Dynamic Programming Applied to Electromagnetic Satellite Actuation

by

Gregory John Eslinger

B.S., United States Air Force Academy (2011)

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

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author . . . .
Department of Aeronautics and Astronautics
May 22, 2013

Certified by. . . . .
Alvar Saenz-Otero
Principal Research Scientist
Thesis Supervisor

Certified by. . . . .
David W. Miller
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Dynamic Programming Applied to Electromagnetic Satellite Actuation

by

Gregory John Eslinger

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Electromagnetic formation flight (EMFF) is an enabling technology for a number of space mission architectures. While much work has been done for EMFF control for large separation distances, little work has been done for close-proximity EMFF control, where the system dynamics are quite complex. Dynamic programming has been heavily used in the optimization world, but not on embedded systems. In this thesis, dynamic programming is applied to satellite control, using close-proximity EMFF control as a case study. The concepts of dynamic programming and approximate dynamic programming are discussed. Several versions of the close-proximity EMFF control problem are formulated as a dynamic programming problems. One of the formulations is used as a case study for developing and examining the cost-to-go. Methods for implementing an approximate dynamic programming controller on a satellite are discussed. Methods for resolving physical states and dynamic programming states are presented. Because the success of dynamic programming depends on the system model, a novel method for finding the mass properties of a satellite, which would likely be used in the dynamic programming model, is introduced. This method is used to characterize the mass properties of three satellite systems: SPHERES, VERTIGO, and RINGS. Finally, a method for position and attitude estimation for systems that use line-of-sight measurements that does not require the use of a model is developed. This method is useful for model validation of the models used in the dynamic programming formulation.

Thesis Supervisor: Alvar Saenz-Otero
Title: Principal Research Scientist

Thesis Supervisor: David W. Miller
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank the Air Force for allowing me the opportunity to attend graduate school. I would also like to recognize a number of people and entities for their support of RINGS.

- DARPA, especially Dave Barnhart.
- Aurora Flight Sciences, including John, Roedolph, and Joanne.
- The RINGS team at the University of Maryland: Allison Porter, Dustin Alinger, and Dr. Raymond Sedwick.
- My thesis supervisors, Prof. Miller and Dr. Saenz-Otero.

I would also like to recognize the following people:

- Alex Buck, my partner in crime. Despite you being in the Navy, we managed to make a good team. Best of luck.
- Bruno Alvisio, good luck carrying on RINGS.
- Hal Schmidt and Evan Wise, its been a pleasure working at the mill with you gentlemen. May our adventures in California prove even more fruitful.
- My fellow labmates and SPHERES team members. Keep up the good work.
- Michael Frossard, for keeping America safe (or at least training to) while I wrote this work.
- My mother, father, sister, and girlfriend. Thank you for your love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electromagnetic Formation Flight (EMFF) is the concept of using electromagnets to control the relative position and orientation of spacecraft while flying in a formation. EMFF is an enabling technology for a number of spacecraft missions architectures. While EMFF has been demonstrated on the ground [1], it has yet to be demonstrated in space. The Resonant Inductive Near-field Generation System (RINGS) will be the first time EMFF is demonstrated in a microgravity environment [2]. However, the close proximity of the non-holonomic coils makes control of RINGS difficult, requiring advanced control methods. While there are many potential control methods for RINGS, this work will use RINGS as a case study for applying dynamic programming to satellite control.

Dynamic programming has been used as a tool for optimization in many different fields of study, including mathematics, economics, and computer science; however, it is rarely used for spacecraft applications. Until recently, spacecraft dynamics, which include translational and rotational dynamics, were simple enough that linear controllers were sufficient. With the advent of novel, nonlinear actuators, spacecraft dynamics are becoming increasingly complex, making dynamic programming a potentially useful option. The limiting factor for dynamic programming was the limited processing power and storage capacity of spacecraft avionics. While computing on spacecraft still lags behind terrestrial processors, there have been advances in spacecraft processing, making dynamic programming a possibility.

However, applying dynamic programming to a physical, embedded systems is not as straightforward as applying it to a simulation. An embedded system is a system that a person does not have the ability to directly interface with the system. For example, engineers typically cannot connect a monitor and keyboard to a satellite, instead the engineers must infer the state of the system via telemetry and physical actuation. When dynamic programming is applied to simulations, the simulations typically are built to easily interface with the dynamic program. Embedded systems, on the other hand, are not inherently designed to accommodate controllers built using dynamic programming. This thesis discusses methods for bridging the gap between embedded physical systems, specifically satellites, and dynamic programming.

The rest of this thesis is organized in two general sections. Chapters 2 to 5 discuss applying dynamic programming to RINGS. Chapter 2 gives an overview of dynamic programming and approximate dynamic programming. Chapter 3 formulates the RINGS system as a dynamic programming problem. Chapter 4 discusses how to find the cost-to-go using the formulations described in Chapter 3. Chapter 5 describes methods for implementing the solutions derived in Chapter 4 on the physical system.

The second half of the thesis, Chapters 6 to 8, describes methods for finding and validating the spacecraft model, which is significant because the model is an integral part of the dynamic programming formulation. Chapter 6 presents a novel method for determining the mass properties of a satellite using nonlinear programming. Chapter 7 applies Chapter 6 to three different systems, including RINGS. Chapter 8 discusses methods for state estimation without the use of models, which is useful for dynamic programming model validation.

1.1 Motivation

This thesis applies dynamic programming to spacecraft actuation, with a primary focus on EMFF and RINGS. However, this work can be abstracted to the more general notion of spacecraft formation control using dynamic programming, or even general satellite control using dynamic programming. The areas this thesis is motivated by

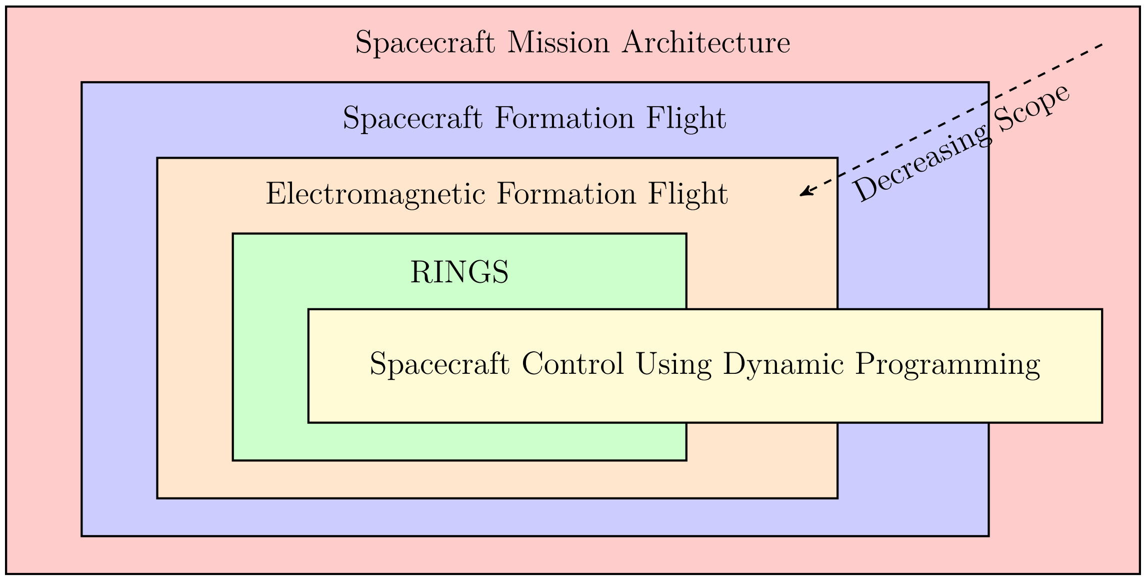

Spacecraft Mission Architecture
Spacecraft Formation Flight
Decreasing Scope
Electromagnetic Formation Flight
RINGS
Spacecraft Control Using Dynamic Programming

Figure 1-1: Scope of Motivation

is described by Fig. 1-1.

While the motivation for spacecraft development is well known, the motivation for spacecraft formation flight, electromagnetic formation flight, and RINGS are not. The motivations for these are discussed in Sections 1.1.1, 1.1.2 and 1.1.3, respectively.

1.1.1 Spacecraft Formation Flight

Spacecraft formation flight is the concept of having multiple spacecraft flying in proximity in order to accomplish a mission. Spacecraft formations can be used to enhance a number of missions, including:

- Remote Sensing
- Robotic Assembly
- Fractionated Spacecraft Architectures
- Large Satellite Arrays

Remote Sensing

Remote sensing stands as one of the major advantages of space [3]. However, monolithic remote sensing satellites have reached the maximum size and weight allowed by launch vehicles [4]. By using a distributed spacecraft architecture, scientists can achieve higher resolution images by taking advantage of the distributed, reconfigurable constellation.

For example, synthetic aperture radar (SAR) has proven to be a revolutionary technology for scientific, commercial, and military applications. However, "the accuracy of present spaceborne SAR interferometers is severely limited by either temporal de-correlation associated with repeat pass interferometry (e.g. Envisat, ERS) or by the physical dimensions of the spacecraft bus that constraints the achievable baseline length (e.g. X-SAR/SRTM Shuttle Topography mission). These limitations may be overcome by means of two spacecrafts flying in close formation building a distributed array of sensors, where the two antennas are located on different platforms" [5].

Fractionated Architectures

Tightening budgets, increasingly demanding missions, and a shifting geopolitical landscape have given rise to a potential new type of spacecraft architecture. Instead of putting all of the components of a satellite in one monolithic unit, individual components of a spacecraft could be flown in proximity. In this architecture, components can be added or replaced by launching a new component, instead of an entirely new satellites. This architecture can make the system resilient to technical failures, funding fluctuations, changing missions, and even physical threats [6].

Robotic Assembly

As the complexity of space missions architectures increase, launch vehicle limitations will limit the size of monolithic spacecraft. If fractionated architectures are not an option for a particular space mission, then assembly of the spacecraft on-orbit will be required. Instead of relying on astronauts for assembly of these systems in orbit, robotic assembly stands as a potentially cheaper and safer method of assembling these satellites [7].

An example of a spacecraft mission that uses robotic assembly is the X-ray Evolving Universe Spectroscopy (XEUS) satellite. XEUS is designed to search for large black holes created about 10 billion years ago at the beginning of the universe [8]. During the middle of the mission, the satellite will dock with the International Space Station in order to replace one part of the satellite with a new part [9].

The University of Southern California has developed a method for robotic assembly that involves using retractable tethers in a microgravity environment to assemble the structure in a particular configuration [10]. Tethers are a lighter alternative to a truss structure and allow limited reconfiguration of the system. However, the tethers must be kept in tension, which complicates the concept of operations.

Carnegie Mellon University is developing the Skyworker system. The concept of Skyworker is that a machine would "walk" on an existing structure in order to add new pieces to the existing structure [11]. This system eliminates the problems associated with free-flying assembly methods, but requires significant infrastructure and may not be able to service all parts of a structure.

