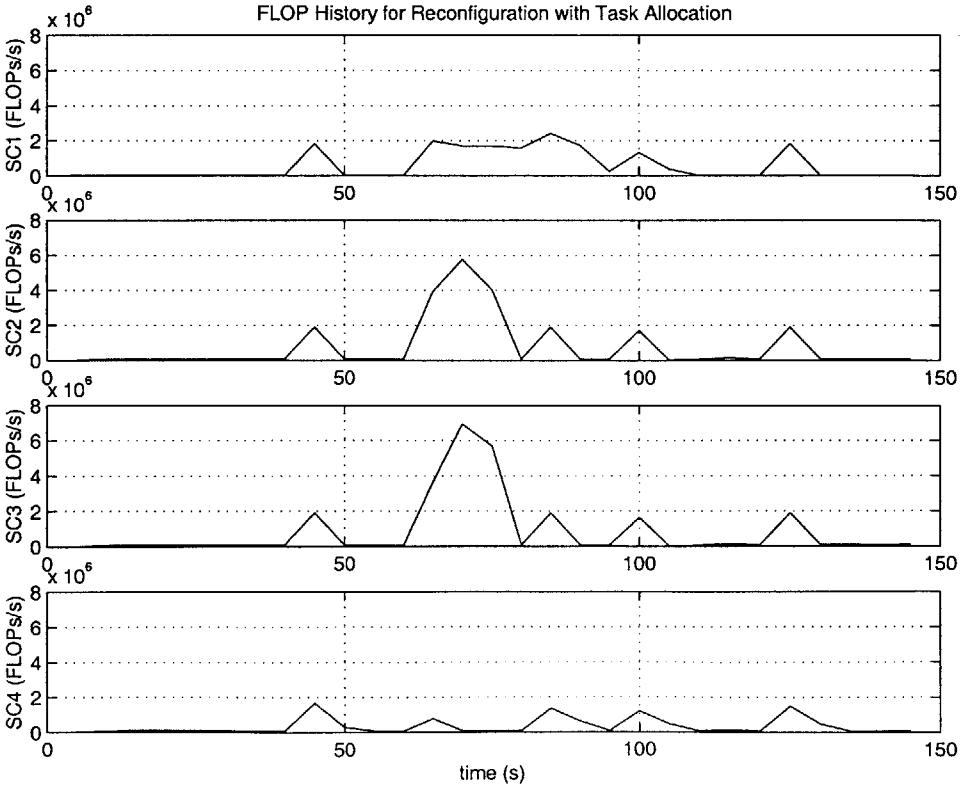


Figure 3-10: FLOP History using Task Allocation.

evidenced by the small spike at 115 sec on each plot. Since the control algorithm requires all results before a coordination plan can be made, the total computation time is the longest completion time for the fleet (70 sec). Using the Task Allocation algorithm, the coordinator determines that the SC4 is heavily loaded and actually assigns its computation to the other, more lightly-loaded computers. In this case, the total computation completed in 50 sec (29% improvement over the previous case).

Figure 3-11 is a closer view of the FLOP profile during a reconfiguration maneuver for one slave. Spikes in computation every 40 sec are observed (beyond the limits of the vertical axis) as a result of the processor timing tests required to estimate the computational load of each spacecraft.

Some interesting behavior occurs at 110 sec after the entire computation is done. Note in Figure 3-11 the sharp drop in FLOP rate (to lower than the average baseline) followed by a small peak between 110 and 120 sec. The sharp drop in FLOPs can be explained by the large data transfer required to transfer the final plan between the LP thread and the main thread. During this transfer, the processor waits for its completion before continuing, thus causing a drop in the observed FLOP rate. The reason for the small peak afterwards requires more data to explain. Figure 3-12 shows the amount of data transmitted by the master spacecraft. At 100 sec, a sharp drop is observed for the same reasons as stated above for the FLOP count drop. However, raw GPS data continues to stream in from the simulated GPS receivers and it backs up in the master's outgoing data queue. When the processor frees up at approximately 110 sec, 2 data packages are sent out to the fleet for processing at once (as evidenced by the large spike immediately after the drop). Each slave must then perform 2 update steps in succession, which accounts for the small peak observed at 115 sec in Figure 3-11. This indicates that a change to the estimator data handling functions may be necessary to ensure efficient and timely performance. As before, the FFIT testbed has illuminated an aspect of the distributed estimation architecture that would have been difficult, if not impossible, to ascertain using *à priori* analysis.

3.7 Conclusions

This chapter presented a unique testbed for implementing and testing distributed estimation and control architectures for formation flying satellites. The testbed uses serial cables to emulate actual inter-spacecraft communication and takes advantage of

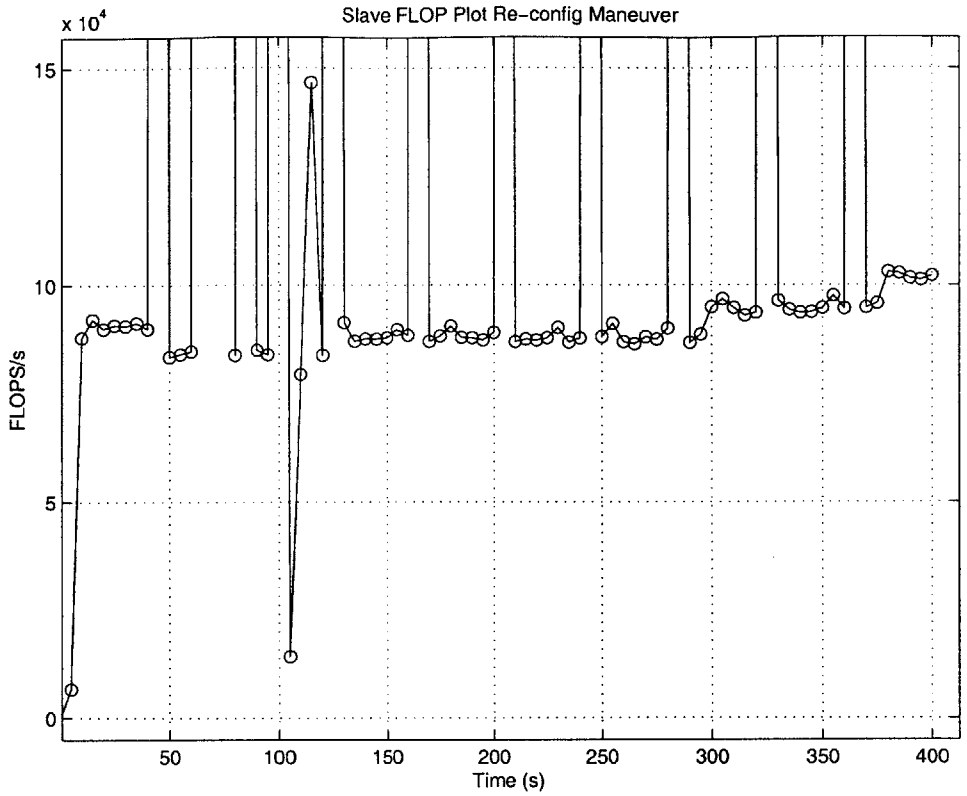


Figure 3-11: Slave FLOP profile.

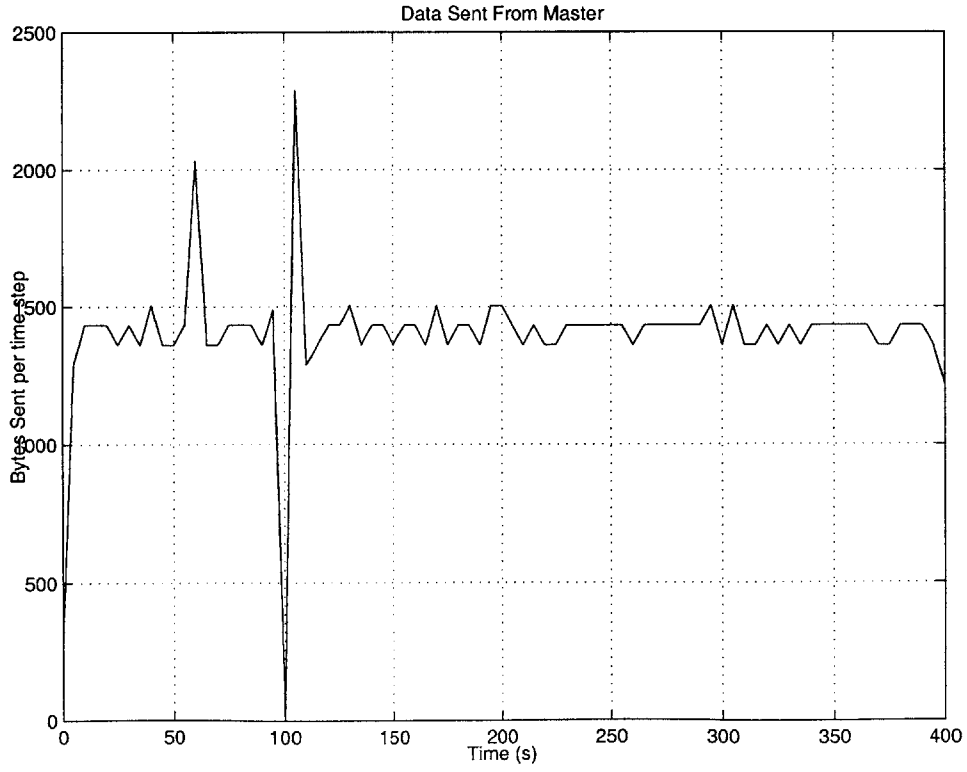


Figure 3-12: Bytes sent from Master.

the MATLAB programming environment to permit easy coding without the specific issues associated with the target operating system. Results from the testbed indicate that it can be a very useful tool for architecture evaluation and development. In the next chapter, the FFIT testbed is used to evaluate the merits of various decentralized estimation schemes when local ranging information is incorporated.

Chapter 4

Decentralized Estimation Techniques

4.1 Introduction

Chapter 3 presented the FFIT testbed for analyzing the performance of estimation and control architectures. The usefulness of the testbed was demonstrated using a single estimation and control architecture. This chapter focuses more attention on the estimation algorithms and associated architectures and considers several different decentralized estimation techniques. At the end of this chapter, the FFIT testbed is used to perform a trade study of the various proposed estimators to assess which ones are best suited to certain mission scenarios.

State estimation for a fleet of many vehicles is challenging from many perspectives. First, the raw measurement data is typically collected in a decentralized manner (*i.e.*, each vehicle takes measurements that pertain only to its own state), strongly suggesting a decentralized estimator to handle the data. Decentralized estimators are also desired to manage the large computational burden associated with large fleet estimation. Second, many of the estimation techniques commonly used are nonlinear and require the use of extended (often iterated) Kalman filters.

A commonly-used, highly accurate sensor for fleet state estimation is the Global Positioning System (GPS). Recent work on GPS estimators for relative navigation in LEO using Carrier-Phase Differential GPS (CDGPS) demonstrated a 1.0 cm accuracy in relative position and 0.5 mm/s in relative velocity [16, 18, 17]. These results were obtained using a fully decentralized filter, and the high levels of accuracy validate that

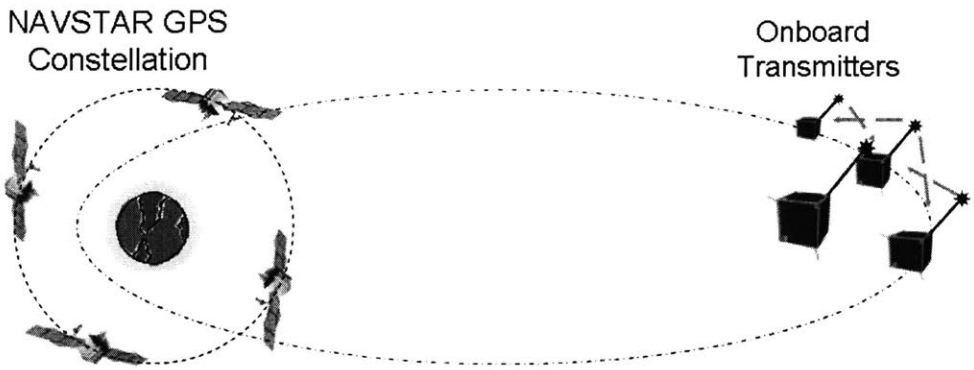


Figure 4-1: GPS Estimation with Local Ranging Augmentation. Note: Not to scale.

the relative GPS measurements (single differences relative to the master) taken on one vehicle can be treated as if they are entirely uncorrelated from the single difference measurements taken on other vehicles. Thus the full fleet measurement matrix, H, can be treated as block-diagonal and small coupling effects (such as a differential ionosphere) can be ignored if the fleet separation is less than 10 km [16, 18, 17].

While GPS can be used as an effective sensor for many ground, air and space applications, its viability relies on constant visibility of the NAVSTAR GPS constellation. For terrestrial applications, this visibility can be interrupted by buildings or trees. In space, NAVSTAR visibility begins to breakdown at high orbital altitudes such as those seen in highly elliptic or L2 orbits. Thus, a measurement augmentation is desired to permit estimation through these spells of invisibility and also to improve estimation accuracy when the NAVSTAR constellation is visible.

Figure 4-1 illustrates measurement augmentation through the use of local ranging devices on each vehicle that measure a scalar range and velocity between each pair of vehicles in the fleet [25, 26, 27, 28, 29]. Unfortunately, however, the local range measurements taken by each vehicle by definition strongly correlate the states of each vehicle, thus making the full fleet measurement matrix, H, no longer block-diagonal and non-trivial to decentralize [50]. In contrast to the GPS-only estimation scenario which effectively decentralized for reasonably sized fleet separations, this estimation problem does not decorrelate at any level. As a result, care must be taken to decentralize the estimation algorithms while retaining as much accuracy as possible and keeping the computation and inter-spacecraft communication to a minimum.

Ultimately, an estimation architecture (see Chapter 3 for definitions of the architectures studied in this Thesis) is desired that can provide accurate relative state

estimates in many different estimation regimes. However, in order to evaluate the viability of various architectures, algorithms must be developed to populate them. This chapter presents various estimation algorithms and associated architectures, developed to mesh appropriately with existing control algorithms and/or communication systems for several different types of proposed missions [6, 58, 9, 4, 7, 59]. These missions provide a wide range of formation flying scenarios including those with limited GPS visibility and extremely large numbers of vehicles. In particular, the Magnetospheric Multiscales (MMS) mission [59] will require highly elliptic orbits, causing the fleet to move in and out of NAVSTAR GPS range once an orbit. The MAXIM mission [58] will require a large number of satellites ($\approx$ 32) in a heliocentric orbit (*i.e.*, never in contact with the NAVSTAR GPS constellation).

In the following sections, several different decentralized estimation techniques will be presented. At the end of this chapter, the FFIT testbed [60] is used to perform a trade study of the proposed estimators to assess which ones are best suited to certain mission scenarios. With this data, it is anticipated that we will be in a position to make decisions on optimal estimation architectures for various mission scenarios.

4.2 Centralized Architectures

Figure 4-2 illustrates the basic algorithmic structure of a centralized algorithm on a fleet consisting of three vehicles (for clarity, vehicle number 1 will be designated the *master* and all others as *slaves*). The master gathers all measurement data for processing in a centralized filter. Depending on the fleet control requirements, solutions from the centralized filter may be sent back out to the slave vehicles for control and/or science use. The following sections present two algebraically equivalent forms of the optimal filter: The Kalman Filter and The Information Filter. Each form has its merits and disadvantages regarding communication and computation, as will be further explored in later sections.

4.2.1 Kalman Filter

The optimal filter that minimizes the mean-squared estimation error of a state vector is the well known Kalman filter given by the following equations:

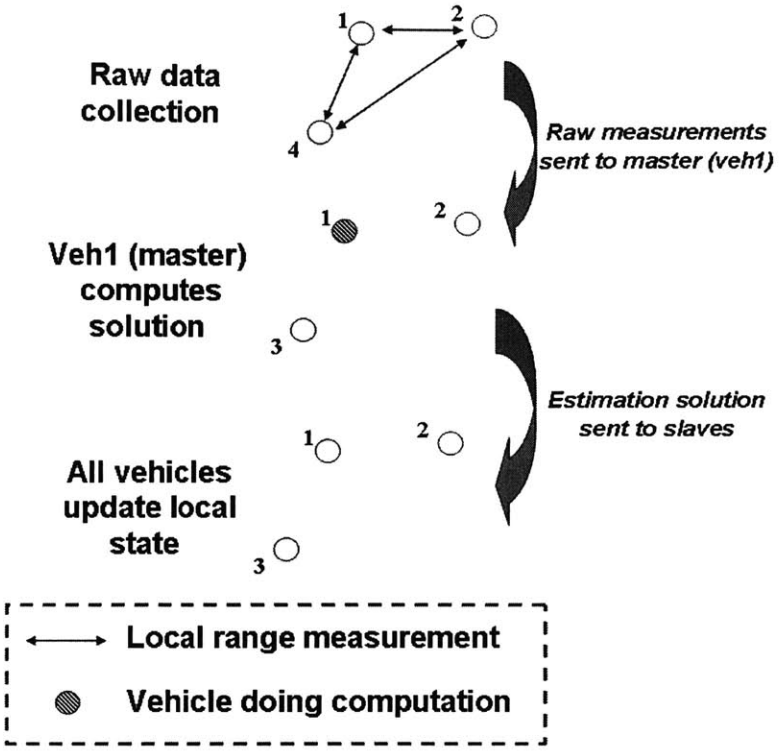


Figure 4-2: Centralized algorithmic flow.

Kalman Measurement Update

$$S_k = H_k P_k^- H_k^T + R_k \tag{4.1}$$

$$K_k = P_k^- H_k^T S_k^{-1} \tag{4.2}$$

$$\hat{X}_k^+ = \hat{X}_k^- + K_k \left(z_k - H_k \hat{X}_k^- \right) \tag{4.3}$$

$$P_k^+ = \left(\mathbf{1} - K_k H_k \right) P_k^- \tag{4.4}$$

Kalman Time Update

$$\hat{X}_{k+1}^- = \Phi_k \hat{X}_k^+ \tag{4.5}$$

$$P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + Q_k \tag{4.6}$$

For a fleet of vehicles implementing the centralized form of the Kalman filter (see Figure 4-2), each slave sends its local measurement vector, z_{k_i}, to the master at every time-step. Having received all local measurement vectors (including all GPS measurements and local ranges), the master assembles the full fleet measurement vector,

z_k and executes Eqs. 4.1-4.6. For proper integration with the control architecture, the master may need to broadcast the resulting state estimate to the slave vehicles.

4.2.2 Information Filter

The information filter is an algebraically equivalent form of the Kalman filter, cast in a light to make explicit the information content of each measurement and how it impacts the global state estimate. As described in Section 4.3.1, the information filter became the heart of research into data fusion techniques [30, 31]. The information filter definition requires the introduction of new quantities that aid in the representation of the Kalman filter from an *information* standpoint.

$$Y_k^{\pm} \triangleq P_k^{\pm^{-1}} \tag{4.7}$$

$$\hat{y}_k^{\pm} \triangleq Y_k^{\pm}\hat{X}_k^{\pm} \tag{4.8}$$

$$i_k \triangleq H_k^T R_k^{-1} z_k \tag{4.9}$$

$$I_k \triangleq H_k^T R_k^{-1} H_k \tag{4.10}$$

With these definitions, the entire information filter is summarized below.
Information Measurement Update

$$\hat{y}_k^+ = \hat{y}_k^- + i_k \tag{4.11}$$

$$Y_k^+ = Y_k^- + I_k \tag{4.12}$$

Information Time Update

$$M_k = \left(\Phi_k^{-1}\right)^T Y_k^+ \Phi_k^{-1} \tag{4.13}$$

$$Y_{k+1}^- = \left[I - M_k\left(M_k + Q_k^{-1}\right)^{-1}\right] M_k \tag{4.14}$$

$$\hat{y}_{k+1}^- = \left[I - M_k\left(M_k + Q_k^{-1}\right)^{-1}\right] \left(\Phi_k^{-1}\right)^T \hat{y}_k^+ \tag{4.15}$$

Similar to the Kalman filter, the centralized implementation of the information filter requires all information be gathered at the master for processing at every time-step. The slaves send their local information vector, i_{k_i} and matrix contributions, I_{k_i} based on their local measurements, z_{k_i} to the master. Upon receipt, the master incorporates

the new information as follows:

$$\hat{y}_k^+ = \hat{y}_k^- + \sum i_{k_i} \tag{4.16}$$

$$Y_k^+ = Y_k^- + \sum I_{k_i} \tag{4.17}$$

The master then executes Eqs. 4.13-4.15 to complete the update cycle.

4.3 Decentralized Architectures

For large fleets of vehicles, it may be desirable to evenly spread the estimation effort across the fleet. Also, since the raw measurements are gathered in a decentralized fashion (*i.e.,* each vehicle measures its own GPS and local ranging information), a decentralized estimator would prevent the transmission of measurements around the fleet. This section describes several viable decentralized estimators, split up into two classes of algorithms: *Full Order* and *Reduced Order* filters.

4.3.1 Full Order Decentralized Filters

The first class of decentralized filters is the full order filter class. A fleet running a full order decentralized estimator will have each vehicle estimating the entire fleet state.

Decentralized Information Filter

A lot of work was done in the early 1990's on the development of decentralized filters to handle large amounts of measurement data collected at remote locations. This field of study became known as *data fusion* [30, 31]. The premise was that the data could be remotely collected and then assimilated at central locations to produce estimates that are identical to centralized filter solutions.

Previous decentralized estimation research [30, 49] has relied heavily the use of information filters (see Section 4.2.2). The typical argument for why this is the case lies in Eqs. 4.11 and 4.12. Notice that the form of these measurement update equations is such that the updated quantity is simply the old quantity plus the new *information* provided by the new measurement. Thus, in order to decentralize this filter, each spacecraft sums the new information from every other node (thus requiring a *fully-connected* network). Furthermore, since the information filter is algebraically equivalent to the centralized form of the filter, no information is lost and the best

possible estimate (*i.e.,* the centralized estimate) is available on each spacecraft in the fleet. Although rarely mentioned in the data fusion literature, an equivalent decentralized form of the Kalman filter is presented by Kaminski [61]. In his work, Kaminski develops a sequential update form of the optimal Kalman filter, permitting similar decentralization characteristics to the information filter.

The decentralized information filter sounds appealing initially, and indeed it is an excellent solution for some scenarios. However, in the case of many nanosatellites estimating the fleet state, the information filter has some substantial short comings:

1. While the measurement update is considerably simpler than the conventional Kalman filter, the propagation step is much more complicated.
2. The information filter deals strictly in information variables that have little significance to the actual problem. In order to obtain the actual state estimate and its associated covariance, one must solve Eqs. 4.7 and 4.8 together to back out the state estimate and covariance matrix.
3. A fully connected network must exist to trade the information data from each vehicle to every other vehicle in the fleet. Furthermore, this data can potentially be quite large since an information vector and information matrix (see Eqs. 4.8 and 4.7) must be sent for each update[1].

Note that it has previously been suggested [62] that, depending on the scenario, it may be possible for each vehicle in the fleet to determine the measurement matrix, H, for each vehicle thus eliminating the need to transmit the information matrix at each time step. But this does not work well for nonlinear filters because a discrepancy arises from the fact that the linearized H matrix must be obtained using the state estimate to generate the information components I and i in Eqs. 4.9 and 4.10. If each vehicle has different estimates of the fleet state, then small errors (due to second order variations in $h(\hat{X})$) will accumulate through the additive updates of Eqs. 4.11 and 4.12. To consider this point in more detail, denote the state estimate of the master vehicle as $\hat{X}_1$ then its linearized measurement matrix takes the form $H(\hat{X}_1)$. If we assume that the slave vehicle has a slightly different state estimate ($\hat{X}_2 = \hat{X}_1 + \Delta X$), then the linearized measurement matrix for the slave vehicle is

$$H(\hat{X}_2) \approx H(\hat{X}_1) + \left.\frac{\partial H(X)}{\partial X}\right|_{\hat{X}} \Delta X \tag{4.18}$$

[1]Note that only half of the information matrix need be sent since it is symmetric.

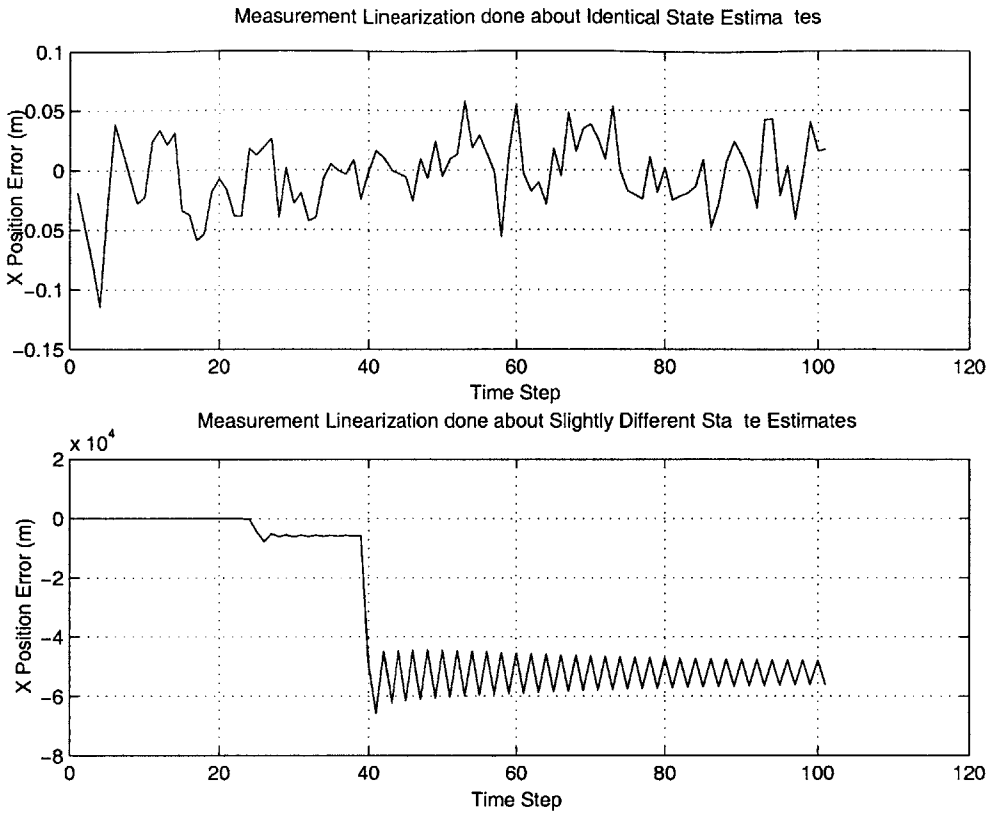


Figure 4-3: The effect of measurement equation linearization about slightly different state estimates.

The second term in Eq. 4.18 creates a difference in the estimates that can accumulate over time.

The result of this second order effect is demonstrated in Figure 4-3. In this 2-vehicle simulation, one vehicle computes i_i as per Eq. 4.9 using an H that was linearized about the slave's current state estimate. The slave then transmits i_k to the master vehicle. To perform the update step, the master must compute the I_k for the slave vehicle based on an H that was linearized about the *master's* current state estimate. In this scenario, the master's state estimate and the slave's state estimate differ by less than a centimeter. Figure 4-3 demonstrates the filter going unstable at approximately the 23rd time-step. Clearly, this performance is unacceptable. To prevent this instability, the vehicles must transmit both the information vector *and* the information matrix at every time step (requiring a substantial increase in communication).