Large Satellite Arrays

Satellite arrays with a large number of satellites could potentially be used to solve some of the planet's strategic problems, such as energy shortages. Solar power satellites have the potential of delivering megawatts of power from space [12]. These satellite arrays would require hundreds of satellites that would have to be either assembled in space or flown in a formation.

1.1.2 Electromagnetic Formation Flight

In order to maintain a spacecraft formation, each satellite must have an actuator in order to maintain its position and attitude relative to the other satellites. Thrusters, the conventional technology used for relative and absolute satellite position and attitude control, have limited fuel and can cloud sensitive optics. Tethers are another option for maintaining a satellite formation; however, they are difficult to deploy, limit the orientations of the structure, and require physical attachment of components. Electrostatic forces, which are generated by charging up spacecraft, could be used for formation flight; however, this introduces a risk of arcing to the satellites [13].

Electromagnetic formation flight is an enabling technology that would remove the limitations on spacecraft formations due to thruster or tether requirements. Electricity can be generated via solar panels, giving the system a replenishable power source. Electromagnets will not cloud optics, nor do they require physical attachment between spacecraft.

Satellite Array Control

The primary function of electromagnetic formation flight is to control and maintain a spacecraft formation, making it an enabling technology for satellite formation flight (Section 1.1.1). Using electromagnets, spacecraft can control their position and attitude *relative to other spacecraft.* Since all forces on the satellite array are internal, the center of mass of the system does not change.

Electromagnetic formation flight has several major advantages over other formation flight options. Unlike thrusters, EMFF does not require an expendable propellant; instead, electricity is used. That electricity can be harvested from the sun via solar panels or produced by other means. This means EMFF can theoretically operate so long as the satellite has sufficient power. In reality this means that the "propellant," electricity in this case, may no longer be the limiting factor for mission lifetime. Another advantage of EMFF for formation flight is the fact that EMFF is contamination-free. This makes EMFF a potential candidate for formation flight missions where contamination of optical sensors is a concern [14], [15].

The idea for electromagntic formation flight was originally motiviated by the engineering challenges of the Terrestrial Planet Finder (TPF). The primary goal of the Terrestrial Planet Finder was to detect Earth-like planets located in habitable orbits around solar type stars using a nulling infrared interferometer [16]. This architecture requires a constantly rotating formation of satellites. Different methods have been proposed, including tethers and plasma propulsion systems [17]. Propulsion systems will limit the mission lifetime due to fuel constrictions. Also, plumes from the thrusters could also cloud the sensitive optics required for such a mission. EMFF could solve both of these issues, providing a centripetal force without any consumables or optical impingement.

Satellite thrusters are not the only method for formation flight. As discussed in Section 1.1.1, tethers and physical attachments have also been proposed for the specific formation flight mission of robotic assembly. The advantage EMFF holds is that physical attachment between spacecraft is not required. Tethered formation flight requires the tethers to be attached to each spacecraft and deployed. This means the system must either be fully assembled on the ground, which defeats some of the advantages of spacecraft formation flight, or assembled on-orbit, which is not possible with tethers alone. Tether anchors cannot not be reconfigured on-orbit, reducing the flexibility of the system.

Physical attachments for robotic assembly, such as the Skyworker system [11], require a structure that allows another system to traverse upon it. While this allows the system to be re-configurable, this also requires building the system robustly enough that a system could traverse the structure of the satellite. This structural robustness, given the high cost of launch opportunities, may severely stunt the growth of an on-orbit assembly project. EMFF for robotic assembly would still require extra infrastructure for the coils; however, given the close proximity of movement for robotic assembly, the infrastructure required may be relatively small.

The next generation X-ray telescope, or Gen-X, serves as an excellent case study for the advantages of EMFF. The Gen-X system will require focal lengths greater than 50 m. In a trade study for Gen-X, Ahsun, Rodgers, and Miller compared a structurally connected X-ray telescope (SCX), a propellant-based spacecraft formation (PFF), and an EMFF-based spacecraft formation [18]. The authors found that an EMFF-based system is less massive than a SCX or PFF based system under certain mission conditions. They also found that EMFF becomes even more advantageous when satellite movement is required. The advantages of EMFF also increase as the mass of the detector module increases.

EMFF is not without its own disadvantages for formation flight. First, to achieve actuation authority at significant distances, large amounts of current are required. While these currents can be achieved using superconducting electromagnets, these electromagnets then need to be cooled. To realize the advantages of an extended mission life and no optical contamination, a closed-loop cooling system is required. While work has been done on these systems, they are not at the same technological readiness level as EMFF [19]–[21].

Wireless Power Transfer

Wireless power transfer is another potential benefit of electromagnetic formation flight. If the satellites use electromagnets with an alternating current, an alternating magnetic field can be generated. This alternating magnetic field can be used to power or charge other spacecraft without a physical connection.

Consider a satellite architecture where only one satellite in a formation is responsible for power generation. This satellite could have solar panels, a Radioisotope Thermoelectric Generator (RTG), or other source of power. It could then transfer some of its power to satellites flying in formation. This would mean the other satellites would not need a power generation source since the power can be supplied to them remotely. This allows the other satellites to be more specialized for a different purpose. This architecture is illustrated in Fig. 1-2.

Other Applications

Another use of electromagnetic formation flight is maintaining non-Keplerian motion of a satellite. Although the center of mass of a satellite formation using electromagnetic formation flight would move in a Keplerian manner, each satellite would appear to move in a manner that was not governed by Kepler's laws of motion. Since most satellite orbit estimators rely on Keplerian assumptions, flying in a non-Keplerian

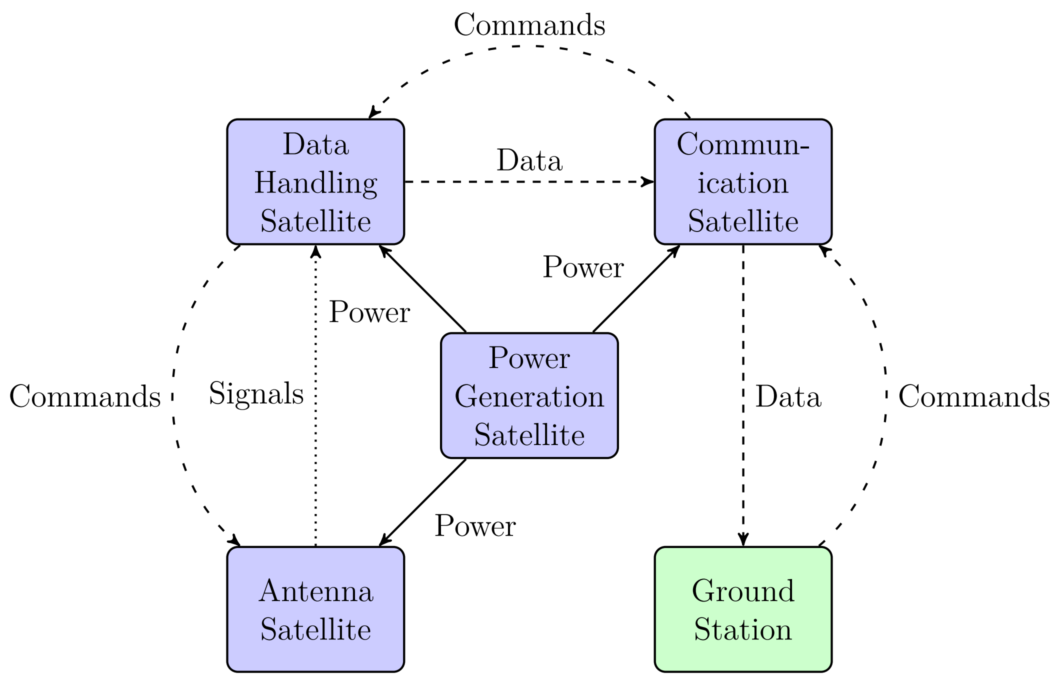


Figure 1-2: Example of a Mission Architecture That Uses Wireless Power Transfer

manner could defeat conventional satellite tracking methods. This would make the satellite difficult to track, which is advantageous from a satellite defense perspective.

Fig. 1-3 illustrates a potential orbit where EMFF is used to maintain a non-Keplerian orbit. Satellites A and B maintain a spin about the cross-track direction, keeping formation together using an attractive electromagnetic force between the two. The system center of mass obeys Kepler's laws, but Satellite A and Satellite B individually appear to move in a manner that is not predictable under Keplerian assumptions.

The electromagnets used for electromagnetic formation flight can also be used to protect the spacecraft from energetic spacecraft radiation. Energetic radiation can damage spacecraft electronics, resulting in data corruption, satellite resets, or even permanent satellite failure [22]. However, active magnetic shielding can be used to deflect the radiation, thus reducing the chances of radiation affecting the spacecraft [23]. By using the electromagnets on the spacecraft, the spacecraft can generate an active magnetic field that would act as a radiation shield for the spacecraft.

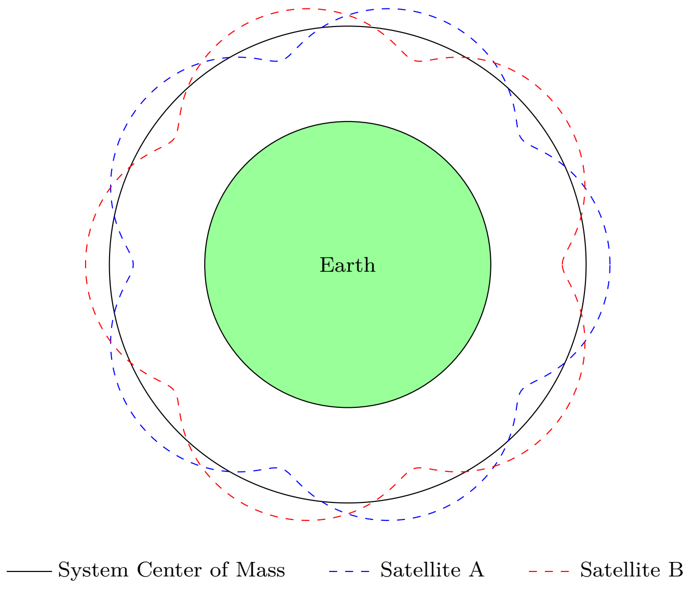


Figure 1-3: Example of Non-Keplerian Orbits Using EMFF (Not To Scale)

1.1.3 Resonant Inductive Near-Field Generation System

The Resonant Inductive Near-Field Generation System (RINGS) is a testbed that will be used to demonstrate electromagnetic formation flight in a space. RINGS is sponsored by the Defense Advanced Research Project Agency (DARPA) and is lead by Professor Raymond Sedwick of the University of Maryland. The RINGS are designed to be integrated into the Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) testbed, which has been onboard the International Space Station since 2006 [24]–[26]. The combined RINGS and SPHERES system can be seen during an Reduced Gravity Aircraft (RGA) flight in Fig. 1-4. The RINGS system consists of two units; each unit is an AC electromagnet with supporting control avionics. The specifications of RINGS can be seen in Table 1.1