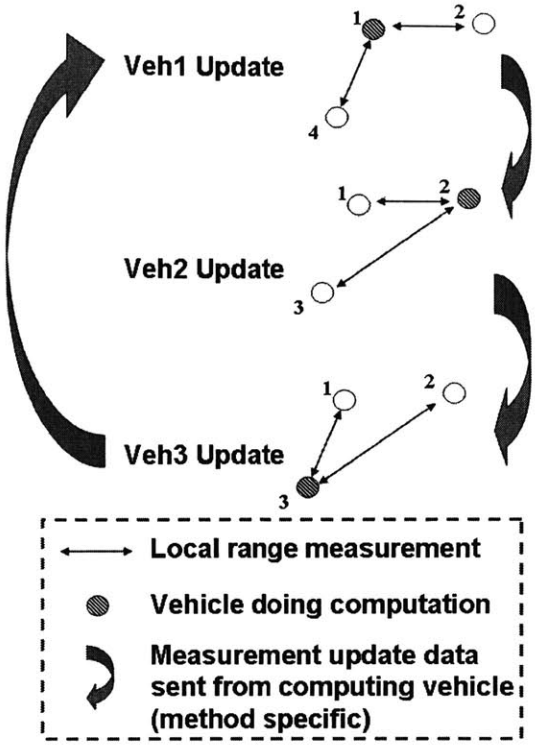


Figure 4-4: Cascade algorithmic flow.

Full-Order Iterative Cascade Filter

Another filter considered is the full-order iterative cascade filter. In this algorithm (depicted pictorially in Figure 4-4) each vehicle employs a standard Kalman filter estimating the full fleet state, but using only the locally available measurements. After the measurement update, each vehicle broadcasts its local state solution to every other vehicle. Upon receipt, the vehicles re-compute their measurement matrices, H_i and re-compute a measurement update. This update procedure is identical to that proposed by Park [26, 50], except that in this case, each vehicle is estimating the entire fleet state, whereas in Park's filter, each vehicle estimated only their local state (see Section 4.3.2 for more details).

Note that the full-order iterative cascade filter is sub-optimal because there is no single filter in the fleet that can accurately estimate the inter-vehicle correlations (the correlations are approximated by iterations around the fleet). Furthermore, given the amount of computational effort that must be exerted to execute a full-order filter, this method may not provide a good balance between estimation accuracy and computational effort (see Section 4.5.3).

4.3.2 Reduced-Order Decentralized Filters

For many control and science applications, each vehicle is primarily interested in estimating its own local state. While every vehicle needs some estimate of the fleet state in order to linearize its measurement equation, it is not clear that much effort should be spent on estimating other vehicles' states if it can be avoided. This prompted the next type of decentralized filter, the *reduced order decentralized filters.* In the following algorithms, each vehicle estimates only its local state, thus substantially reducing the computational demands on each vehicle. The price paid for this reduction in computation is that the estimator is suboptimal. The extent to which this suboptimality affects estimator performance is examined in Section 4.5 when each method is compared in detailed analyses and simulations. The algorithmic flow for all of the following reduced-order decentralized filters is identical to that shown in Figure 4-4.

Reduced-Order Iterative Cascade Filter

Recent work by Park resulted in a reduced-order decentralized filter known as the Iterative Cascade Extended Kalman Filter (ICEKF) [26, 50]. In this estimation technique, the first vehicle runs a local filter, solving approximately for its own local state vector (*i.e.,* not the entire fleet relative state vector) based on the latest estimate of the entire fleet state and sends this result to the next vehicle in the fleet. This vehicle updates its copy of the fleet relative state vector and runs its own local filter to obtain an approximate estimate of its associated local state and sends this information onto the next vehicle. This process continues with each vehicle updating its local state vector in sequence, iterating through the entire fleet until the changes to the fleet state are small. Once the measurement iterations have terminated, each vehicle independently performs a measurement update on its local covariance matrix using the H and K that the iterations converged to. Following the P update, each vehicle performs a state and covariance propagation step and then sends the updated local state vector out to the fleet[2].

The purpose of the ICEKF was to provide a decentralized local ranging augmentation for LEO applications using GPS pseudolites. Furthermore, Park's algorithm was intended for a relatively small number of range measurements (compared to the

[2]This same iteration technique was used in the *full order iterative cascade filter* in Section 4.3.1

NAVSTAR GPS measurements) with identical accuracy to the other GPS measurements. In this environment, the ICEKF works extremely well and Park shows in Refs. [26, 50] that near optimal performance can be achieved. For this Thesis, we are interested not only in local ranging for LEO augmentation, but also for MEO and beyond. For these scenarios, the estimator will have to rely almost solely on the local ranging data that could be much more accurate than the GPS measurements. In this scenario, the ICEKF exhibits poor performance (see Section 4.5 for simulation results), typically taking 4-5 iterations through the entire fleet before the solution converges. As a result, this process is complicated, communication intensive, and often yields unstable results.

The primary problem with the ICEKF is that the relative state vectors from every other vehicle are assumed to be perfect, when in reality, these states are in the process of being estimated and thus are erroneous. One *ad hoc* method for accounting for the estimation error associated with other vehicles' states is to absorb it into the measurement error variance matrix R. Before starting the estimator, R is increased ("bumped up") by a constant amount that corresponds to the other vehicles' estimated error

$$R_{\text{bump}} = R + JP_{\text{yy}}J^T \tag{4.19}$$

where R is the original measurement error variance matrix, J is the measurement matrix for all other non-local measurements in the fleet and P_{yy} is the initial covariance matrix for all other non-local states in the fleet state vector. Increasing R indicates that the measurements might not be as good as suggested by the accuracy of the ranging device.

Another possible technique for reducing the number of fleet iterations and possibly improving stability and accuracy is to transmit some state covariance information along with the local state vector. Section 4.3.2 describes a technique developed to incorporate this state uncertainty information based on the theory behind the Schmidt-Kalman filter.

Schmidt-Kalman Filter

The traditional purpose of the *Schmidt-Kalman filter* (SKF) [32] is to reduce the computational complexity of a standard Kalman filter by eliminating states that are of no physical interest, but required for estimation of noise and/or biases. The

formulation of the Schmidt-Kalman filter begins with a partitioning of the standard state propagation and measurement equations as well as the covariance matrix as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix}_{k+1} = \begin{bmatrix} \phi_x & 0 \\ 0 & \phi_y \end{bmatrix}_k \begin{bmatrix} x \\ y \end{bmatrix}_k + \begin{bmatrix} w_x \\ w_y \end{bmatrix}_k \tag{4.20}$$

$$z_k = \begin{bmatrix} H & J \end{bmatrix}_k \begin{bmatrix} x \\ y \end{bmatrix}_k + \nu_k \tag{4.21}$$

$$P_k = \begin{bmatrix} P_{xx} & P_{yx} \\ P_{xy} & P_{yy} \end{bmatrix}_k \tag{4.22}$$

where x represents the state vector containing the states of interest and y represents the remaining states. After applying the partitions of Eqs. 4.20 and 4.21 to the general Kalman filter equations, solving for each block and *setting the gain for the y states to zero,* the following equations result[3]:

Schmidt-Kalman Measurement Update

$$\alpha_k = H_k P^-_{xx_k} H_k^T + H_k P^-_{xy_k} J_k^T + J_k P^-_{yx_k} H_k^T + J_k P^-_{yy_k} J_k^T + R_k \tag{4.23}$$

$$K_k = (P^-_{xx_k} H_k^T + P^-_{xy_k} J^T)\alpha_k^{-1} \tag{4.24}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^- - J_k\hat{y}_0) \tag{4.25}$$

$$P^+_{xx_k} = (I - K_k H_k)P^-_{xx_k} - K_k J_k P^-_{yx_k} \tag{4.26}$$

$$P^+_{xy_k} = (I - K_k H_k)P^-_{xy_k} - K_k J_k P^-_{yy_k} \tag{4.27}$$

$$P^+_{yx_k} = P^{-T}_{xy_{k+1}} \tag{4.28}$$

$$P^+_{yy_k} = P^-_{yy_k} \tag{4.29}$$

Schmidt-Kalman Time Update

$$\hat{x}^-_{k+1} = \phi_{x_k}\hat{x}_k^+ \tag{4.30}$$

$$P^-_{xx_{k+1}} = \phi_{x_k} P^+_{xx_k}\phi^T_{x_k} + Q_{x_k} \tag{4.31}$$

$$P^-_{xy_{k+1}} = \phi_{x_k} P^+_{xy_k}\phi^T_{y_k} \tag{4.32}$$

$$P^-_{yx_{k+1}} = P^{-T}_{xy_{k+1}} \tag{4.33}$$

[3] Note that in Eq. 4.25, the $J_k\hat{y}_0$ term is only required if the y states are thought to have a non-zero mean. If this is the case, a best guess of the $\hat{y}$ states is inserted for $\hat{y}_0$. In many applications, the best guess of $\hat{y}_0$ might be 0.

$$P^-_{yy_{k+1}} = \phi_{y_k} P^+_{yy_k} \phi^T_{y_k} + Q_{y_k} \tag{4.34}$$

The SKF equations may appear more complicated than those of the traditional Kalman filter; however, substantial computational savings are embedded in the fact that the filter only solves for the x state and not the y state which, in typical applications of this technique, is of no physical interest. It is this aspect of the Schmidt-Kalman filter that is appealing for the design of reduced-order decentralized filters. In particular, a Schmidt-Kalman filter is used here to incorporate a covariance estimate of the relative state estimates, which eliminates the prior assumption that the states of all other vehicles are known perfectly.

This method introduces the relative states of all fleet spacecraft into the state vector, and then uses a Schmidt formulation to incorporate the uncertainty in other vehicles' states, while reducing the estimated state vector to include only the local state. Each vehicle follows the same procedure given for the ICEKF, but instead of transmitting just its local state vector to the next vehicle in the fleet, it sends the local state vector along with its local covariance matrix (or some representative portion of its local covariance matrix - see section 4.5.3 for details). Also, instead of executing the standard Kalman filter equations, the Schmidt-Kalman filter equations are used replacing P_{yy} with the transmitted covariance matrices of each other vehicle placed on a block-diagonal. Thus, with P_{yy} being transmitted to each vehicle at every time-step, Eqs. 4.29 and 4.34 are omitted from the standard Schmidt formulation.

To understand how this method accounts for uncertainty in the other vehicles' state estimates, note that Eq. 4.23 is simply Eq.4.1 of the traditional Kalman filter with three additional terms, $H_k P^-_{xy_k} J^T_k$, $J_k P^-_{yx_k} H^T_k$ and $J_k P^-_{yy_k} J^T_k$. Thus one could regard the SKF as a "dynamic Bump Up R" method. Instead of adding a constant amount to R (as described in a possible modification to the ICEKF in Section 4.3.2), the SKF chooses how much to bump up R at every time-step.

4.4 Hierarchic Clustering

The reduced-order methods presented in the previous sections provide a viable solution to the fleet navigation problem for medium-sized fleets. For larger fleets, the iteration techniques will become very difficult (due to vehicle synchronization) and an alternative architecture will be required. One approach to mitigate this scaling

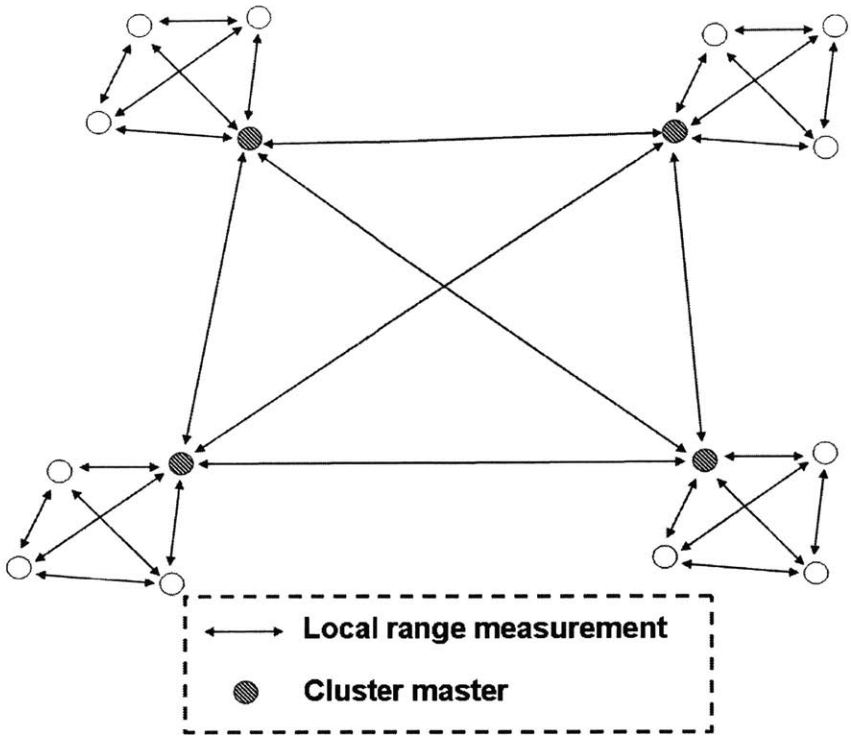


Figure 4-5: Hierarchic Clustering Topology.

problem, is to employ a hierarchic architecture, which performs the detailed estimation for smaller groups of vehicles and then assimilates partial results at a higher level. Figure 4-5 illustrates one strategy for setting up a hierarchic architecture for spacecraft employing local ranging. The fleet is split up into smaller clusters that perform their ranging and navigation independently with the exception of one vehicle in each cluster termed the "cluster master". To link the estimates of each cluster to one another, each cluster master joins together to form a "super-cluster", so the hierarchy looks essentially identical at each level. The key benefit of this approach is that the clusters and super-clusters do not need to be tightly synchronized, or even run at the same update rate.