The RINGS have two general modes: Electromagnetic Formation Flight and Wireless Power Transfer. In electromagnetic formation flight mode, the units generate a synchronized alternating magnetic field that attracts, repels, and torques the units. Both the amplitude and phase of each unit can be independently controlled. In Wireless Power Transfer mode, one unit generates an alternating magnetic field while the other "receives" the power by placing a load resistor in line with the circuit, dissipating

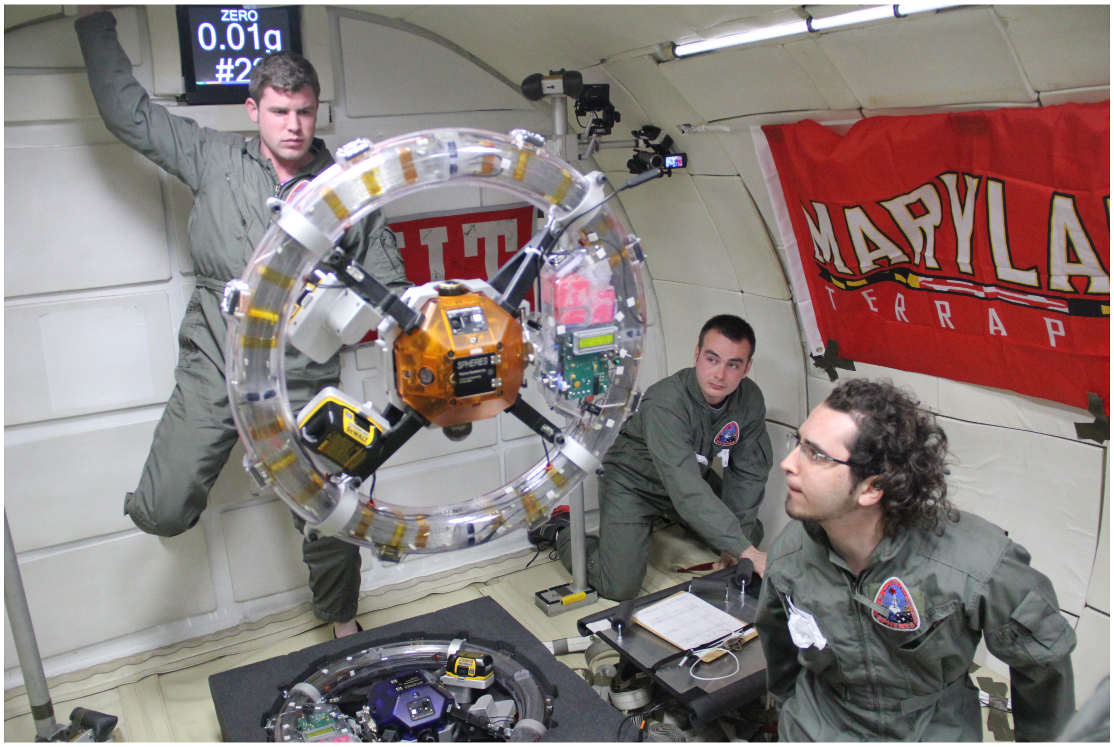


Figure 1-4: RINGS and SPHERES During an RGA Flight

Table 1.1: RINGS Technical Specifications

Description	Value
Number of Units	2
Mean Coil Diameter	0.64 m
Number of Turns	100
Max Current (RMS)	18 A
EMFF Frequency	83 Hz
WPT Frequency	460 Hz
Unit Mass	17.1 kg

the transferred power. While wireless power transfer is a promising new technology, the focus of this work will be on electromagnetic formation flight relating to RINGS.

1.1.4 RINGS Control

To date, all of the work on EMFF control has assumed a fully controllable dipole. In a six degree of freedom environment, this requires three orthogonal coils (or at least three coils where no two are parallel). This allows the magnetic dipole to be pointed in any direction by controlling the amount of current in each coil. For EMFF testbeds on the ground, only two coils were required for a controllable dipole, since the system is limited to three degrees of freedom.

Despite operating in a six degree of freedom environment, RINGS only has one coil per vehicle. This means the dipole is not fully controllable; instead, only the magnitude and the polarity of the coil are controllable by the electromagnet itself. To control the direction of the dipole, the system must be physically rotated. This makes the system non-holonomic, and subsequently makes the control of RINGS more difficult than a holonomic EMFF system.

RINGS control is also difficult because of the proximity of the coil. Previous EMFF work assumed the coils were at least several coil diameters apart. At this distance, the first term of the Taylor series expansion of the force and torque equations is sufficient for describing the system behavior. This "far-field" model is used in nearly all of the previous EMFF research. However, the RINGS will be confined to the SPHERES working volume. This means the RINGS will rarely ever be more than a few coil diameters apart. Therefore, the "near-field" model will need to be used to describe the motion of the RINGS. There is no closed form solution of the "near-field" force and torque model as a function of attitude and separation, making controller development even more difficult. Additionally, the "near-field" model is highly nonlinear with satellite position and attitude.

1.1.5 Dynamic Programming

Because of the unique and difficult dynamics presenting by RINGS, dynamic programming was chosen as the control method for this work. While other control methods can be used [27], dynamic programming was chosen for two main reasons. First, dynamic programming offers a robustness for nonlinear systems, something that many other control methods do not. Secondly, modeling the RINGS system for a dynamic program requires making less assumptions than other types of controllers.

While dynamic programming is used often in simulation and optimization, it is rarely used for embedded systems. This is most likely because of the large amount of storage space required for the lookup tables generated by dynamic programming. However, this does not mean dynamic programming cannot be applied to embedded systems. The next generation of aircraft collision avoidance uses dynamic programming to advise pilots of the optimal action when faced with a potential collision [28]. Neural dynamic programming has also been applied to the tracking, control, and trimming of helicopters [29]–[31].

While Chapters 3 and 4 are specific to the RINGS system, Chapter 2, which discusses the theory behind dynamic programming and approximate dynamic programming, and Chapter 5, which discusses implementing dynamic programming controllers, can be applied to any embedded system. This means that dynamic programming is not necessarily limited to RINGS or EMFF. Given the advances in computing power and data compression, dynamic programming may become a more prevalent option for control engineers.

1.2 Previous Work

The previous work discusses in this section pertains to electromagnetic formation flight. The limited amount of previous work on applying dynamic programming to embedded systems is discussed in Section 1.1.4. The previous work completed on mass property identification is discussed in Section 6.1.

In a series of papers, Hashimoto, Sakai, Ninomiya, *et al.* introduce the idea of using superconducting electromagnets to maintain a satellite formation in low earth orbit (LEO) [32], [33]. Since then, the Massachusetts Institute of Technology's Space Systems Laboratory has taken the initiative on EMFF. While a significant amount of work has been done examining EMFF at the systems level [15], [34]–[39], the focus of this literature review will be on the technical aspects of EMFF, namely modeling and simulation, testbed, and control.

1.2.1 EMFF Dynamics

In the thesis by Elias, a non-linear model of EMFF dynamics was developed [40]. This analysis assumed each satellite had a fully controllable dipole. The satellites were also assumed to be separated enough such that the "far-field" assumption held. This "far-field" model assumes the coils are far enough apart such that only the first term of the Taylor series approximation of the force between two coils is sufficient to describe their interaction. The model derived by Elias accounts for electromagnetic forces as well as reaction wheels, the spacecraft bus, and the coupling between the reaction wheels and the spacecraft.

In a thesis by Schweighart, an in-depth model of the intra-satellite forces generated by electromagnets is developed [41]. A "near-field" model is developed for the first time, which describes the forces and torques produced by electromagnetic coils regardless of separation distance. From the "near-field" model, the "far-field" model is then derived and compared against the "near-field" model. Methods for solving the equations of motion are then discussed.

1.2.2 EMFF Testbeds

Under the direction of Professor David Miller, the Massachusetts Institute of Technology's Space Systems Laboratory (MIT SSL) has developed several EMFF testbeds. Elias developed an 1 Degree of Freedom (DoF) EMFF testbed which rode on a linear air track [40]. The system consisted of a permanent magnet on a low friction carriage, which was allowed to move along the linear track, and an electromagnet fixed

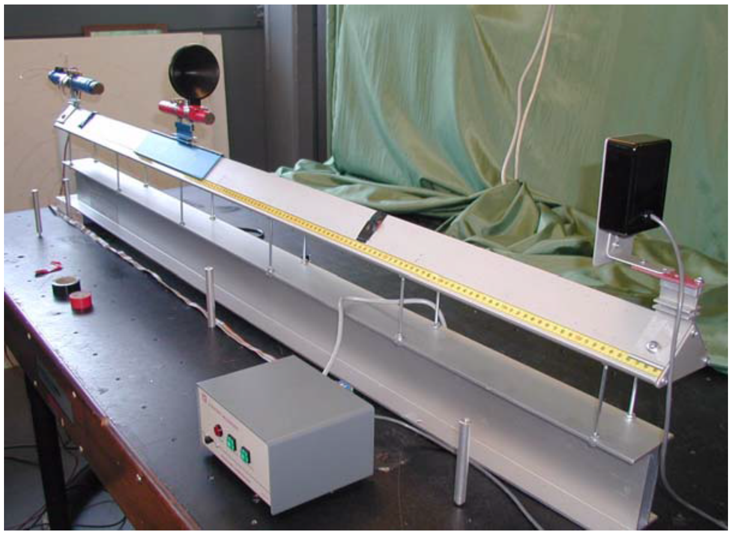


Figure 1-5: Linear Track EMFF Testbed [40]

at the end of the track. The current in the electromagnet was manipulated in order to control the position of the permanent magnet. An ultrasonic ranging device was used to provide position data. The poles of the system could be changed by changing the inclination of the air track. This testbed is seen in Fig. 1-5.

The next testbed was 3 DoF testbed consisting of two vehicles, each with two orthogonal high temperature superconducting electromagnets [42]. Each vehicle floated on a planer surface using air bearings. The first testbed was originally designed by undergraduates [43], with additional work completed by graduate students [1]. The testbed was able to demonstrate position holds as well as the ability to follow a trajectory using EMFF [44]. One of the vehicles, along with Professor Miller, is seen in Fig. 1-6[1].

In an investigation of the use of non-superconducting electromagnets for EMFF, Sakaguchi developed the µEMFF testbed [45]. This testbed consisted of two traditional (non-superconducting) electromagnetic coils. One coil was fixed to a servo motor while another was cantilevered from a bar that was allowed to rotate about an air bearing. The attitude of the cantilevered coil was also controlled via a servo motor. The system was able to demonstrate EMFF without the use of superconducting electromagnets. The µEMFF testbed is seen in Fig. 1-7.