The type of filter for the cluster estimators is chosen based on the cluster sizes, available communication bandwidth, CPU loading and required accuracy. Depending upon the number of layers in the architecture (only two are shown in Figure 4-5), determining a vehicle's position with respect to the fleet center may involve vector additions of several different cluster solutions. Since each vector addition step introduces another source of error (due to summing two quantities, each with an uncertainty),

Table 4.1: Error scaling through hierarchy with σ^2 as basic estimator position error variance. Variances add every time a level is traversed. For instance, the position error variance from one local slave to another is $2\sigma^2$ since the vector position of each slave with respect to their local cluster master must be added.

Estimate	Error Variance
Slave to Local Master	σ^2
Slave to Local Slave	$2\sigma^2$
Local Master to Local Master	σ^2
Local Master to Fleet Center	σ^2
Slave to Fleet Center	$2\sigma^2$
Slave to Remote Master	$3\sigma^2$
Slave to Remote Slave	$4\sigma^2$

from a performance perspective, larger clusters are more desirable than smaller ones. Assuming a two-layer hierarchy, and each cluster running identical estimators, the growth of estimation error variance is shown in Table 4.1.

Cluster sizing and selection could be done based on several different criteria, including geographic separation, common GPS visibility or even existing communication connectivity from science experiments. If the fleet communication architecture permits, the clusters could be dynamic. From an estimation standpoint, the best clustering approach would be to form clusters of vehicles that are ranging with each other; however, inter-cluster ranging could be permitted provided enough state information is exchanged to formulate the measurements. For example, if a vehicle in one cluster wants to range with another vehicle in different cluster, information must be traded regarding the other cluster's global position (*i.e.,* must communicate with the cluster masters). However, if inter-cluster ranging is limited to only cluster masters (*i.e.,* cluster slaves can only do inter-cluster ranging with masters of other clusters), then the only extra information required is known by the local cluster master and thus substantially reduces data trade requirement. Performing inter-cluster ranging is a good way to reduce the error growth illustrated in Table 4.1 since error variances associated with positions between ranging vehicles is at most $2\sigma^2$.

Using hierarchic clusters such as these permit much flexibility in terms of estimation algorithms. Any of the above-mentioned algorithms could be used at any level of the hierarchy. Choosing the most appropriate algorithms to populate the hierarchic

levels depends upon the required estimation accuracy, and available computation and communication bandwidth. Section 4.5.3 explores this type of algorithm selection in more detail.

4.5 Analysis

The following sections present results from several different types of analyses. The first analysis studies the steady state filter covariances of each method. Next, to verify the predicted errors, long, multiply averaged simulations are carried out for each method to obtain representative accuracy estimates. Finally, results are presented from tests run on the FFIT testbed. The results from these tests indicate the overall performance of each method from an estimation accuracy/communication/computation point of view.

The intent of these analyses is to gain insight into the feasibility of the proposed algorithms and architectures in this chapter. Thus, the scenarios studied have been simplified in the following ways:

- All dynamics are limited to two dimensions.
- GPS and local range measurements are treated as x, y positions and scalar ranges respectively without the need for estimating time.

These simplifications remove the sensor and implementation-specific complexities from the problem while retaining the most important aspect of this study - the non-linearity of the local ranging measurements that couple the vehicles states. While the results of these analyses will lead to a better understanding of the relative merits of each algorithm, an important next step will be to apply these algorithms to the true mission-specific scenarios expected for future formation flying missions.

4.5.1 Two-Vehicle, Two Dimensional Covariance Analysis

In order to verify and test the effectiveness of the SKF and other filtering methods, several different types of analyses were conducted. The focus of the first study was to observe the structure of the filter covariance matrix in each filter. Constant probability contours are a useful visualization technique for studying covariance matrices. In the $2x2$ case, a covariance matrix can be represented as a rotated ellipse in the

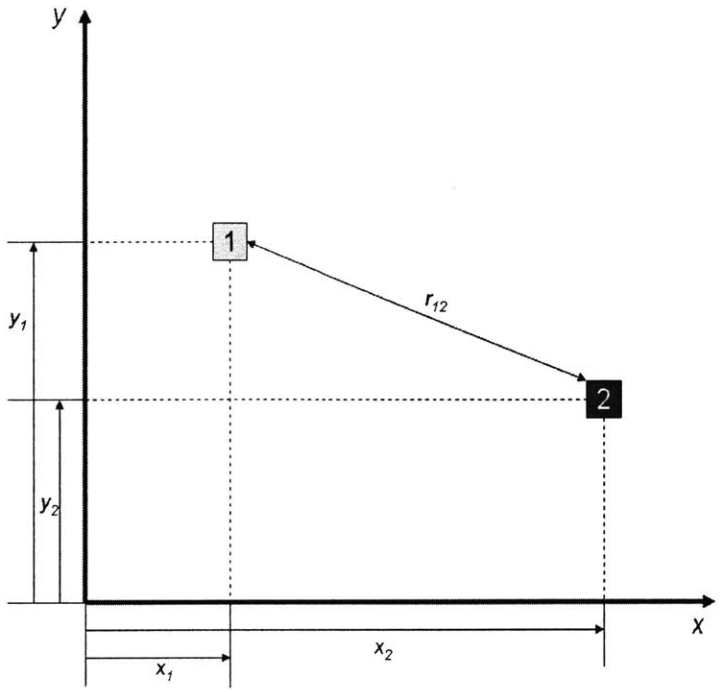


Figure 4-6: Two Dimensional Problem Geometry

Cartesian plane:

$$P = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \Rightarrow \begin{cases} a & = & \sigma_1 & (\text{semi} - \text{major axis}) \\ b & = & \sigma_2 & (\text{semi} - \text{minor axis}) \\ \tan(2\theta) & = & \frac{2\rho_{12}\sigma_1\sigma_2}{\sigma_1^2 - \sigma_2^2} & (\text{rotation angle}) \end{cases} \tag{4.35}$$

Figure 4-6 illustrates the problem geometry for this analysis. For this scenario, each vehicle receives an x and y measurement for their own position as well as a measurement of the range between the vehicles, r_{12}. The algebraic Riccati equation is solved for each filter to obtain the steady state covariance matrix. Figure 4-7 shows the error ellipses for vehicle 1 for each type of centralized and decentralized filter described above (note that since the Kalman filter and the Information filter provide identical covariance results, only one ellipse is shown to represent both filters). This type of analysis provides insight into the degree to which the measurements are being used. For example, a narrow ellipse aligned with the x-axis would indicate that the filter had been able to make very good use of a measurement in the y-direction (resulting in better confidence and hence a narrower ellipse, in the y-direction).

The ellipse representing the case with no local ranging is a circle with a radius larger than any of the semi-major axes of the other ellipses. The circular shape is ex-

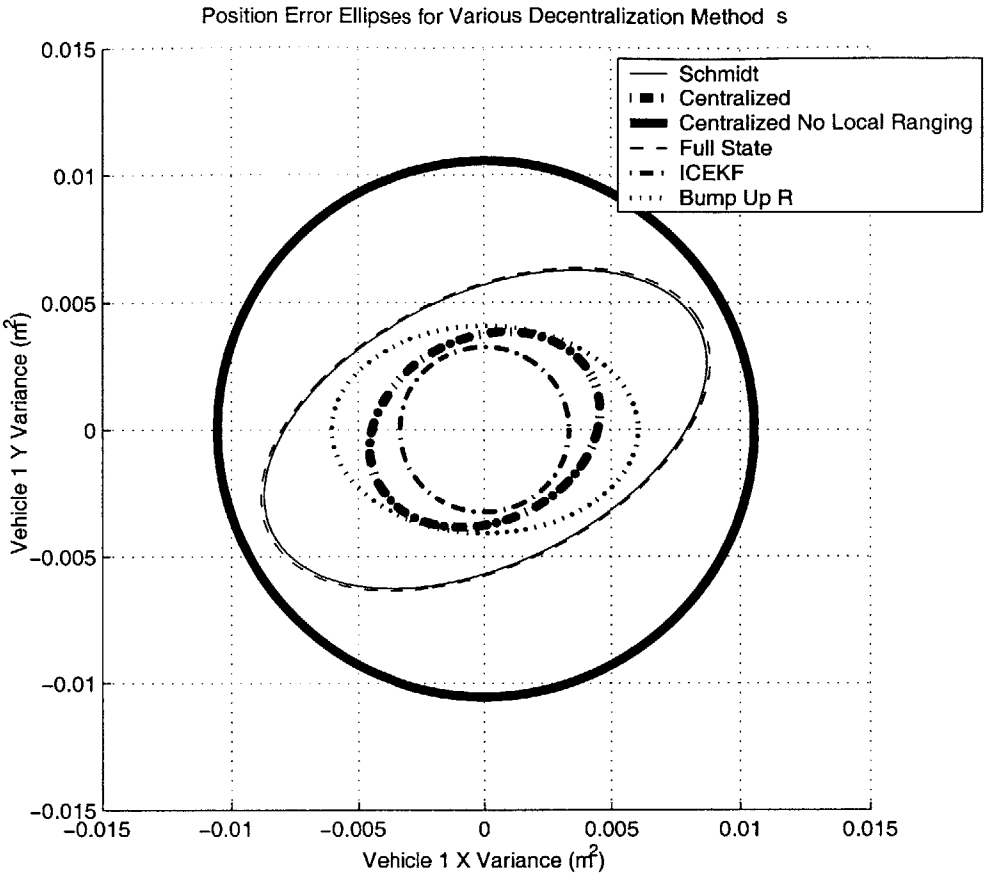


Figure 4-7: Error ellipses for various decentralization methods (2D Case)

pected due to the equal measurement accuracy in the linear positional measurements of x and y. The ellipse for the centralized filter represents the *best case* possible since the centralized method captures all vehicle and state correlation in a single, unified filter. Both the *Bump Up R* method and the *ICEKF* method appear to have covariance matrices that are relatively close to that of the centralized method, suggesting near-centralized performance, but this is slightly misleading because the filter covariance for the *Bump Up R* and *ICEKF* methods do not provide good figures of merit, as outlined in the following.

In a full order, centralized Kalman filter, the measurement equation is

$$z = CX \tag{4.36}$$

where X denotes the entire fleet state and z denotes the full complement of measurements available in the fleet. This equation can be expanded into terms that

correspond to the local states x and the states of the other vehicles y.

$$z = CX = Hx + Jy \tag{4.37}$$

Note that this partitioning is identical that used to derive the Schmidt-Kalman filter. The ICEKF method is a reduced-order estimator so it only has the local state available to use in the measurement equation. So its measurement equation is given by the approximation

$$z_i \approx H_i x \tag{4.38}$$

where z_i is the vector of measurements available on a local vehicle. The term Jy is completely omitted from the ICEKF equations; however, the regular Kalman filter equations derived assuming an optimal K with correct values of R, Q, Φ and H are still used. This means that the filter covariance cannot be trusted as the true error covariance and is thus, not a good indication of the filter performance. Furthermore, the fact that the ICEKF covariance ellipse is smaller than the centralized ellipse (*i.e.*, the *true* covariance[4]) indicates that the ICEKF method is trusting the measurements too much. Similarly, the *Bump Up R* covariance cannot be trusted either since here, not only is the H incorrect, but the R is incorrect as well.

The filter covariance of the full state decentralized estimator, however, can be trusted since it uses the full H matrix and the correct R value. Notice that in one direction, it provides only a minimal improvement over the *No Local Ranging* case. Also, since this filter attempts to estimate many more states than are observable, the covariance corresponding to the states not pertaining to the local vehicle will tend to grow without bound (depending upon the dynamics, Φ). This could lead to an ill-conditioned P matrix and cause numerical problems over time if not adequately accounted for.

The Schmidt-Kalman filter uses a sub-optimal K but the filter covariance *can* be trusted. Recall that the SKF derivation starts with the general update equation (Eq. 4.39) for any (optimal or sub-optimal) P.

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \tag{4.39}$$

[4]Actually, since these filters are Extended Kalman Filters, the *true* error covariance is not accessible. However, the covariance error introduced in using an EKF formulation over that introduced by using an incomplete measurement matrix can be shown to be small if the measurement linearization works sufficiently well.

Using this equation, the true covariance is recovered and hence, the SKF covariance can be trusted. The ellipse for the SKF method appears only slightly better than the full state case. This is somewhat surprising based on the correction that the SKF attempts, but the SKF does give a reduced computational load (see Section 4.5.3). Also the SKF does not suffer from the ill-conditioned P matrices that are typically experienced with the full state filters.

The preceding analysis has provided some valuable insight into the expected performance of each estimator. These results indicate that filters such as the centralized and SKF methods should perform well since the filter's conception of the error covariance is accurate. Methods such as the ICEKF are not expected to perform well since they seem to be over-emphasizing the usefulness of the measurements as evidenced by small covariances. Furthermore, being able to trust the filter's covariance provides valuable sensor integrity information which could be used in fault detection routines (e.g. more ranging partners may be required if the filter covariance becomes too high).

4.5.2 Multi-Vehicle Simulation Results

To obtain more detailed performance comparisons, several sets of multi-vehicle two-dimensional simulations were conducted over several orbits that demonstrate the relative effectiveness of the various filters presented in this chapter. These two-dimensional simulations were done to provide additional insight on the performance, computation, and communication requirements of the fleet estimation. The two-dimensional results in this chapter are directly relevant to several future missions that plan to employ large arrays of collecting vehicles arranged in very "flat" configurations [63, 58].

The presented results are averaged over each vehicle in the fleet and then over all runs with the same geometry but different random error sets. All standard deviation errors are presented as root sum squared errors (RSS) to combine all directional errors into one number.

The first set of simulations presented are taken from a five-vehicle fleet consisting of one master vehicle and four slave vehicles. The goal of each slave vehicle is to estimate as accurately as possible its relative two-dimensional state with respect to the master vehicle. Thus, the full state vector for the relative fleet estimation problem

with N slaves and one master is:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \tag{4.40}$$

where

$$X_i = \begin{bmatrix} x_i & y_i & \dot{x}_i & \dot{y}_i \end{bmatrix}^T \tag{4.41}$$

is the local state vector pertaining to the ith slave. Since only relative states are of interest for this particular simulation, the absolute state of the master is taken arbitrarily to be at the origin. Initially, the four slaves are located one on each axis some specified distance from the origin. Each slave follows different circles of constant radii and angular speeds around the master as depicted in Figure 4-8. Vehicle positions such as this provide good fleet geometry (PDOP) for utilizing the local ranging measurements most effectively. Process noise was added that results in slight perturbations on each orbit around the master (as can be seen on the "Slave 1" trajectory).

For estimation purposes, there are two types of measurements available. The first type are coarse measurements of x and y relative position and relative velocity pairs with respect to the master. These measurements are intended to depict the results from some basic navigation system (*i.e.*, NAVSTAR-only GPS). The second type are fine local ranging measurements. These measurements are performed between each pair of vehicles in the fleet and provide scalar ranges and velocities of a higher degree of accuracy than the coarse measurements. Each simulation run has three measurement phases. Phase 1 lasts 500 seconds and the only measurements available to each vehicle are the coarse position and velocity measures. Phase 2 is also 500 seconds long and adds local ranging measurements, but only between the master and slaves (not between slaves). Phase 3 lasts 1000 seconds and includes the full complement of coarse and fine measurements including local ranging between slaves.

When all local ranging measurements are used, the full fleet measurement vector takes on the following form:

$$Z = h(X) = \begin{bmatrix} X \\ D(X) \\ \dot{D}(X) \end{bmatrix} \tag{4.42}$$

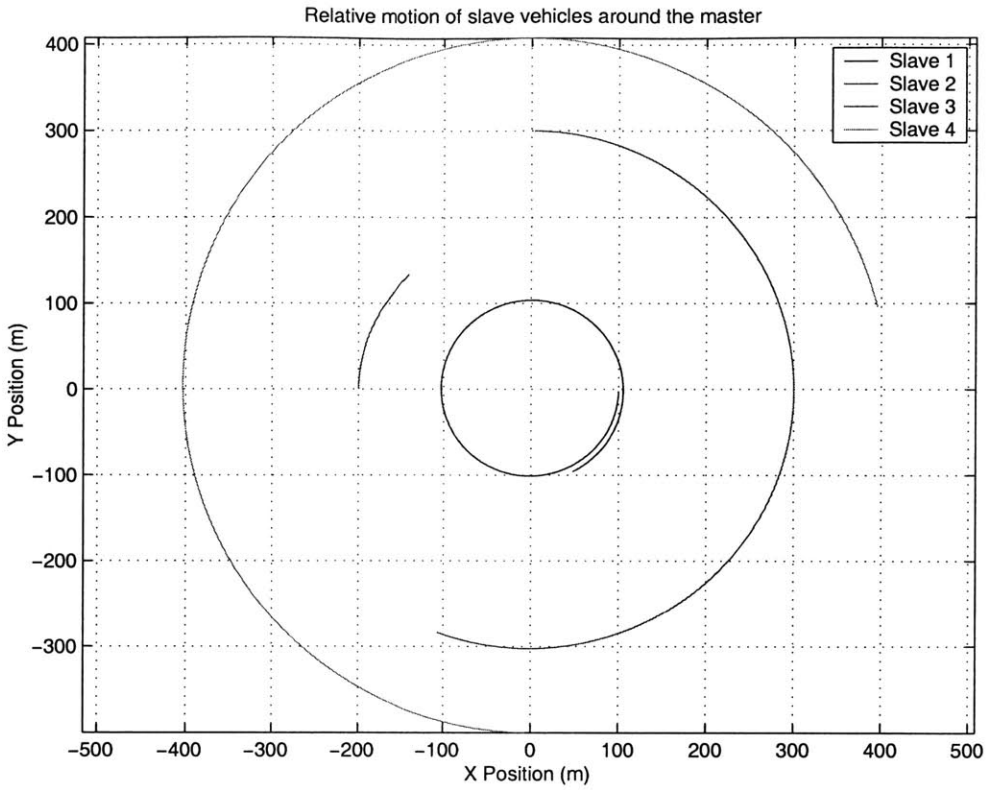


Figure 4-8: Vehicle motion for two-dimensional simulation.

where

$$D(X) = \begin{bmatrix} r_{1m}(X_1) \\ r_{2m}(X_2) \\ \vdots \\ r_{ij}(X_i, X_j) \\ \vdots \\ r_{N-1,N}(X_N, X_{N-1}) \end{bmatrix} \text{ are the } \frac{N(N-1)}{2} \text{local ranges}$$

$$\dot{D}(X) = \begin{bmatrix} \dot{r}_{1m}(X_1) \\ \dot{r}_{2m}(X_2) \\ \vdots \\ \dot{r}_{ij}(X_i, X_j) \\ \vdots \\ \dot{r}_{N-1,N}(X_N, X_{N-1}) \end{bmatrix} \text{ are the } \frac{N(N-1)}{2} \text{ local range rates}$$

$x_i,\ y_i$ = relative position measurements for vehicle i

$\dot{x}_i,\ \dot{y}_i$ = relative coarse velocity measurements for vehicle i

$r_{i,m}$ = fine local range measurement from vehicle i to master

$\dot{r}_{i,m}$ = fine local range rate measurement from vehicle i to master

$r_{i,j}$ = fine local range measurement from vehicle i to j

$\dot{r}_{i,j}$ = fine local range rate measurement from vehicle i to j

The nonlinearity in Eq. 4.42 results from the range and range rate measurements that are defined as:

$$r_{i,m} = \sqrt{x_i^2 + y_i^2} \tag{4.43}$$

$$\dot{r}_{i,m} = (x_i\dot{x}_i + y_i\dot{y}_i)/r_{i,m} \tag{4.44}$$

$$r_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.45}$$

$$\dot{r}_{i,j} = \left((x_j - x_i)(\dot{x}_j - \dot{x}_i) + (y_j - y_i)(\dot{y}_j - \dot{y}_i)\right)/r_{i,j} \tag{4.46}$$

Linearizing this relationship, for the decentralized filters (*i.e.*, with a reduced measurement set) gives

$$h(X) \approx HX \tag{4.47}$$

where

$$H = \begin{bmatrix} I & 0 \\ 0 & I \\ \left.\frac{\partial D(X)}{\partial X}\right|_{\hat{X}} & 0 \\ \left.\frac{\partial \dot{D}(X)}{\partial X}\right|_{\hat{X}} & \left.\frac{\partial \dot{D}(X)}{\partial X}\right|_{\hat{X}} \end{bmatrix} \tag{4.48}$$

In the decentralized schemes, it is assumed that each vehicle only has available to it measurements that pertain to itself. For instance, the ith slave's measurement vector would take on the form:

$$h_i(X) = \begin{bmatrix} X_i^T & r_{i,m} & \dot{r}_{i,m} & r_{i,1} & \dot{r}_{i,1} & \cdots \\ & \cdots & r_{i,(i-1)} & \dot{r}_{i,(i-1)} & r_{i,(i+1)} & \dot{r}_{i,(i+1)} & \cdots & r_{i,N} & \dot{r}_{i,N} \end{bmatrix}^T \tag{4.49}$$

Figure 4-9 shows the results (RMS position errors for each vehicle) of the simulation during each phase. The results summarized in this chart are averaged over 10 different measurement sets. The methods tested include (in order from left to right) ICEKF, 4 different types of SKF, Bump-Up-R, Full State and Centralized.

The following describes the differences between the 4 SKF methods tested:

- **Schmidt Full Covariance** - The full P_{xx} for each vehicle is exchanged at each time-step to be used in each other vehicles' P_{yy} (as per the algorithm presented in Section 4.3.2).
- **Schmidt Diagonal Covariance** - This approach saves some inter-spacecraft communication by transmitting only the diagonal of each vehicle's P_{xx} to form the P_{yy} matrix.
- **Schmidt Max Diagonal Covariance** - This method is similar to the previous method, but only the maximum diagonal element of each vehicles' P_{xx} is transmitted. To form the P_{yy} matrix, each vehicle multiplies this maximum diagonal element by the identity matrix.
- **Schmidt Own Covariance** - In this final SKF method, each vehicle assumes that its own covariance matrix, P_{xx}, is similar to the covariance matrices of all other vehicles in the fleet. So, nothing is transmitted and each vehicle simply uses its own P_{xx} to form P_{yy}.

Figure 4-9 compares the performance of each decentralized method to that of the centralized method. Since it uses all measurements in a single filter update, the centralized method is able to capture all of the available information and thus produce the best possible answer given the problem geometry and noise characteristics. The three clusters of bars in the charts represent the three different phases in the estimation process. In the first phase, no local ranging measurements are used; this phase is akin to the NAVSTAR-only GPS estimation scenario. As discussed earlier, this problem cleanly decentralizes because the measurement matrix, H, is block diagonal[5]. Hence, each method of decentralization performs equally well and are exactly as accurate as the centralized method. A similar argument can be made for the case where each slave ranges only with the master. Again, the H matrix has a block-diagonal structure that decentralizes cleanly. As would be expected, all decentralization methods produce the same accuracy as the centralized. It is also clear from the charts the effect that the local ranging has on the best achievable accuracy.

In the third cluster, each vehicle uses the full complement of local ranging available to it. Here, the H matrix is no longer block diagonal and the differences between

[5]In this case, the measurements representing NAVSTAR GPS measurements are simply a measurement of the local state vector, so the H matrix is actually identity.

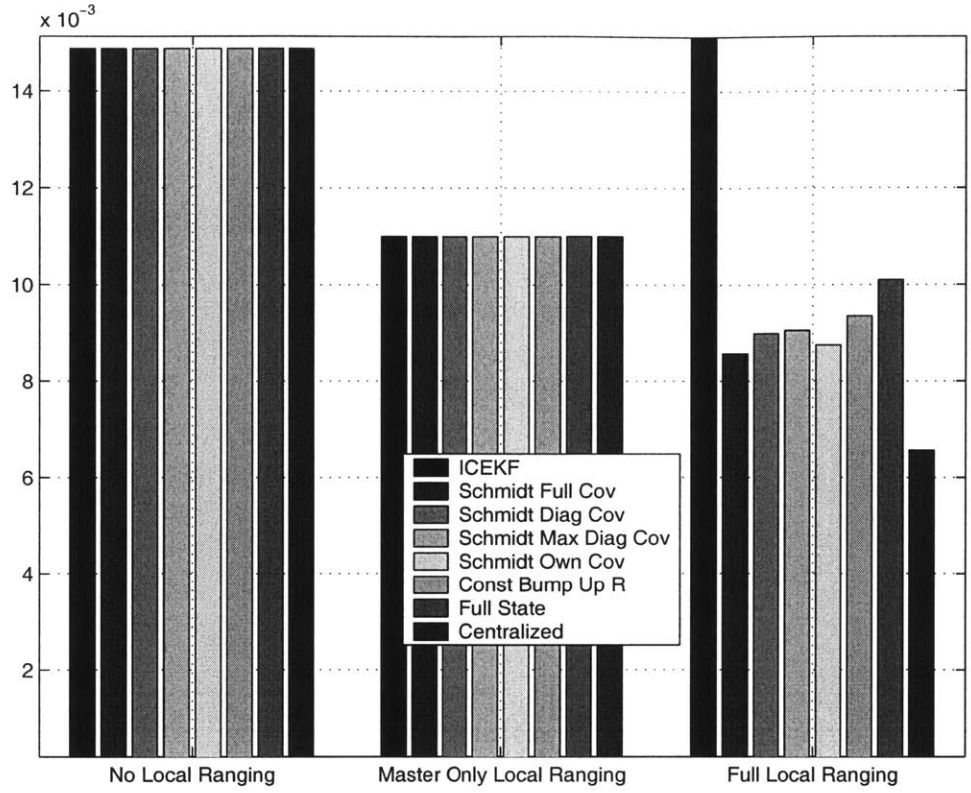


Figure 4-9: Bar chart illustrating position estimation error for various decentralization methods (ICEKF bar is beyond top of chart).

the decentralization methods become evident. In particular, the poor results of the ICEKF with no correction (and only 1 iteration) are readily apparent. Increasing the number of iterations can improve this result, but the fundamental problem is that the approach uses the measurements too aggressively. Of the methods tested, the SKF method with full covariance transmission performs the best, second only to the centralized method. As might be expected, the methods that perform the best require the most inter-spacecraft communication. This plot shows that the more covariance information that is traded, the better the achieved performance. However, it is also clear that this improvement (at least for these 2-D tests) is not dramatic. Another interesting observation is how well the "Schmidt Own Covariance" case performs. Due to similar geometries in two dimensions, it is likely that the covariances look similar for each vehicle, providing accuracies almost as high as the "Schmidt Full Covariance" case.