[1]Credit: William Littant/MIT Photo

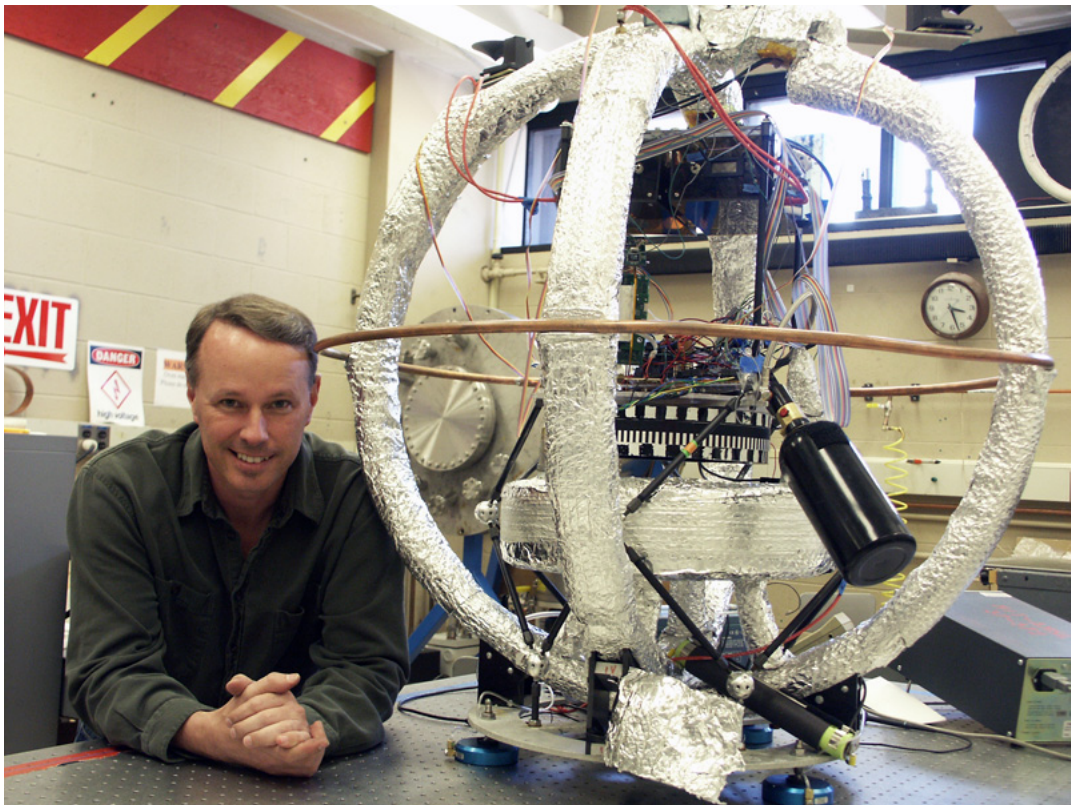
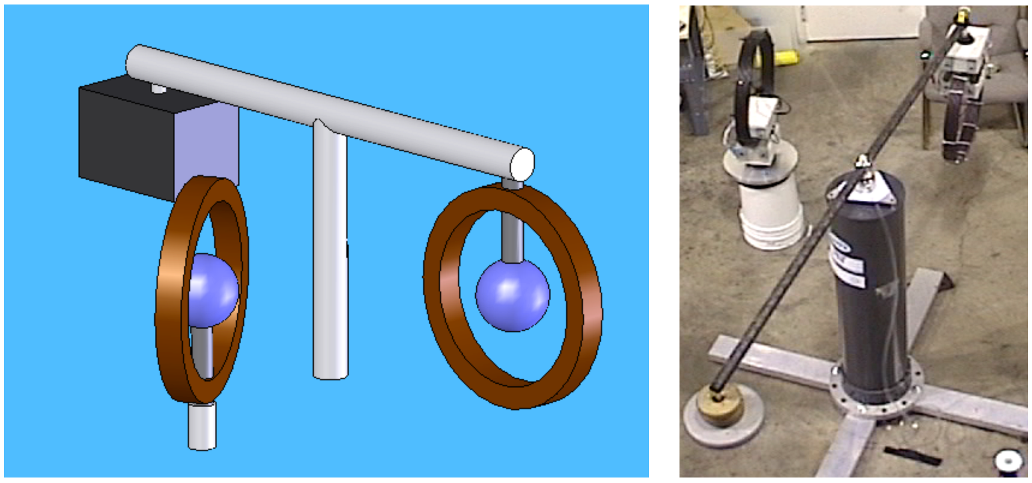


Figure 1-6: Professor David Miller with the 3 Degree of Freedom EMFF Testbed



(a) Conceptual Rendering. (b) Actual System.

Figure 1-7: µEMFF Testbed [45]

1.2.3 EMFF Control

The first investigation into the control of an electromagnetic formation flight was conducted by Hashimoto, Sakai, Ninomiya, *et al.* [32], [33]. In their analysis, a two satellite co-planer formation is considered. These satellites maintain an inertially fixed separation distance. The effects of disturbance torques and forces are analyzed for low earth orbit. A phase-shift controller is presented to control the satellite positions.

Kong examined the controllability of an EMFF system [35]. The analysis models each electromagnetic field as a stationary multi-pole. The analysis showed that the system is able to control all degrees of translational motion from small perturbations from a nominal position.

In a thesis [46] and series of papers [37], [47]–[49] by Ashun, a non-linear control law is developed for EMFF systems. The controller is designed for a system with any number of satellites. Ashun also shows that, under general assumptions, a multi-satellite formation can be stabilized [47]. Legendre Psedudospectral Methods are used to generate the optimal trajectories, while adaptive control is used to account for the disturbances of Earth's magnetic field and the errors of the "far-field" EMFF model [48], [49].

In a thesis by Ramirez Riberos, a method for controlling an EMFF system using decentralized control is presented [50]. The control method presented uses a "token" approach where one satellite, which has the "token", is allowed to actuate at a time. The control problem is split into a "high level" and "low level" problem. In the high level problem, the order in which the satellites actuate is determined using dynamic programming. In the lower level problem, a Legendre pseudospectral decomposition approach to find the appropriate actuator control is used.

In a series of papers by Schweighart [39], [51], [52], control methods are presented for a mutli-satellite EMFF formation. Control methods for spinning a satellite array using EMFF are presented [51]. These maneuvers are significant because they would be required for a sparse aperture satellite formation architecture. For other maneuvers, a method of using Newton's method combined with the continuation method was presented [39], [52]. Since the EMFF dipole solutions are over-determined, a method for choosing the dipole strengths is also presented [52].

In a paper by Cai, Yang, Zhu, *et al.*, methods for finding the optimal trajectory for a satellite formation reconfiguration using EMFF is developed [53]. In their formulation, the objective is to minimize the power required to reconfigure a satellite formation using EMFF. Power was chosen as the objective in order to minimize the strain on an EMFF cooling system. Gauss Pseudospectral Methods are used to determine the optimal trajectories. In order to implement the trajectories, a trajectory tracking controller is presented that uses both output feedback feedback linearization on an inner control loop and adaptive sliding mode control on an outer loop. Theoretical convergence guarantees for the control are also presented. Simulations are also shown. The simulations show proper trajectory tracking but also show control "chatter" in the dipole solutions.

In a series of papers by Wawrzaszek and Banaszkiewicz, the control of a two satellite planer spinning array is discussed [54], [55]. A linearized model is developed using the "near-field" model of force is developed for two satellites spinning in an array, with the coils axially aligned along the line connecting the two satellites' center of mass. Stability analysis showed this system is unstable without feedback control. A linear controller is presented that stabilizes the system for a range of disturbances. A three body planer formulation where a third coil is centered between two coils is also presented. The system is shown to be neutrally stable in low earth orbit without control. Using a linear controller, the system is demonstrated to be stabilized for a limited set of disturbances.

In a series of papers by Zhang, Yang, Zhu, *et al.*, a nonlinear control method for docking satellites in six degrees of freedom using EMFF is presented [56], [57]. The "far-field" force and torque model is used to develop a nonlinear system for control. The satellites are assumed to have independent angular control, so only translational control is considered. A controller is developed that uses a linear quadratic regulator around a pre-designed trajectory. The controller was shown in simulation to control a docking of two satellites. The paper is novel in that it is the first to address docking of EMFF satellites, but uses the "far-field" model for control development despite operating the "near-field". The paper does not state whether the "far-field" or "near-field" model was used for the dynamics of the simulation.

In a paper by Zeng and Hu, a finite time controller for translational dynamics is presented [58]. A standard "far-field" model is developed and is used for a planer two satellite system. Convergence of the controller in finite, rather than asymptotic, time is proven. The controller was shown in simulation to converge in finite time for multiple scenarios.

Chapter 2

Dynamic Programming

The idea of dynamic programming was conceived by Richard Bellman. In his work, Bellman writes "An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [59]. This is known as *Bellman's Principle*, and is the central idea to dynamic programming. More simply stated, if an trajectory is known to be optimal, then any sub-trajectory of that trajectory is also optimal.

To illustrate this idea, consider the process illustrated in Fig. 2-1. Each circle is a state, while each line is a path with an associated transition cost. The objective is to move from node A to node F with minimum cost. If the optimal trajectory from A to F is A-C-D-F, then Bellman's principle states that the optimal trajectory from C is C-D-F.

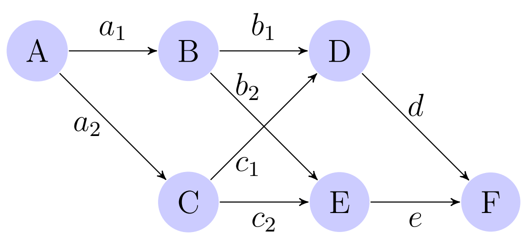
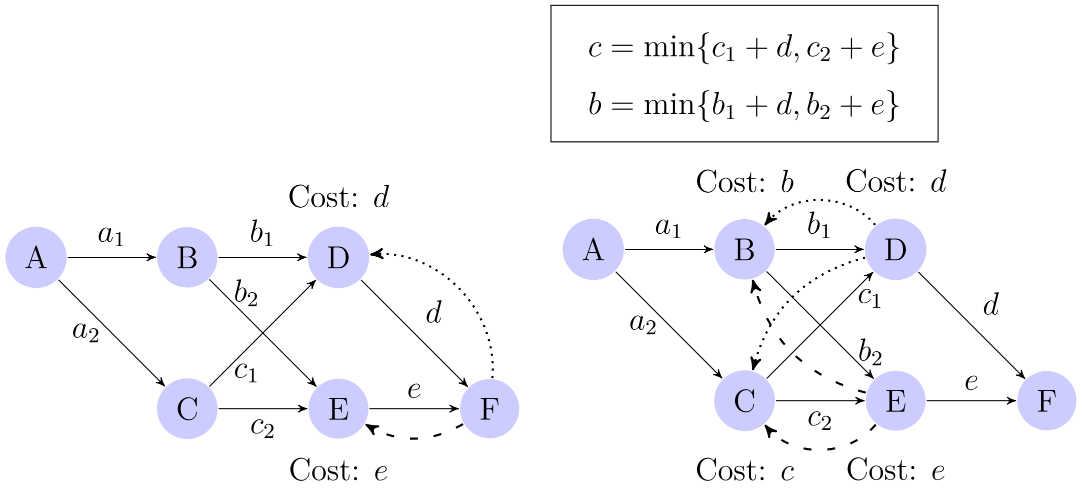


Figure 2-1: Simple Markov Process

The power of dynamic programming lies in the fact that the optimal trajectory can be found by working backwards from node F. Any trajectory that reaches node F must go through either nodes D or E. The "cost-to-go" from node D to node F is known, as well as the cost-to-go from node E to node F. This step is illustrated in Fig. 2-2(a). Now, the cost-to-go from node B to node F through node D is simply the summation of the transition cost from B to D and the cost-to-go from node D to F. The cost-to-go from B to F going through node E can also be computed in the same manner. The optimal trajectory from node B is simply the trajectory that results in the lowest cost-to-go. The cost-to-go for node C can be found in a similar manner. This step is illustrated in Fig. 2-2(b).