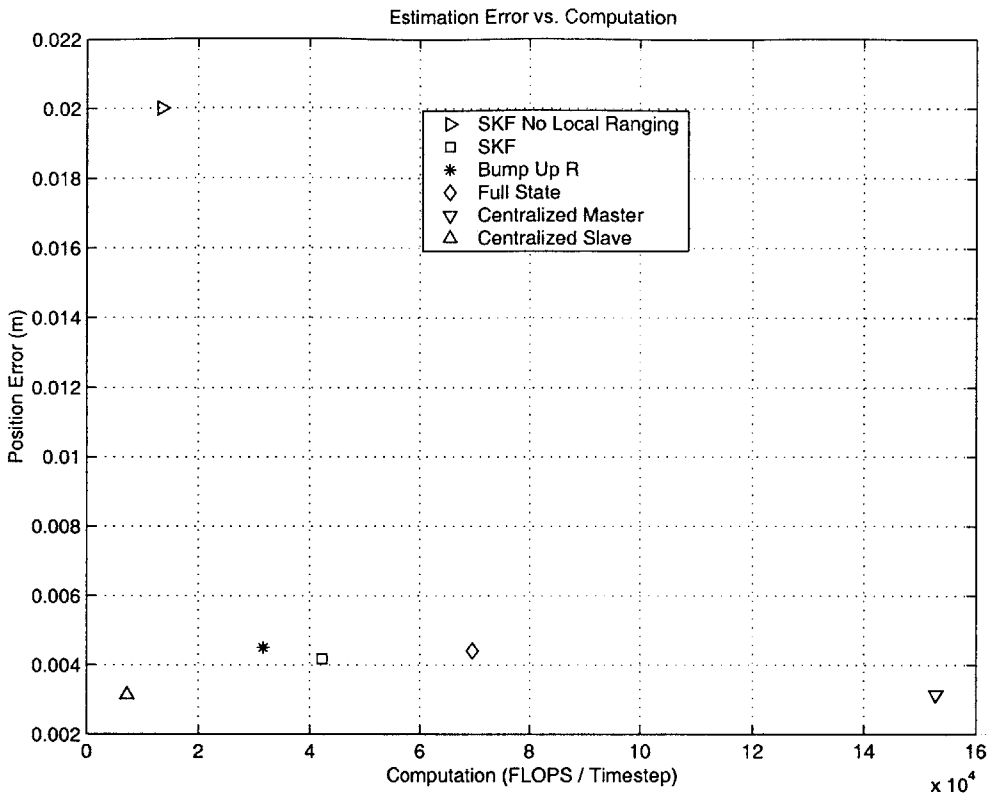


Figure 4-10: Estimation error as a function of computation per vehicle for various decentralization methods.

4.5.3 Data Flow Performance Validation

The FFIT testbed [60] enables data to be collected on the communication and computational demands for each algorithm. The relative complexity of the algorithms can also be studied from the time required to generate a useful solution. The FFIT testbed currently contains 5 PCs in total - 4 spacecraft PCs and 1 simulation engine, so the simulations are limited to a fleet of 4 vehicles. While this is not as large as desired, the results provide a baseline that can be extrapolated for larger fleets (*e.g.*, 16-32 vehicles). The results that follow were taken directly from the FFIT testbed using a situation similar to the one in the previous section (*i.e.*, 2-D vehicles dynamics with vehicles traveling in circles). The results were taken in real-time at a time-step of 5 seconds, with each vehicle transmitting its data at a rate of 9600 baud (typical data rate for a Nanosat missions [9]).

Figure 4-10 shows the trade between estimation accuracy and computation. The centralized method provides the lowest error, but this plot clearly shows the computational price that is paid for this high accuracy. The master vehicle in the centralized

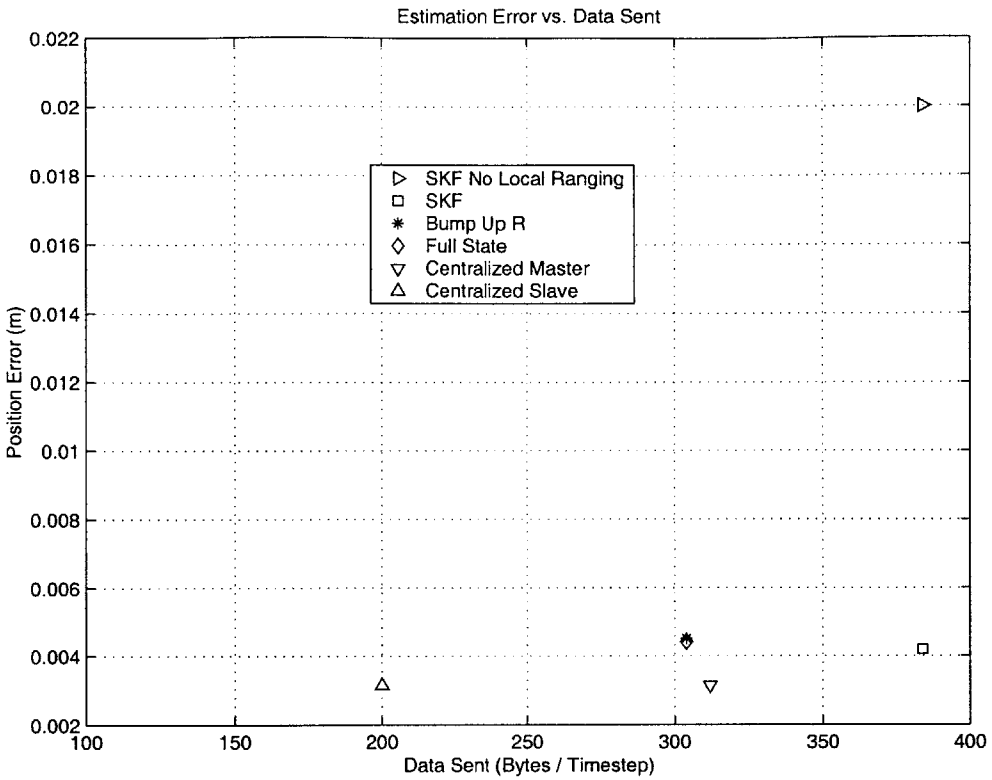


Figure 4-11: Estimation error as a function of amount of transmitted data per vehicle for various decentralization methods.

estimation scheme ends up more than 10 times more computationally loaded than the slaves. In a mission where science objectives need to be carried out along with relative navigation, it may not be desirable to have one vehicle so heavily loaded while the others are doing virtually nothing.

The reduced-order methods (*i.e.,* all SKF methods and bump-up-R) provide a good trade of accuracy vs. computation plot (especially compared to the "no local ranging" data point). Since each vehicle only estimates its local state, the computation is well distributed and as a result, the computation level for each vehicle is only slightly more than a centralized slave. The results in Figure 4-10 show that the full-state method has a higher computational load than the other reduced-order methods (*e.g.,* full SKF method), but this extra computation provides no additional performance benefit.

Overall, the other reduced-order methods require relatively small amounts of inter-spacecraft communication as evidenced in Figure 4-11. The method with the most communication is the SKF due to the transmission of portions of covariance matrices. However, given the small accuracy benefit of the SKF, it may be desirable to use

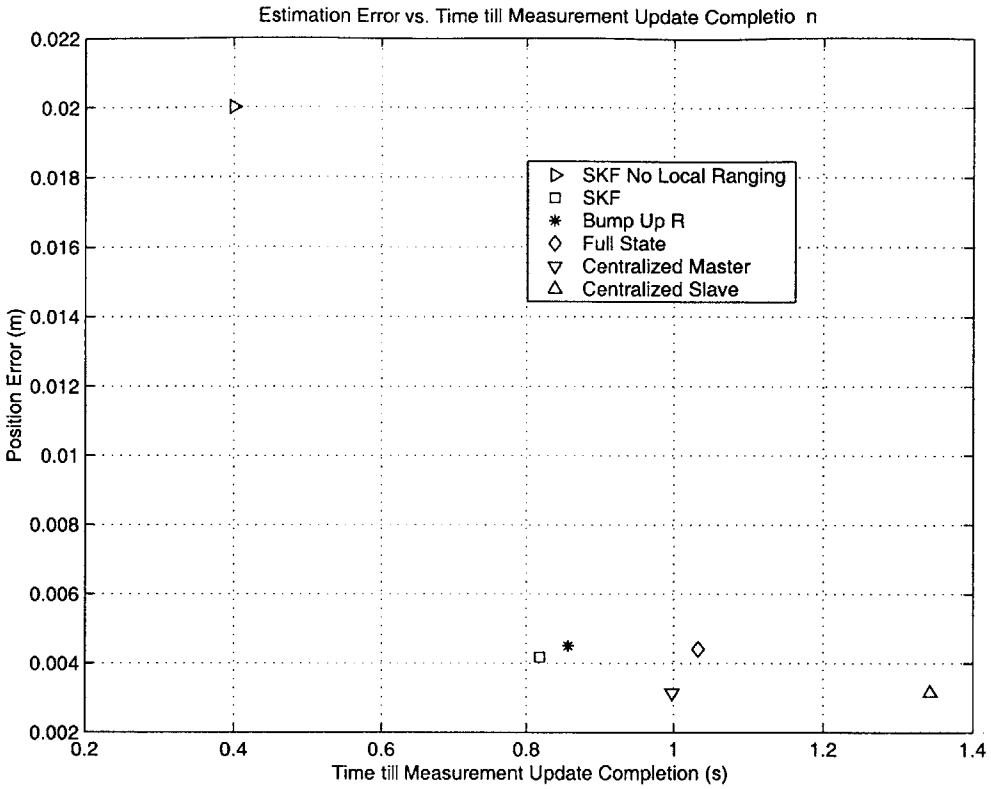


Figure 4-12: Estimation error as a function of solution time per vehicle for various decentralization methods.

more communication bandwidth in exchange for the increased estimator accuracy. The centralized method does very well in this comparison because the fleet only has 4 vehicles and only the measurement vectors and solutions for the two-dimensional problem are transmitted. The issues when scaling to larger fleets are explored later in this section.

Figure 4-12 shows the estimation error vs. solution time trade. This plot makes explicit the degree of complexity of each method. Points further to the right indicate that a great deal of computation, communication and/or fleet synchronization is required, thus making the method more complicated to execute overall. The reduced order methods appear to be the clear winners in this trade study for $N = 4$. They minimize computation as well as communication and still perform quite well from an estimation accuracy standpoint. Furthermore, due to the cascade nature of the measurement updates, each vehicle need only be synchronized with the vehicle before it in the fleet. In the centralized case, all vehicles must synchronize with the master prior to receiving a measurement update, resulting in longer solution times.

The results presented above for a 4-vehicle fleet provided insights on the relative

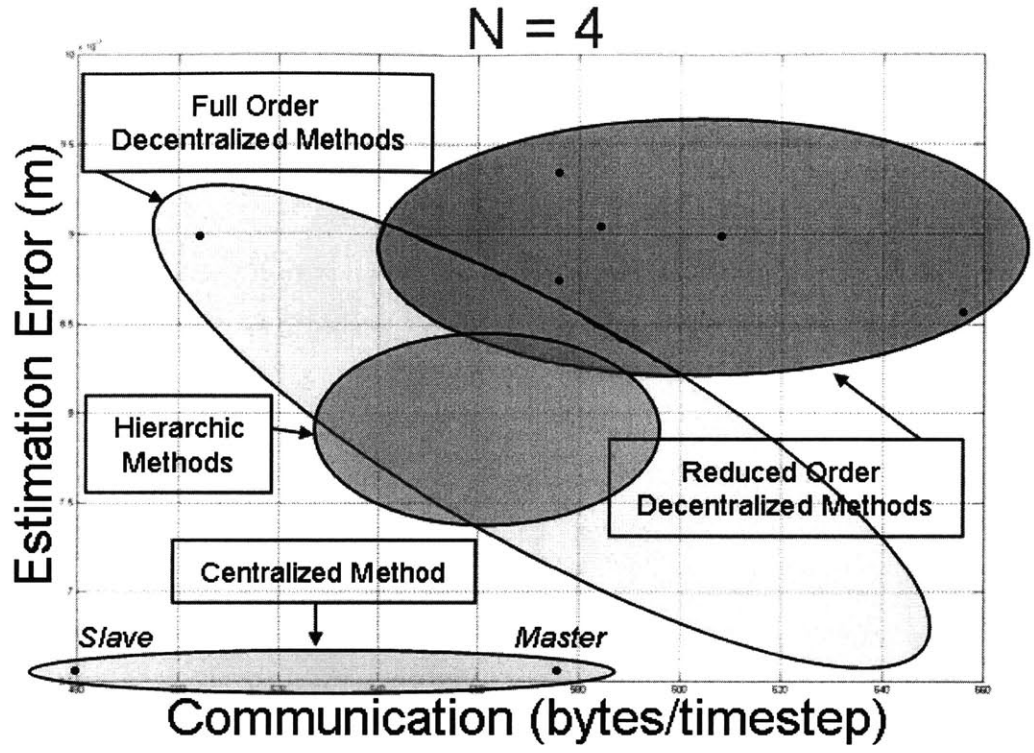
merits of the various estimation architectures. The reduced-order methods exhibit near optimal estimation accuracy while keeping communication and computation to a minimum, proving that decentralized estimation is possible even with correlated states. The full state methods were shown to not provide any performance increase over the reduced-order methods in terms of computation or estimation accuracy. From a communication and computation perspective, for these simplified simulations, the centralized filter appears to be a viable option for fleets of only 4 spacecraft, provided the imbalance in computational load is acceptable. However, this analysis has not included any notion of fleet robustness or required fleet connectivity, which are the commonly cited disadvantages of centralized estimators [49, 64].

To gain some insight into how these algorithms scale to a larger fleet, Figures 4-13 and 4-14 use the $N = 4$ case as a baseline and illustrate graphically how the performance of the various estimation architectures changes when the fleet size increases past 4 vehicles. The scaling presented in these figures comes from an algebraic analysis of the required matrix calculations [65]. Figure 4-13 illustrates the approximate regions on the communication and computation plots where the various architectures lie. The "Full State" area stretches down to the bottom right on both plots because it includes the information filter as well. Figure 4-14 shows how these regions evolve when for N larger than 4 (*i.e.* 16 or 32).

In the communication plot, the scaling for the full state and centralized methods grows with N^2 due to the need to send information matrices in the information filter case and full measurement vectors in the centralized Kalman filter architecture. For the reduced-order methods, there is *no* communication scaling with N since all transmitted data is only a function of the local vehicles' states. In the computation plot, the scaling for the full order methods grows with N^3, while the scaling is N^6 for the centralized methods. The reduced-order methods however, experience no scaling in computation.

The scaling results presented above indicate that the reduced-order methods perform the best in larger fleets due to their limited state size. The communication and computation requirements for the centralized and full state methods grow extremely rapidly for larger fleets, quickly rendering those methods infeasible. However, all methods presented in this chapter require some degree of fleet synchronization, which might limit the implementation for large N.

Due to the physical limitations of the FFIT testbed, it is not possible to simulate



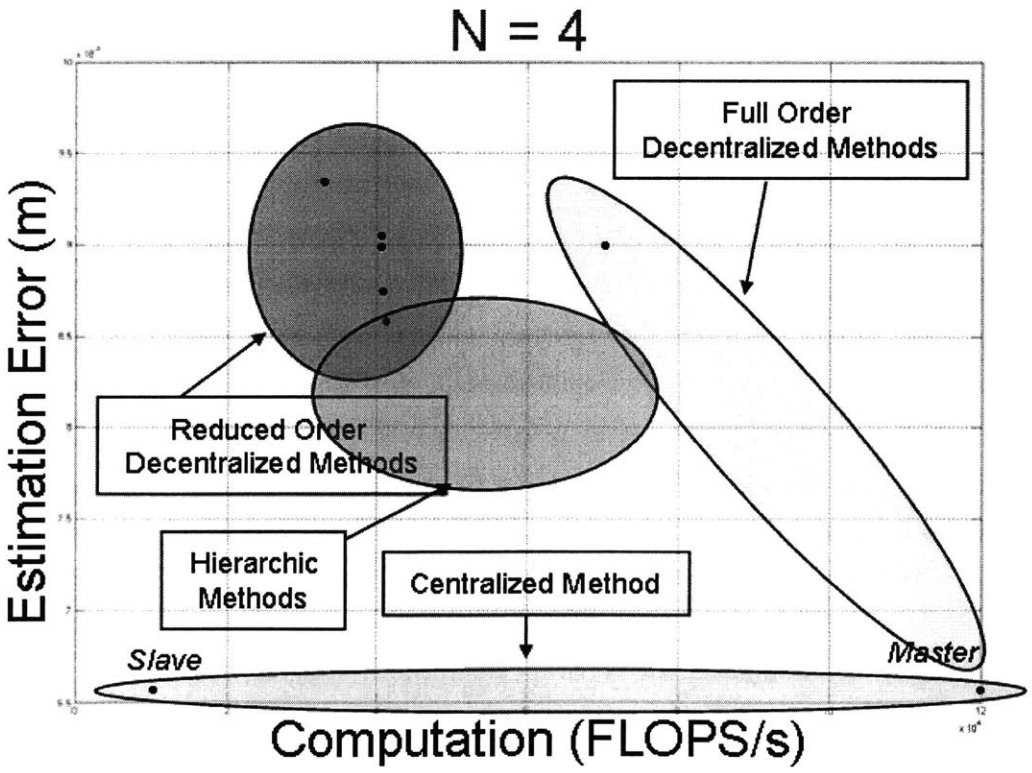


Figure 4-13: Trends for $N = 4$.

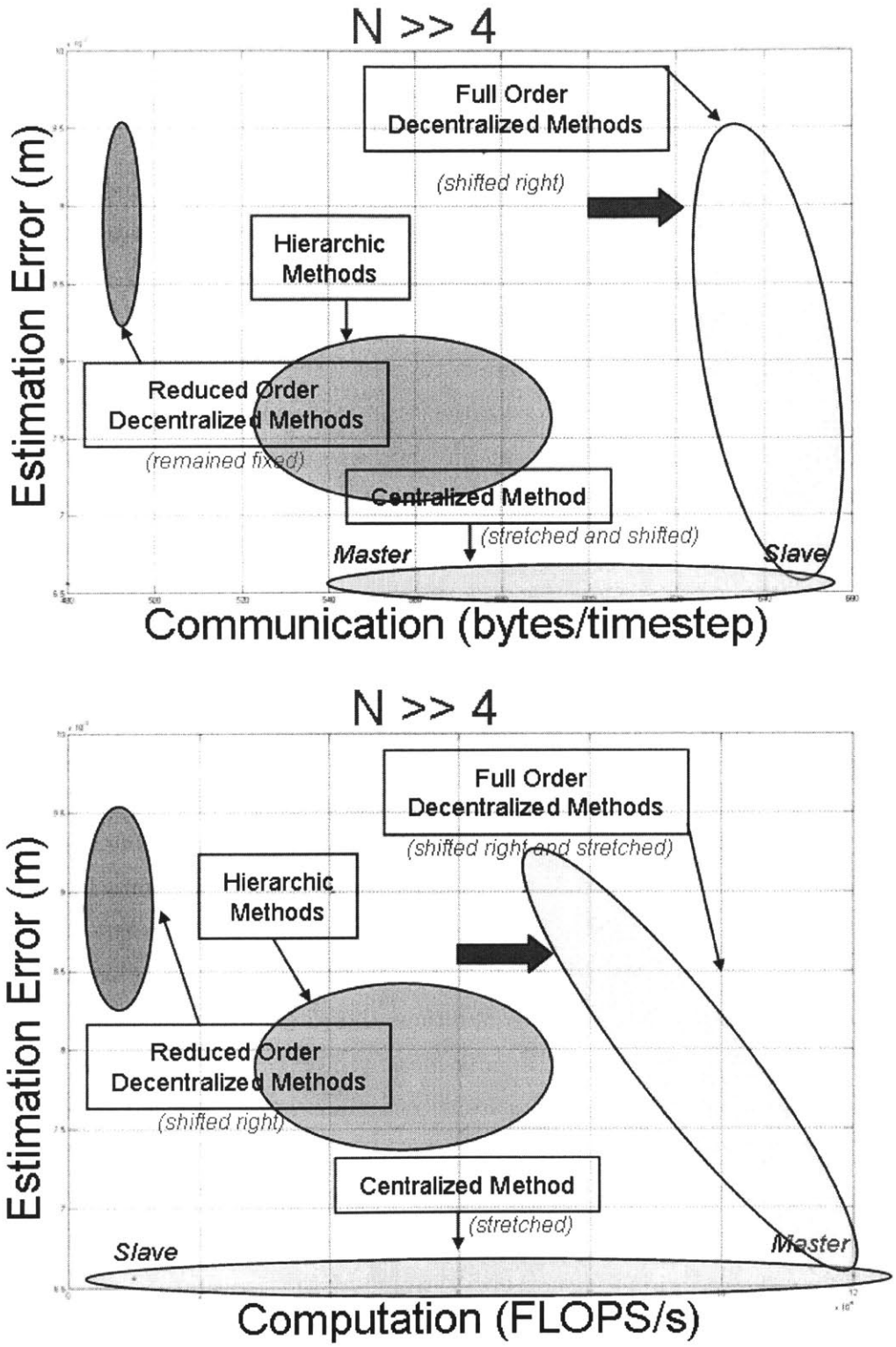


Figure 4-14: Trends for $N \gg 4$.

the hierarchic method described in this chapter. However, since the hierarchic methods are simply other estimators running on various clusters, the above results can be extrapolated to the hierarchic architectures. If one assumes that the same estimator is run at each level of the hierarchy, then the communication and computational load for each cluster master would be exactly twice that of the cluster slaves. The slaves' communication and computational load scale exactly the same as predicted in Figures 4-13 and 4-14. Thus, the hierarchic methods permit scaling mitigation by simply reducing the size of each cluster.

An important drawback to hierarchic clustering is the summation of cluster error variances shown in Table 4.1. Of course, to reduce the impact of these errors, larger clusters can be used. Using reduced-order estimators such as the SKF permit large cluster sizes due to their appealing scaling characteristics. Thus, a viable navigation option for almost any large fleet would be a reduced order estimator implemented in a hierarchic cluster architecture.

4.6 Conclusions

Fleet estimation for formation flying missions in and beyond LEO presents a challenging problem due to the nonlinear ranging measurements that are required to retain fleet observability in the absence of GPS. These local ranging measurements strongly couple the states of each vehicle and complicate the decentralization of the estimation algorithms by requiring cascaded fleet iterations during the measurement update. This chapter analyzed several estimation architectures and compared various algorithms using simplified 2D simulations that facilitated detailed studies of the effects of the nonlinearity in the ranging measurements and the correlation between the vehicle state estimates. Results from these simulations showed that the decentralized reduced-order estimators provide a good balance between communication, computation and performance when compared to centralized and full order methods, and thus could be a feasible relative navigation approach for future missions. The extrapolation of these results to larger fleets strongly indicated that the centralized (and decentralized full-order) filters will have prohibitively high communication and computational requirements. However, the reduced-order estimators presented, of which the Schmidt-Kalman filter was the best, exhibit no such growth in the communication or computation demands. While the reduced-order decentralized approaches reduce

and distribute the computation more equitably, they are fundamentally limited by the degree of synchronization required within the fleet to calculate the state estimates. The hierarchic architectures discussed address this limitation by splitting the fleet into sub-teams that can function asynchronously. Thus the conclusions from these initial studies are that a viable estimation architecture for almost any sized fleet would be comprised of reduced-order estimators implemented within a hierarchic architecture.

Chapter 5

Conclusions

As the proposed formation flying missions become more complex, it is becoming increasingly important to analyze the effects of integrating the entire set of guidance, navigation and control algorithms. This thesis has described several contributions made towards the development of distributed estimation and control technologies for formation flying spacecraft. The following sections summarize the primary contributions of this thesis.

5.1 Orion

The preparation for the Orion Formation Flying mission provided a real-world application to the distributed architecture studies in this thesis. Furthermore, this work strongly emphasized the hardware and software challenges associated with microsatellites that drive some of the studies presented in this work.

Science Computer Development: The main processing entity for the Orion spacecraft, the Science Computer, was built and tested. A detailed design of the communications interface was completed to permit timely GPS telemetry to be read from all three GPS receivers over a single serial port. Once built, flight software was written to drive the communications. This was tested successfully to prove that all required data could be sent within the allotted time. A payload power system was also built to permit the Command and Data Handling (C&DH) CPU onboard Orion to provide power-switching services to the Orion Payload.

Orion Software Framework: Due to infrequent ground communications, the Orion software will need to be almost entirely autonomous. This requires an extremely

robust software platform on which to execute the GNC algorithms. Embedded Linux was chosen as the operating system for the Science Computer to provide a cheap, feature-rich environment for writing the Orion flight code. A soft-realtime framework was developed and tested to control the execution, timing and data interactions of the required GNC algorithms. A demonstration of this framework proved that all algorithms could be executed at the correct rate with all required data being passed appropriately.

5.2 Distributed GNC Architectures

With many formation flying missions (such as Orion) being proposed and flown in recent years, it is becoming increasingly important to evaluate the impact of distributed computing across the fleet of vehicles. The following summarizes the advancements made into the development of efficient and effective control and estimation architectures for formation flying spacecraft.

Task Allocation Routine: A key step in distributed programming is ensuring that each processor has a uniform computational load. Part of the GNC architecture research involved the development of an algorithm for distributing the computational load associated with the optimal assignment problem of placing spacecraft on a closed-form ellipse in LEO. The task allocation algorithm was shown to adapt to the changing spacecraft computational loads, while keeping communication to a minimum.

FFIT Testbed: In order to verify the GNC architectures for formation flying missions prior to flight, a testbed was required that could quickly validate the communication and computational loads for any architecture. The *Formation Flying Information Technology* (FFIT) testbed was developed to easily create and evaluate different GNC architectures. The FFIT testbed was demonstrated to be a valuable tool for detecting subtle oversights in fleet architecture design such as communication bottlenecks, algorithmic synchronization, and unbalanced CPU loading.

5.3 Decentralized Estimation Using GPS Augmented with Local Ranging

Estimation for a large fleet of vehicles can be challenging due to the vast amounts of distributed measurement data. Decentralized estimation algorithms are required to

prevent excessive communication of this data between the vehicles. However, the additional ranging measurements required to perform relative navigation beyond LEO tend to couple the states describing the motion of each vehicle with respect to the master, which greatly complicates the process of decentralizing the algorithms. The following summarizes the contributions made towards developing scalable decentralized estimation algorithms with augmented GPS measurements.

Schmidt-Kalman Filter for Reduced Order Estimation: For large fleets, estimation tasks become onerous due to the many measurements and large state size. Thus, a reduced-order estimator was required that would enable each vehicle to focus on estimating only its own state rather than the entire fleet state. However, distributing the estimation process into several reduced-order filters is complicated because the local range measurements used to augment the GPS measurements tend to couple the vehicle states. An iterated Schmidt-Kalman filter formulation was used to eliminate the other vehicles' states from the local estimator's state vector. The Schmidt-Kalman filter is an approximation, but the results in this thesis illustrate that it provides estimation accuracy that approaches the centralized result. Furthermore, results from the FFIT testbed indicate that the amount of communication and computation is manageable, even for large fleets.

Hierarchic Methods for Scaling: While the reduced-order methods presented in this thesis were shown to provide excellent accuracy for small amounts of communication and computation, it was apparent that the synchronization requirements of these algorithms would prevent scaling to very large fleets. Hierarchic architectures were proposed as a method for mitigating the scaling issues of large fleets. It was shown in this analysis that reducing the fleet to several smaller clusters permitted greater scaling because individual clusters need not synchronize with each other. One drawback of hierarchies is the error incurred when ranging measurements are used from vehicles in other clusters. It was shown in this thesis that the use of reduced order methods in a hierarchic architecture reduced this error by permitting fewer and larger clusters due to their applicability to large fleets.

5.4 Future Work

Orion: With the revised Orion mission plan, future work will focus on the required changes to fly two Orion vehicles. Many of these changes will be hardware in nature

(*i.e.*, release mechanisms from the Shuttle cargo bay) since much of the software is not spacecraft-specific. Once the hardware has been finalized and accepted by NASA safety personnel, end-to-end tests will be required to verify the hardware and software. This will require simulation computers to emulate the space environment for many of Orion's systems.