(a) Step 1. (b) Step 2.

Figure 2-2: Dynamic Programming Analysis of a Simple Markov Process

With the cost-to-go to node F from both B and C known, the cost-to-go from node A to F through node C is the summation of the transition cost from A to C and the cost-to-go from node C to F. This is significant because the cost of all paths from node C to F are not required; rather, only the cost-to-go from C to F is required. The path from A to F through B can also be evaluated in such a manner. The optimal path from A is whichever path that results in minimum cost, determined by Eq. (2.1).

$$a = \min\{a_1 + b, a_2 + c\} \tag{2.1}$$

If $a_2 + c < a_1 + b$, then the optimal path from A is the path to C. The optimal path from C is found in a similar manner, solving $\min\{c_1 + d, c_2 + e\}$. If $c_1 + d < c_2 + e$, then the optimal path from C is to D and then F. This example illustrates the fundamental principles of dynamic programming.

While it is easy to evaluate all paths for Fig. 2-1, typical dynamic programming problems have many more states, making the possibility of evaluating all paths nearly impossible. The power of dynamic programming is that every path does not need to be evaluated. Instead, the optimal path can be methodically found in reverse, reducing the number of paths evaluated.

2.1 Fundamentals of Dynamic Programming

Let $\mathbf{x}_k$ be the state of a system at time k. The state equation of a system, seen in Eq. (2.2), computes the propagation of state $\mathbf{x}_k$ one time step based on the input $\mathbf{u}_k$.

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) \tag{2.2}$$

In a traditional control sense, the objective of a controller is to minimize the cost function

$$J_0\left(\mathbf{x}_0\right) = \sum_{k=0}^{N} g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right) \tag{2.3}$$

where $g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right)$ is the cost of transitioning from state $\mathbf{x}_k$ to state $\mathbf{x}_{k+1}$ using input $\mathbf{u}_k$. The cost at time k can be written as

$$J_k\left(\mathbf{x}_k\right) = g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right) + J_{k+1}(\mathbf{x}_{k+1}) \tag{2.4}$$

where

$$J_{k+1}(\mathbf{x}_{k+1}) = \sum_{i=k+1}^{N} g_i\left(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}\right) \tag{2.5}$$

$J_{k+1}(\mathbf{x}_{k+1})$ is called the cost-to-go from state $\mathbf{x}_{k+1}$, since it is encapsulates the remaining cost of the trajectory. Bellman's Principle states that if the optimal cost-to-go at time $k + 1$, written as $J^*_{k+1}(\mathbf{x}_{k+1})$, is known for all $\mathbf{x}_{k+1}$, then the optimal cost-to-go at time k can be solved using Eq. (2.6).

$$J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U_k} g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\right) + J_{k+1}^*(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)) \tag{2.6}$$

Eq. (2.6) is known as Bellman's Equation and is the basis of dynamic programming. Dynamic programming finds the optimal cost-to-go for a system by using Bellman's Equation backwards on a trajectory. If the terminal costs $J_N^*(\mathbf{x}_N)$ are known for all $\mathbf{x}_N$, then $J_{N-1}^*(\mathbf{x}_{N-1})$ can be found for all $\mathbf{x}_{N-1}$ by using Bellman's Equation. Once $J_{N-1}^*(\mathbf{x}_{N-1})$ is known for all $\mathbf{x}_{N-1}$, then $J_{N-2}^*(\mathbf{x}_{N-2})$ can be found for all $\mathbf{x}_{N-2}$. This process can be repeated until $J_0^*(\mathbf{x}_0)$ is known for all $\mathbf{x}_0$. With the optimal cost-to-go known for all states and times, the optimal input of the system at $\mathbf{x}_k$ can be found by solving Eq. (2.7)

$$\mathbf{u}_k^* = \arg\min_{\mathbf{u}_k \in U_k} g_k\left(\mathbf{x}_k, \mathbf{u}_k, f(\mathbf{x}_k, \mathbf{u}_k)\right) + J_{k+1}^*(f(\mathbf{x}_k, \mathbf{u}_k)) \tag{2.7}$$

The advantage of dynamic programming is that it computes the optimal trajectory for every possible state, meaning dynamic programming converges on the global minimum of Eq. (2.3). Other control methods, such as Gauss Pseudospectral methods, often do not have the global minimum convergence guarantees that dynamic programming offers. The convergence guarantees come at the cost of having to compute and store the cost-to-go for every possible state at every time step. While this is done off-line, this can still be cumbersome in terms of both required computational power and required memory. The amount of computation and memory for a dynamic program increases exponentially with the number of states. This is known as the "curse of dimensionality".

One way to reduce the amount of computation and memory required is to formulate the control problem as an infinite horizon problem. In an infinite horizon formulation, the system is assumed to be run for an infinite amount of time. This makes the optimal cost-to-go time invariant, which removes the dimension of time from formulation. The cost-to-go for an infinite horizon formulation is seen in Eq. (2.8).

$$J\left(\mathbf{x}_0\right) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right) \tag{2.8}$$

In anticipation of the large number of states for dynamic programming formulations of satellite control problems, the remainder of this work will focus on infinite horizon dynamic programming.

2.2 Dynamic Programming Formulation Types

If an infinite horizon dynamic program is not formulated correctly, the cost-to-go described by Eq. (2.8) can become unbounded, which renders the program unsolvable. There are three general formulations for infinite horizon dynamic programming that provide bounds on Eq. (2.8). They are:

- Discounted Cost
- Stochastic Shortest Path
- Average Cost

These formulations are described in Sections 2.2.1 to 2.2.3, respectively.

2.2.1 Discounted Cost

In a discounted dynamic programming formulation, the cost function seen in Eq. (2.8) is adapted by adding a *discount factor*, $\alpha \in (0, 1)$. Using the discount factor, Eq. (2.8) becomes

$$J\left(\mathbf{x}_0\right) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} \alpha^k g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right) \tag{2.9}$$

Assuming that $g_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}\right) \in \mathbb{R}$ for all $\mathbf{x}_k, \mathbf{x}_{k+1} \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbf{u}_k$, Eq. (2.9) will converge finitely [60].

This formulation has the best convergence guarantees out of the three methods discussed in Section 2.2. It also requires the least amount of adaptation when converting a finite horizon problem into an infinite horizon problem. The downside of a discounted cost formulation is that the trajectories governed by the discounted cost formulation, when compared against trajectories from a stochastic shortest path or average cost formulation, may be sub-optimal.

A discounted cost formulation generally works well with a linear quadratic regulator (LQR) cost function, seen in Eq. (2.10). An LQR cost function is quite common in the control field due to its desirable properties of convexity, existence and simplicity of the first derivative, and performance guarantees for linear systems. Let $\mathbf{x}_t$ be the target state of the system. The LQR cost function takes the form

$$g_k\left(\mathbf{x}_k, \mathbf{u}_k\right) = \left(\mathbf{x}_k - \mathbf{x}_t\right)^\top Q \left(\mathbf{x}_k - \mathbf{x}_t\right) + \mathbf{u}_k^\top R \mathbf{u}_k \tag{2.10}$$

where Q and R are positive definite weighting matrices corresponding to state error and control input, respectively. An LQR style cost function would work well with a discounted dynamic programming formulation for regulating a system around a target state. Assuming that all possible states and controls are in the set of real numbers, then the LQR cost function will always be bounded, meaning all trajectories in a discounted DP formulation will have a finite cost.

2.2.2 Stochastic Shortest Path

Consider the Markov process illustrated in Fig. 2-3. In this process, the state of the system is allowed to travel freely between nodes A,B,C,D, and E. However, once the state of the system reaches node T, the system will remain at node T for the remainder of time.

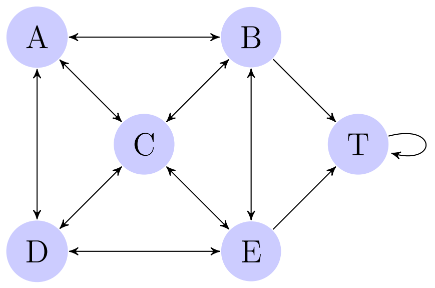


Figure 2-3: Stochastic Shortest Path Markov Process

In a stochastic shortest path dynamic programming formulation, the Markov process is augmented with a state called the *termination state*, T. If the transition costs are defined as

$$g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}) = \begin{cases} g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}), & \mathbf{x}_k \notin T \\ 0, & \mathbf{x}_k \in T \end{cases} \tag{2.11}$$

and there is a positive probability of reaching the termination state from any given state, then the cost-to-go will remain bounded [60].

As the name suggests, this type of formulation is best for driving a system to a target set of states via the shortest path. The definition of path will drive the form of the transition cost function. If this objective is to drive the system to a state via the shortest spatial route, then the cost function will be driven by the distances between nodes. If a minimum-time path is desired, then the transition cost will be driven by the transition time between nodes.

The advantage of stochastic shortest path formulations is that it finds the true shortest path between any node and the termination state. Discounted cost formulations, when used to find the shortest path to target states(s), may not converge on the actual shortest path. This is because the discount factor reduces the importance of the cost-to-go from a particular node, increasing the weight of transition costs on the state's cost-to-go. This may result in convergence on a trajectory that is not truly the shortest path.

The disadvantage of stochastic shortest path formulations is the potential existence of improper policy. An improper policy exists when a trajectory from a certain state never reaches the termination state. This will cause the trajectory cost to become unbounded, resulting in an infinite cost-to-go for certain states. To ensure that the cost-to-go is bounded, there must be a positive probability that the system will reach the termination state from any state.

2.2.3 Average Cost

The average cost formulation can be considered an extension of the stochastic shortest path formulation. Instead of a termination state, there is a *recurring state*. The goal of the dynamic program is to reduce the average cost of the system, where the cost is reset when the system reaches the recurring state. Unlike the stochastic shortest path formulation, the system may leave the recurring state. The cost-to-go of an average cost formulation is bounded so long as there is a positive probability of reaching the recurrent state from any given state of the system.

2.3 Approximate Dynamic Programming

Despite continual advances in computers, many dynamic programs cannot be directly solved due to the shear number of states. However, recent advances in approximate dynamic programming provide approximate solutions for the cost-to-go for a system that cannot be solved using dynamic programming. There are two general ways to solve dynamic programs of large scales: state aggregation and cost approximation. State aggregation is analogous to "lumping" similar states together into an aggregate state, reducing the total number of states of the formulation. Cost evaluation involves approximating the cost function with an basis function. This requires projecting the cost vector into a subspace.

2.3.1 Aggregation

Let C be the set of all possible states of the system. For dynamic programming to be feasible, the state space must be reduced to a finite number of states. Therefore,

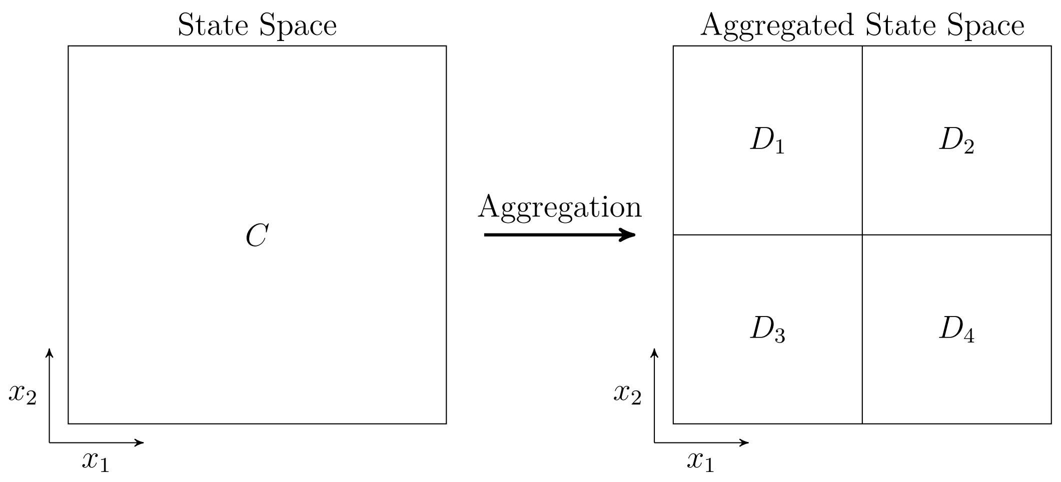


Figure 2-4: Illustration of State Aggregation

subsets of points in C are aggregated to make mutually disjoint subsets of C. The state of the system of this formulation becomes $i = 0, ..., n$, while $D_i \subset C$, $i = 0, ..., n$ represent the set of states that the discrete state i represents. The concept of aggregation is illustrated in Fig. 2-4.

This type of formulation transforms a deterministic formulation into a stochastic one. When the system is at state i, the actual state of the system is one of the states in D_i. Let $p_{ij}(\mathbf{u})$ be the probability of transitioning from aggregate state i to aggregate state j when input $\mathbf{u}$ is applied to the system. Using the framework described in Fig. 2-5, the aggregate state transition probabilities can be computed using Eq. (2.12).

$$p_{ij}(\mathbf{u}) = \sum_x d_{ix} \sum_y p_{xy}(\mathbf{u}) a_{yj} \tag{2.12}$$

There are many ways to define the dissaggregation and aggregation probabilities. The simplest method is called hard aggregation. The hard aggregation technique is equivalent to a nearest-point approximation, where the aggregation probabilities are assigned based on singular aggregate state that encompasses the point. Hard aggregation can be described using Eqs. (2.13a) and (2.13b) and is illustrated in Fig. 2-6(a).

$$d_{ix} = \begin{cases} 1, & \mathbf{x} \in D_i \\ 0, & \mathbf{x} \notin D_i \end{cases} \tag{2.13a}$$

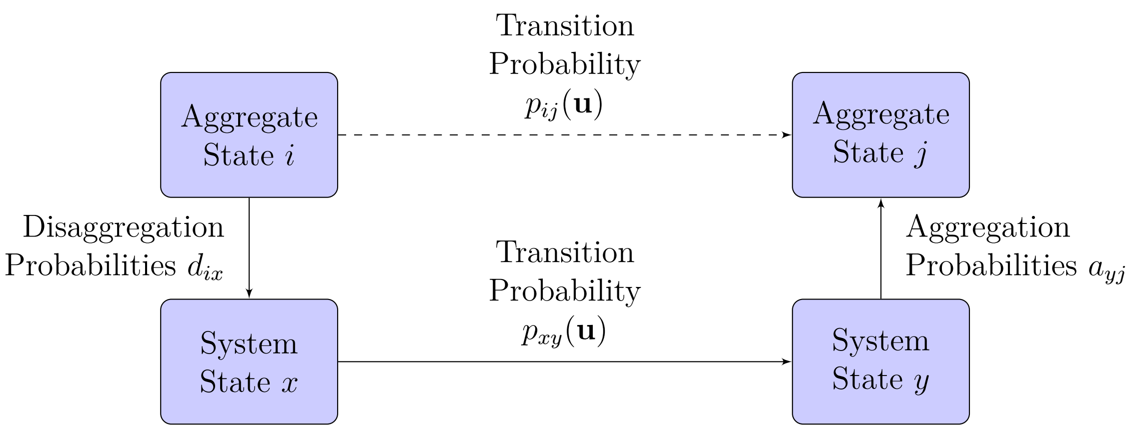
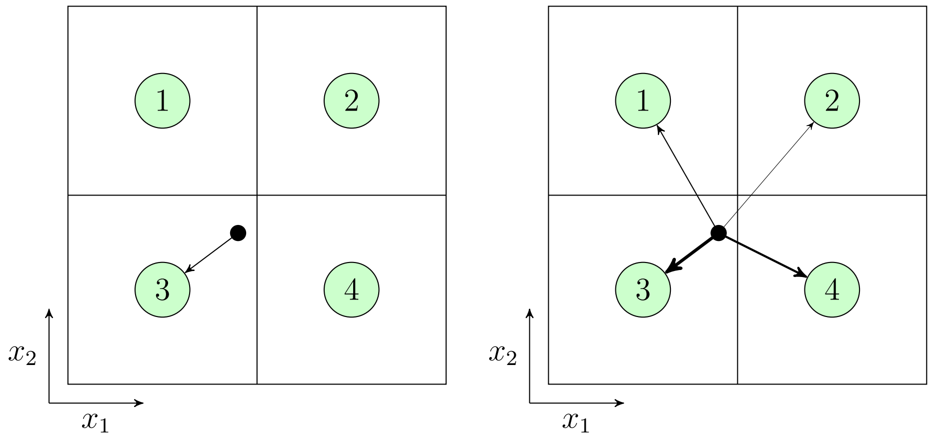


Figure 2-5: Aggregation Formulation [60]



(a) Hard Aggregation. (b) Soft Aggregation.

Figure 2-6: Illustration of Aggregation Techniques

$$a_{yj} = \begin{cases} 1, & \mathbf{y} \in D_j \\ 0, & \mathbf{y} \notin D_j \end{cases} \tag{2.13b}$$

Another aggregation and disaggregation method is called soft aggregation. In soft aggregation, the probabilities are are assigned based on an interpolation of the probabilities of the aggregate states. Soft aggregation is illustrated in Fig. 2-6(b).

Bellman's equation for an aggregate state formulation can be seen in Eq. (2.14).

$$J^*(i) = g(i, \mathbf{u}) + \sum_{j=0}^{n} p_{ij}(\mathbf{u}) J^*(j) \tag{2.14}$$

To solve for the fixed point described in Eq. (2.14), there are three general methods to use:

- Value Iteration
- Policy Iteration
- Linear Programming

Policy iteration and linear programming are better suited problems with a small number of states, while value iteration is more suitable for problems with a large number of states. In anticipation of dealing with a large number of states, value iteration will be the solution method considered here. Let

$$(TJ)(i) = \min_{u \in U(i)} g(i, \mathbf{u}) + \sum_{j=1}^{n} p_{ij}(\mathbf{u}) J(j) \tag{2.15}$$

be the mapping of T. Value iteration looks to find the fixed point $J^*(i) = TJ^*(i)$ for all $i = 0, ..., n$ by iterating the mapping $TJ(i)$.

$$J^*(i) = \lim_{k \to \infty} (T^k J)(i) \tag{2.16}$$

For value iteration to be implemented, the term $p_{ij}(\mathbf{u})J(j)$ must be approximated. If points in D_i, $i = 0, ..., n$ are uniformly distributed and sampled q times, $p_{ij}(\mathbf{u})J(j)$ can be approximated by

$$p_{ij}(\mathbf{u})J(j) \approx \frac{1}{q} \sum_{s=1}^{q} J(j_s) \tag{2.17}$$

where

$$\mathbf{x}_s \in D_i \tag{2.18a}$$

$$\mathbf{f}(\mathbf{x}_s, \mathbf{u}) \in D_{j_s} \tag{2.18b}$$

The other other adjustment for aggregation methods is if $f(\mathbf{x}_k, \mathbf{u}_k) \notin C$ for all $\mathbf{u}_k \in U_k$. This happens when the system, by the nature of the dynamics, transitions out of the state space regardless of the control applied. In this case, the system can be penalized with cost τ. If a stochastic shortest path formulation is used, the system can also transition to the termination state, since the system would terminate anyway.

The algorithm used for solving the the aggregation formulation is seen in Algorithm 1.