Testbed Development: While the current form of the FFIT testbed was useful for analyzing the data and computational flow requirements for estimation and control architectures, it cannot address issues pertaining to the real-time execution of the software that is ultimately required before a software system can be implemented on a fleet. To facilitate this need, the next step in the development of this testbed will be to migrate the existing software to a real-time Linux platform. Moving to real-time Linux will permit the use of ObjectAgent (OA), a real-time distributed communications middleware package designed for distributed spacecraft applications. OA will enable detailed studies of multi-vehicle interactions and permit many more vehicles than the current FFIT testbed since it relies only on TCP/IP connections.

Decentralized Estimation: The next step in the decentralized estimation research will be to apply the techniques outlined in this thesis to the specific scenarios of upcoming formation flying missions. In particular, studies need to be carried out that extend the estimation algorithms into three dimensions. Also, since many local ranging devices require accurate estimation of time, additional states should be added to the estimators that represent the time offset between each vehicle.

5.5 Closing Remarks

The work presented in this thesis has laid the groundwork for detailed mission-specific research. A testbed has been introduced that can be used for detailed architecture analysis and reduced-order decentralized estimators have been proposed and tested that provide highly accurate estimates with low demands on spacecraft resources. Using the insight and tools presented in this thesis, mission designers will be able to test and verify future and current formation flying missions. By augmenting the GPS measurements to include local ranging devices, the decentralized estimation research has extended the applicability of this navigation approach to a much wider class of missions. Furthermore, this research has enabled the scaling of these estimation algorithms to much larger fleets.

Bibliography

[1] F. Morring Jr., "EO-1 Satellite Demonstrates Autonomous Formation Flying," *Aviation Week and Space Technology*, ISSN: 0005-2175), vol. 154, no.21, pp. 46, 48-49. May 21, 2001.

[2] F. H. Bauer, K. Hartman, E. G. Lightsey, "Spaceborne GPS: Current Status and Future Visions," *Aerospace Conference Proc.*, IEEE, March 1998.

[3] F. H. Bauer, K. Hartman, J. P. How, J. Bristow, D. Weidow, and F. Busse, "Enabling Spacecraft Formation Flying through Spaceborne GPS and Enhanced Automation Technologies," Proc. of the *ION-GPS Conf.*, Sept. 1999

[4] A. Das and R. Cobb, "TechSat21 – Space Missions Using Collaborating Constellations of Satellites," Proceedings of the *12th Annual AIAA/USU Conference on Small Satellites,* Aug. 1998.

[5] **eo1.gsfc.nasa.gov/Technology/FormFly.html**

[6] **planetquest.jpl.nasa.gov/Starlight/starlight_index.html**

[7] **nmp.jpl.nasa.gov/st5/**

[8] **www.vs.afrl.af.mil/factsheets/TechSat21.html**.

[9] P. Ferguson, F. Busse, B. Engberg, J. How, M. Tillerson, N. Pohlman, A. Richards, R. Twiggs, "Formation Flying Experiments on the Orion-Emerald Mission", *AIAA Space 2001 Conference and Exposition*, Albuquerque, New Mexico, Aug. 2001.

[10] J. Leitner, F. Bauer, D. Folta, R. Carpenter, M. Moreau and J. How, "Formation Flight in Space," GPS World, Feb. 2002, pp. 22–31.

[11] G. Inalhan, F. Busse, J. How, "Precise Formation Flying of Multiple Spacecraft Using Carrier-Phase Differential GPS," *Proc. of AAS Space Flight Mechanics Winter Meeting, Clearwater, FL.*, AAS 00-109, Jan. 2000.

[12] *International Journal of Robust and Nonlinear Control*, Vol. 12, No. 2, Jan 2002.

[13] Proceedings of the *International Symposium on Formation Flying*, Toulouse France, October 2002.

[14] T. Yang, *Robust Thruster Optimization with Failure Detection for Formation Flying Satellites,* SM Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics. May, 2003.

[15] M. Paluszek, "TechSat-21 Hall Thruster Fault Detection Algorithm Description Document,", *Princeton Satellite Systems* internal paper, September 17, 2002.

[16] F. Busse, *Precise Formation-State Estimation in Low Earth Orbit using Carrier Differential GPS.* Ph.D. Dissertation, Stanford University, Dept. of Aeronautics and Astronautics, November 2002.

[17] F. Busse and J. How, "Real-time Experimental Demonstration of Precise Decentralized Relative Navigation for Formation Flying Spacecraft", *AIAA Guidance, Navigation, and Control Conference*, August 2002.

[18] F. Busse and J. How, "Demonstration of Adaptive Extended Kalman Filter for Low Earth Orbit Formation Estimation Using CDGPS", *Institute of Navigation GPS Meeting*, September 2002.

[19] T. Corazzini, A. Robertson, J. Adams, A. Hassibi, and J. How, "Experimental demonstration of GPS as a relative sensor for formation flying spacecraft," Fall issue of the *Journal of The Institute Of Navigation,* Vol. 45, No. 3, 1998, pp. 195–208.

[20] E. Olsen, C. Park, and J. How, "3D Formation Flight using Differential Carrier-phase GPS Sensors," *The J. of The Institute Of Navigation,* Spring, 1999, Vol. 146, No. 1, pp. 35–48.

[21] D. Miller *et al.* , "SPHERES: A Testbed For Long Duration Satellite Formation Flying In Micro-Gravity Conditions," AAS00-110.

[22] Enright, J., Sedwick, R., Miller, D., "High Fidelity Simulation for Spacecraft Autonomy Development", *Canadian Aeronautics and Space Journal*, Dec., 2001.

[23] F. Busse, J. How, J. Simpson, J. Leitner, "Orion-Emerald: Carrier Differential GPS for LEO Formation Flying," *Proc. of IEEE Aerospace Conference 2001*, Big Sky, MT, Mar 2001.

[24] **http://dshell.jpl.nasa.gov/**

[25] T. Corazzini, *Onboard Pseudolite Augmentation for Spacecraft Formation Flying.* Ph.D. Dissertation, Stanford University, Dept. of Aeronautics and Astronautics, Aug. 2000.

[26] C. P. Park, *Precise Relative Navigation using Augmented CDGPS.* Ph.D. Dissertation, Stanford University, Dept. of Mechanical Eng., June 2001.

[27] J. R. Carpenter, C. Gramling *et al*, "Relative Navigation of Formation-Flying Satellites," from the proceedings of the *International Symposium on Formation Flying*, Toulouse France, October 2002.

[28] P. Stadter, R. Heins, et al., "Enabling Distributed Spacecraft Systems with the Crosslink Tranceiver," *AIAA Space Conference and Exposition*, Aug. 2001. Paper 2001-4670.

[29] G. Purcell, D. Kuang, S. Lichten, S. Wu and L. Young, "Autonomous Formation Flyer (AFF) Sensor Technology Development," TMO Progress Report 42-134, August 1998.

[30] H. Durrant-Whyte, "Sensor models and multi-sensor integration," *Int. J. Robotics Research*, 7(6):97-113, 1998.

[31] J. Speyer, "Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem," *IEEE Trans. Automatic Control,* vol. AC-24(2), 1979, pp.266-299.

[32] S. F. Schmidt, "Application of State-Space Methods to Navigation Problems," in C. T. Leondes (ed.), *Advances in Control Systems*, Vol. 3, New York: Academic Press, 1966.

[33] **http://web.mit.edu/~jhow/www/orion/**

[34] F. Gottifredi, L. Marradi, G. Adami, "Results from the ARP-GPS receiver flight on the ORFEUS-SPAS Mission," Proc. of the *ION-GPS Conf.*, Sept. 1999.

[35] M. J. Unwin, P. L. Palmer, Y. Hashida, C. I. Underwood, "The SNAP-1 and Tsinghua-1 GPS Formation Flying Experiment," Proc. of the *ION-GPS Conf.,* Sept. 2000.

[36] I. Kawano, M. Mokuno, T. Kasai, T. Suzuki, "First Autonomous Rendezvous using Relative GPS Navigation by ETS-VII," Proc. of the *ION-GPS Conf.,* Sept. 1999.

[37] Z. Kiraly, B. Engberg, F. Busse, R. Twiggs, J. How, "The Orion Microsatellite: A Demonstration of Formation Flying on Orbit," *Conference Proc. of AIAA Space Technology 99*, SSC99-VI-8, Sep. 1999.

[38] F. Busse, G. Inalhan, J. How,"Project Orion: Carrier Phase Differential GPS for Formation Flying," *Proc. of AAS Guidance and Control Conference*, Breckenridge, CO, Feb 2000.

[39] J. How, R. Twiggs, D. Weidow, K. Hartman, and F. Bauer, "ORION: A low-cost demonstration of formation flying in space using GPS," in *AIAA/AAS Astrodynamics Specialists Conference*, (Boston, MA), Aug. 1998.

[40] B. Engberg and R. Twiggs, "The Orion Microsatellite: A Testbed for Command, Control, and Communication for Formation Fleets," SSC00-II-4, in *14th Annual AIAA/USU Conference on Small Satellites*, Sep. 2000.

[41] 1Lt. C. Brackett, M. Kumashiro and A. Peffer, "AFRL's Multi-Satellite Deployment System (MSDS)", 15th Annual/USU Conference on Small Satellites, SSC01-XI-5, Logan, Utah, August 13-16, 2001.

[42] C. Kitts, R. Twiggs, J. How, F. Pranajaya, B. Palmintier, "Emerald: A Low-Cost Spacecraft Mission for Validating Formation Flying Technologies," Proc. of the *IEEE Aerospace Conference*, March 1999.

[43] A. Robertson, G. Inalhan, and J. How, "Spacecraft Formation Flying Control Design for the Orion Mission," proceedings of the 1999 *AIAA Guidance Navigation, and Control Conference*, pp. 1562–1575.

[44] E. G. Lightsey, *Development and Flight Demonstration of a GPS Receiver for Space.* Ph.D. Dissertation, Stanford University, Dept. of Aeronautics and Astronautics, Jan. 1997.

[45] J. C. Adams and J. How, "Experiments in GPS Attitude Determination for Spinning Spacecraft with Non-aligned Antenna Arrays," Proc. of the *ION GPS Conf.,* Sept. 1998.

[46] F. Curti and F. Diani, "Study on Active Magnetic Attitude Control for Italian Spacecraft Bus MITA," *14th Int. Symposium on Spaceflight Dynamics.*

[47] F. H. Bauer, J. O. Bristow, J. R. Carpenter, J. L. Garrison, K. Hartman, T. Lee, A. Long, D. Kelbel, V. Lu, J. P. How, F. Busse, P. Axelrad, and M. Moreau,

"Enabling Spacecraft Formation Flying through Spaceborne GPS and Enhanced Autonomy Technologies," in *Space Technology*, Vol. 20, No. 4, pp. 175–185, 2001.

[48] T. Schetter, M. Campbell, D. Surka, "Comparison of Multiple Agent Based Organizations for Satellite Constellations," *2000 FLAIRS AI Conference*, Orlando FL, April, 2000.

[49] A. G. O. Mutambara *Decentralized Estimation and Control for Multisensor Systems*, CRC Press LLC, 1998.

[50] C. Park and J. How, "Precise Relative Navigation using Augmented CDGPS," Proceedings of the *ION-GPS Conference*, Sept. 2001.

[51] A. Richards, T. Schouwenaars,, J. How and E. Feron, "Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed Integer Linear Programming," proceedings of the *AIAA Guidance, Navigation, and Control Conference*, August 2001, and submitted to the *AIAA Journal of Guidance, Control and Dynamics*, October, 2001.

[52] M. Tillerson, G. Inalhan, and J. How,"Coordination and Control of Distributed Spacecraft Systems Using Convex Optimization Techniques," accepted for publication in the *International Journal of Robust and Nonlinear Control*, July 2001.

[53] M. Tillerson. "Coordination and Control of Multiple Spacecraft using Convex Optimization Techniques". SM Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics. June, 2002.

[54] R. Beard and F. Hadaegh, "Constellation templates: An approach to autonomous formation flying", in *World Automation Congress,* Anchorage, Alaska, May 1998.

[55] J. Tierno "Collective Management of Satellite Clusters," proceedings of the *AFOSR workshop on Dynamics and Control*, August 2000.

[56] M. Tillerson and J. How, "Advanced Guidance Algorithms for Spacecraft Formation Flying," submitted to the *2002 American Control Conference*, Sept 2001.

[57] K. London, J. Dongarra, S. Moore, P. Mucci, K. Seymour, T. Spencer, "End-user Tools for Application Performance Analysis, Using Hardware Counters," *Int. Conf. on Parallel and Distributed Computing Systems,* Aug. 2001.

[58] W. Cash, "MAXIM Preliminary Design," Work supported by the *NASA Institute for Advanced Concepts*, August 2000.

[59] S. Curtis, "The Magnetospheric Multiscale Mission ... Resolving Fundamental Processes in Space Plasmas," *NASA Goddard Space Flight Center*, Greenbelt, Maryland, December 1999. NASA/TM2000-209883.

[60] P. Ferguson, T. Yang, M. Tillerson and J. How, "New Formation Flying Testbed for Analyzing Distributed Estimation and Control Architectures." Presented at the *AIAA Guidance, Navigation and Control Conference*, Monterey, California, August 2002.

[61] P. G. Kaminski, "Square root filtering and smoothing for discrete processes," Ph.D. thesis, Stanford University, 1971.

[62] J. R. Carpenter, "Decentralized control of satellite formations," International Journal of Robust and Nonlinear Control. *Int. J. Robust Nonlinear Control* 2002; 12:141-161 (DOI: 10.1002/rnc.680).

[63] **http://origins.jpl.nasa.gov/missions/tpf.html**.

[64] J. R. Carpenter, "Partially Decentralized Control Architectures for Satellite Formations,", *Presented at the AIAA Guidance, Navigation and Control Conference and Exhibit*, Monterey, California, August, 2002.

[65] M. S. Grewal and A. P. Andrews, *Kalman Filtering Theory and Practice*, pp. 208-210, Prentice-Hall Inc., 1993.

 1760
40