Algorithm 1 Aggregation Value Iteration Algorithm

```
Given $J_0(i)$
for $k = 0, ..., N$ do                                        ▷ Run through $N$ iterations
    for $i = 0, ..., n$ do                                    ▷ Loop through all aggregation sets
        for $m = 0, ..., p$ do                                ▷ Loop through inputs in $U$
            $u_m \in U$
            for $s = 1, ..., q$ do                            ▷ Run through $q$ samples
                $x_s^k \in D_i$                               ▷ Sample a point in $D_i$
                $x_s^{k+1} = f(x_s^k, u_m)$                   ▷ Propagate the state equation
                $\hat{g}_k(s) = g(x_s^k, x_s^{k+1}, u_m)$     ▷ Compute transition cost
                if $x_s^{k+1} \notin C$ then                  ▷ Check for out-of-bounds
                    $\hat{J}_k(s) = \tau$                     ▷ Penalize out of bounds
                else
                    $\hat{J}_k(s) = J(j)$, where $x_s^{k+1} \in D_j$
                end if
            end for
            $\tilde{g}_k(m) = \frac{1}{p+1}\sum_{s=0}^{p} \hat{g}_k(s)$    ▷ Compute expected transition cost
            $\tilde{J}_k(m) = \frac{1}{p+1}\sum_{s=0}^{p} \hat{J}_k(s)$    ▷ Compute expected cost-to-go
        end for
        $J_{k+1}(i) = \min_{m\in(1,..,p)} g_k(m) + \tilde{J}_k(m)$     ▷ Evaluate Bellman's equation
    end for
end for
```

2.3.2 Cost Approximation

In cost approximation a basis function, combined with tuning parameters, is used to approximate the cost-to-go of the system. The cost approximation takes the general form of $\tilde{J}(\boldsymbol{\phi}(\mathbf{x}), \mathbf{r})$, where $\boldsymbol{\phi}(\mathbf{x})$ is a basis function that can be computed given state $\mathbf{x}$, and $\mathbf{r}$ is a tuning parameter for the approximation. In the linear case:

$$\tilde{J}(\mathbf{x}, \mathbf{r}) = \boldsymbol{\phi}(\mathbf{x})\mathbf{r} \tag{2.19}$$

where $\boldsymbol{\phi}(\mathbf{x})$ is a row vector and $\mathbf{r}$ is a column vector. The key concept of cost approximation is that an approximation for the cost-to-go can be found on-line using state information combined with a set of pre-computed tuning parameters $\mathbf{r}$. The true cost-to-go is mapped into the subspace described by$\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{r}$. There are several different methods to compute the tuning parameters for a linear approximation, including:

- Temporal Difference
- Least Squares Temporal Difference
- Least Squares Policy Evaluation

For this work the linear cost approximation, shown in Eq. (2.19), will be used. The $\mathbf{r}$ vector will be updated according to the least squares temporal differences (LSTD) method. This method involves simulating a long trajectory and building a matrix C and vector $\mathbf{d}$ such that

$$C\mathbf{r}^* = \mathbf{d} \tag{2.20}$$

The LTSD finds $\mathbf{r}^*$ iteratively using information from the trajectories. As a trajectory moves from i_k to i_{k+1}, C_k and $\mathbf{d}_k$ are updated via Eqs. (2.21) and (2.22)

$$C_k = (1-\delta_k)C_{k-1} + \delta_k \boldsymbol{\phi}(i_k)\left(\boldsymbol{\phi}(i_k) - \boldsymbol{\phi}(i_{k+1})\right)^\top \tag{2.21}$$

$$\mathbf{d}_k = (1-\delta_k)\mathbf{d}_{k-1} + \delta_k \boldsymbol{\phi}(i_k) g(i_k, i_{k+1}) \tag{2.22}$$

When a batch of simulations is complete, $\mathbf{r}_k$ is updated using Eq. (2.23).

$$\mathbf{r}_k = C_k^{-1}\mathbf{d}_k \tag{2.23}$$

Since the trajectories will eventually terminate, it is necessary to restart the trajectories. To ensure sufficient exploration, a random feasible state is used as the starting state of the system.

Using the LSTD algorithm described in Algorithm 2 the parameter vector $\mathbf{r}$ that best matches the cost to go can be found via simulation.

Algorithm 2 Cost Approximation Iteration Algorithm

```
Given $r_0$
$r_k = r_0$
for $k = 0, ..., n$ do                                        ▷ Loop through $n$ iterations
    $x^0 \in D_s$
    $C^{-1} = [0]$
    $d^{-1} = [0]$
    $i = 0$
    while $i < N$ do                                          ▷ Run through $N$ samples
        $u^i = \arg\min_{u \in U(i)} g\left(x^i, f(x^i, u^i)\right) + \tilde{J}\left(\phi\left(f\left(x^i, u\right)\right) r\right)$
        $x^{i+1} = f\left(x^i, u^i\right)$                    ▷ Propagate the state equation
        $\delta_k = \frac{1}{k+1}$
        $C_k^i = (1 - \delta_k)C_k^{i-1} + \delta_k \phi(x^i)\left(\phi(x^i) - \phi(x^{i+1})\right)^\top$
        $d_k^i = (1 - \delta_k)d_k^{i-1} + \delta_k \phi(x^i) g(x^i)$
        if $x^{i+1} \in T$ then                               ▷ Check for termination
            $x^{i+1} \in D_s$                                 ▷ Restart with a new point
        end if
        $i = i + 1$
    end while
    $r_{k+1} = (C_k^N)^{-1} d_k^N$
end for
```

2.4 Applying Dynamic Programming to a Physical System

While dynamic programming has been used to solve a multitude of complex problems, dynamic programming solutions have rarely been used as a control method for physical systems. The large amount of computation required combined with the large amount of storage space required have hindered dynamic programming's use on physical systems. With advances in computing and storage, applying dynamic programming to physical systems is becoming a reality. The flow of development for a dynamic programming controller is described in Fig. 2-7.

There are several major considerations when developing a dynamic programming controller. The first is that the physical system needs to be able to be formulated in a dynamic programming formulation with as few states as possible. This will most likely involve mapping the states of the real system into a subspace, performing dynamic programming on the subspace, and mapping the subspace back to the real system. A

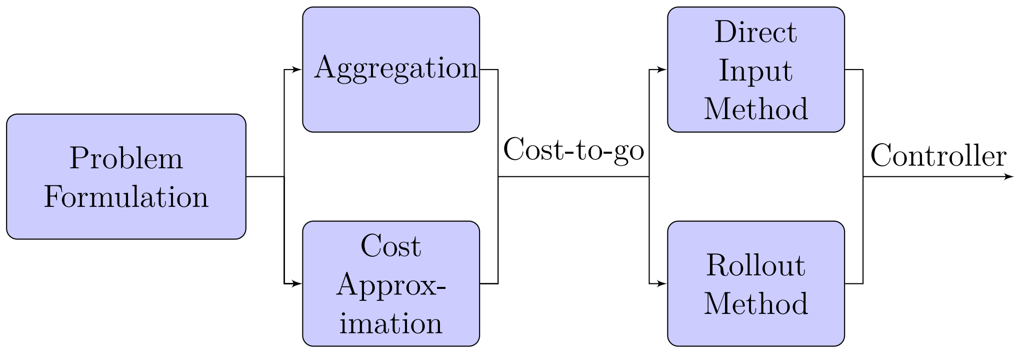


Figure 2-7: Controller Development Using Dynamic Programming

good understanding of the important versus the non-important states of the system will be required for the problem formulation. It is up to the control engineer to decide which states can be simplified and which must be accounted for.

The finding the cost-to-go for a physical system may also be difficult because the system lives in the real world where an infinite number of states are possible. While approximate dynamic programming maps the infinite number of states into a subspace, it is still up to the control engineer to decide which states are feasible and therefore be accounted for in the cost-to-go. For instance, certain states may be impossible to achieve because of obstructions or physical limitations. Also, there may be certain states that have associated trajectories that take the system out of the state space of the dynamic program. It is again up the control engineer on how to account for these cases. Chapter 4 will present one method for dealing with these problems, but there are multiple ways to address these issues.

More considerations for applying dynamic programming to a control problem are discussed in Section 5.1.

Chapter 3

Formulating RINGS as a Dynamic Programming Problem

While the following two chapters will present novel results for RINGS control, the real purpose of these two chapters are to be a case study for formulating and solving a control problem for a satellite system using dynamic programming. This chapter will discuss specific formulations of the RINGS system that are convenient for dynamic programming. Chapter 4 will develop solutions of the cost-to-go the specific formulation outlined in Section 3.2.1.

The most fundamental part of the RINGS formulation is the interaction between the two coils. The forces and torques on one electromagnet from the other can be found by integrating the Biot-Savart law, as seen in Eqs. (3.1) and (3.2) [41], where the terms in these equations are defined in Fig. 3-1.

$$\vec{F}_2 = \frac{\mu_0 i_1 i_2}{4\pi} \oint \left(\oint \frac{\hat{r} \times \mathrm{d}\vec{l}_1}{r^3} \right) \times \mathrm{d}\vec{l}_2 \tag{3.1}$$

$$\vec{\tau}_2 = \frac{\mu_0 i_1 i_2}{4\pi} \oint \vec{a}_2 \times \left[\left(\oint \frac{\hat{r} \times \mathrm{d}\vec{l}_1}{r^3} \right) \times \mathrm{d}\vec{l}_2 \right] \tag{3.2}$$

There are no closed form solutions of Eqs. (3.1) and (3.2), so numerical methods will have to be used to approximate Eqs. (3.1) and (3.2).

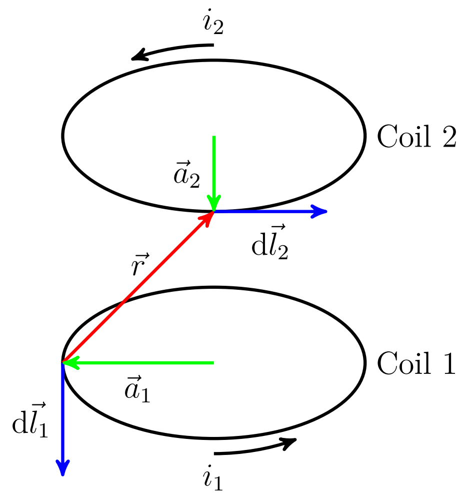


Figure 3-1: Definition of Two Coils in Proximity

3.1 RINGS Dynamic Programming Formulations

Unlike a simulation built for dynamic programming, which limits the states to those that the dynamic program is concerned about, an embedded system has a set number of real, physical states. To implement dynamic programming on an embedded system, the physical states must be mapped into states that are useful for dynamic programming. This section will map the physical states of RINGS into states that are useful for dynamic programming.

A satellite typically has thirteen standard states, which are grouped into four general categories:

- Position (3 states)
- Velocity (3 states)
- Attitude, measured in quaternions (4 states)
- Angular Rate (3 states)

For a two satellite system, this results in 26 states, which is far too many states for a dynamic programming formulation. However, symmetry in the problem can be used to reduce the number of states in the system.

3.1.1 Position State Reduction

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the position of satellite 1 and 2, respectively, in the global frame. If the RINGS units are the only actuators producing force, then all forces on the system are internal, meaning the center of mass of the system does not change. The location of the center of mass can be found using Eq. (3.3), while the position of the primary RINGS unit relative to the center of mass can be found using Eq. (3.4).

$$\mathbf{x}_c = 0.5\left(\mathbf{x}_1 - \mathbf{x}_2\right) + \mathbf{x}_2 \tag{3.3}$$

$$\mathbf{r} = \mathbf{x}_1 - \mathbf{x}_c \tag{3.4}$$

Let $\mathbf{x}_t$ be the target position of the primary RINGS unit in the global frame. The target position of the primary RINGS unit relative to the system center of mass can also be computed using Eq. (3.5).

$$\mathbf{r}_t = \mathbf{x}_t - \mathbf{x}_c \tag{3.5}$$

The objective of a controller is to drive $\mathbf{r}$ to $\mathbf{r}_t$, meaning the target state of the system is $\mathbf{r} = \mathbf{r}_t$ and $\dot{\mathbf{r}} = \dot{\mathbf{r}}_t$. This formulation takes advantage of the fact that the position of one satellite is symmetric about the origin of the system. This removes the three position states and three velocity states of the secondary satellite from the formulation because they can be inferred from the position of the primary satellite with respect to the center of mass. This symmetry is illustrated in Fig. 3-2.

The state space can be further reduced by mapping the motion of the satellites into a plane. Consider is plane defined by $\mathbf{x}_1$, $\mathbf{x}_c$ and $\mathbf{x}_t$. This plane allows the reduction of $\mathbf{r}$ and $\mathbf{r}_t$ into r, r_t, θ, and $\dot{\theta}$, where θ and $\dot{\theta}$ is the angle and angular rate, respectively, between r and r_t in the plane. This reduces the three position and three velocity states to a position and an angle with their associated rates. This reduces the number of states by two.

The distance of a RINGS unit from the center of mass r is found by taking the Euclidean norm, denoted by $\| \bullet \|$, of $\mathbf{r}$. Since the RINGS system is assumed to have

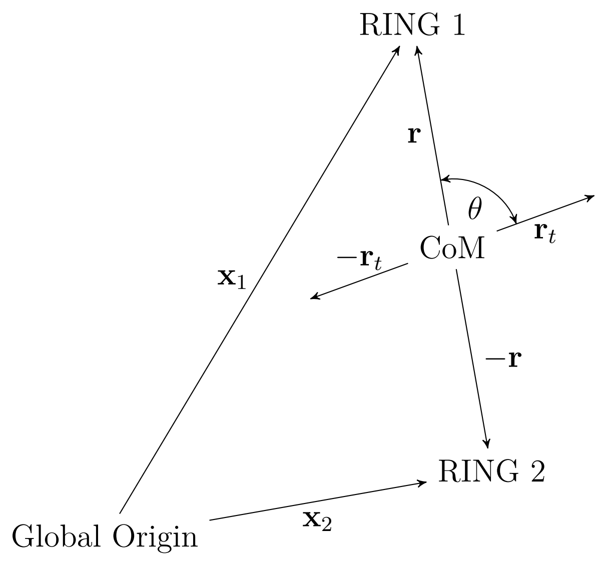


Figure 3-2: General RINGS State Illustration

no external forces, the center of mass does not change. This means the first derivative of r with respect to time becomes

$$\dot{r} = \frac{\dot{\mathbf{x}}_1^\top \mathbf{r}}{r} \tag{3.6}$$

The angle between $\mathbf{r}$ and $\mathbf{r}_t$ can be found using the dot product.

$$\theta = \cos^{-1}\left(\frac{\mathbf{r}^\top \mathbf{r}_t}{r \cdot r_t}\right) \tag{3.7}$$

The rate of change of θ is therefore seen in Eq. (3.8a). By canceling out an r_t from both the numerator and the denominator, Eq. (3.8a) can be simplified to Eq. (3.8b).

$$\dot{\theta} = -\frac{r \cdot r_t \left(\dot{\mathbf{r}}^\top \mathbf{r}_t\right) - \dot{r} \cdot r_t \left(\mathbf{r}^\top \mathbf{r}_t\right)}{\left(r \cdot r_t\right)^2 \sqrt{1 - \left(\frac{\mathbf{r}^\top \mathbf{r}_t}{r \cdot r_t}\right)}} \tag{3.8a}$$

$$\dot{\theta} = -\frac{r \left(\dot{\mathbf{r}}^\top \mathbf{r}_t\right) - \dot{r} \left(\mathbf{r}^\top \mathbf{r}_t\right)}{r^2 \cdot r_t \sqrt{1 - \left(\frac{\mathbf{r}^\top \mathbf{r}_t}{r \cdot r_t}\right)}} \tag{3.8b}$$

3.1.2 Attitude State Reduction

The attitude of each RINGS can be summarized by two angles. The in-plane angle ϕ describes the attitude of the coil in the plane. The out of plane angle ψ describes how far the coil is rotated out of the plane. Since the coil is assumed to be radially symmetric, any rotation about the axis of symmetry, called the coil normal vector, does not affect the dynamic of the system, so this rotation is ignored.

In order to compute ϕ and ψ, it is first necessary to define a relative coordinate system. This coordinate system, defined by $\begin{bmatrix} \hat{\mathbf{i}}_l & \hat{\mathbf{j}}_l & \hat{\mathbf{k}}_l \end{bmatrix}$ will be considered inertial but will be updated at each control update. The primary direction will be parallel to the position vector.

$$\hat{\mathbf{i}}_l = \frac{\mathbf{r}}{r} \tag{3.9}$$

The tertiary vector will be normal to the plane of motion, which is defined by the center of the system, the target position, and the current position.

$$\hat{\mathbf{k}}_l = \frac{\mathbf{r} \times \mathbf{r}_t}{\|\mathbf{r} \times \mathbf{r}_t\|}, \quad \mathbf{r} \neq \mathbf{r}_t \tag{3.10}$$

The secondary direction falls out from the primary and tertiary.

$$\hat{\mathbf{j}}_l = -\hat{\mathbf{i}} \times \hat{\mathbf{k}} \tag{3.11}$$

The rotation matrix from the global frame to the local frame becomes

$$T_{L/G} = \begin{bmatrix} \hat{\mathbf{i}} \cdot \hat{\mathbf{i}}_l & \hat{\mathbf{j}} \cdot \hat{\mathbf{i}}_l & \hat{\mathbf{k}} \cdot \hat{\mathbf{i}}_l \\ \hat{\mathbf{i}} \cdot \hat{\mathbf{j}}_l & \hat{\mathbf{j}} \cdot \hat{\mathbf{j}}_l & \hat{\mathbf{k}} \cdot \hat{\mathbf{j}}_l \\ \hat{\mathbf{i}} \cdot \hat{\mathbf{k}}_l & \hat{\mathbf{j}} \cdot \hat{\mathbf{k}}_l & \hat{\mathbf{k}} \cdot \hat{\mathbf{k}}_l \end{bmatrix} \tag{3.12}$$

Let $\hat{\mathbf{c}}$, the coil normal vector in the global reference frame, be the unit vector normal to the coil plane, as shown in Fig. 3-3.

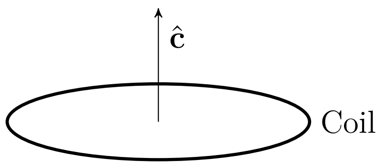


Figure 3-3: Definition of Coil Normal Vector

The coil normal vector in the local frame, $\hat{\mathbf{c}}_l$, can be found by using $T_{L/G}$

$$\hat{\mathbf{c}}_l = T_{L/G} \cdot \hat{\mathbf{c}} \tag{3.13}$$

The in-plane angle ϕ can then be computed by projecting $\hat{\mathbf{c}}_l$ into the plane of motion and computing the dot product between the radial direction and $\hat{\mathbf{c}}_l$. This is shown in Eqs. (3.14a) and (3.14b). Eq. (3.14a) is the dot product, while Eq. (3.14b) computes the magnitude of the projection of $\hat{\mathbf{c}}_l$ into the plane.

$$\phi = \cos^{-1}\left(\frac{\begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot \hat{\mathbf{c}}_l}{\sqrt{c_p}}\right), \quad \bar{c}_z \neq 1 \tag{3.14a}$$

$$c_p = \hat{\mathbf{c}}_l^\top \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \hat{\mathbf{c}}_l \tag{3.14b}$$

The out of plane angle ψ can be found using Eq. (3.15).

$$\psi = \cos^{-1}\left(\hat{\mathbf{k}}^\top \hat{\mathbf{c}}_l\right) \tag{3.15}$$

3.2 Specific Formulations

Using the general formulation described Section 3.1, it is possible to further simplify the problem by removing degrees of freedom. The following cases are specific formulations which have a reduced number of states.

3.2.1 Static Axial Case

In the static axial case, the RINGS are assumed to be fixed in attitude and can only vary position in one dimension. The attitude of both RINGS are fixed such that $\hat{\mathbf{c}}_l$ for each RINGS unit is parallel with the other's. Both RINGS are positioned such that the center of each coil lies on the line parallel to the coil normal vector. The only translation of the RINGS units occurs along this central line. An illustration of this setup is seen in Fig. 3-4.

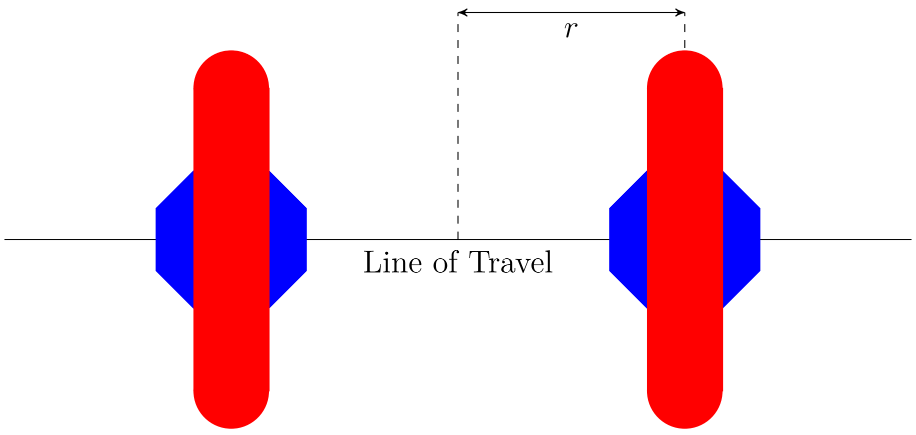


Figure 3-4: Static Axial RINGS Setup

Let r be the distance from the center of mass of the system to the center of a RINGS unit, $\dot{r}$ be the rate of change of the separation distance, and $F_a(r, u)$ be the axial force on a coil found by evaluating Eq. (3.1) with a separation of r and an input of $u = i_1 i_2$. The ordinary differential equation of motion governing this formulation is seen in Eq. (3.16).

$$\ddot{r} = -\frac{F_a(r, u)}{m} \tag{3.16}$$

In this formulation, $\phi = 0$ and $\psi = 0$. When implementing the static axial case, ϕ and ψ will have to be controlled by a separate controller.

3.2.2 Rotating Axial Case

The rotating axial case is the same as the static axial case described in Section 3.2.1, except for the line of travel is not fixed in inertial space. Instead, the line of travel rotates about the center of mass. Since the angular momentum of the system of conserved, the angular rate of the line of travel is affected by the separation of the RINGS. An illustration of the rotating axial case is seen in Fig. 3-5.

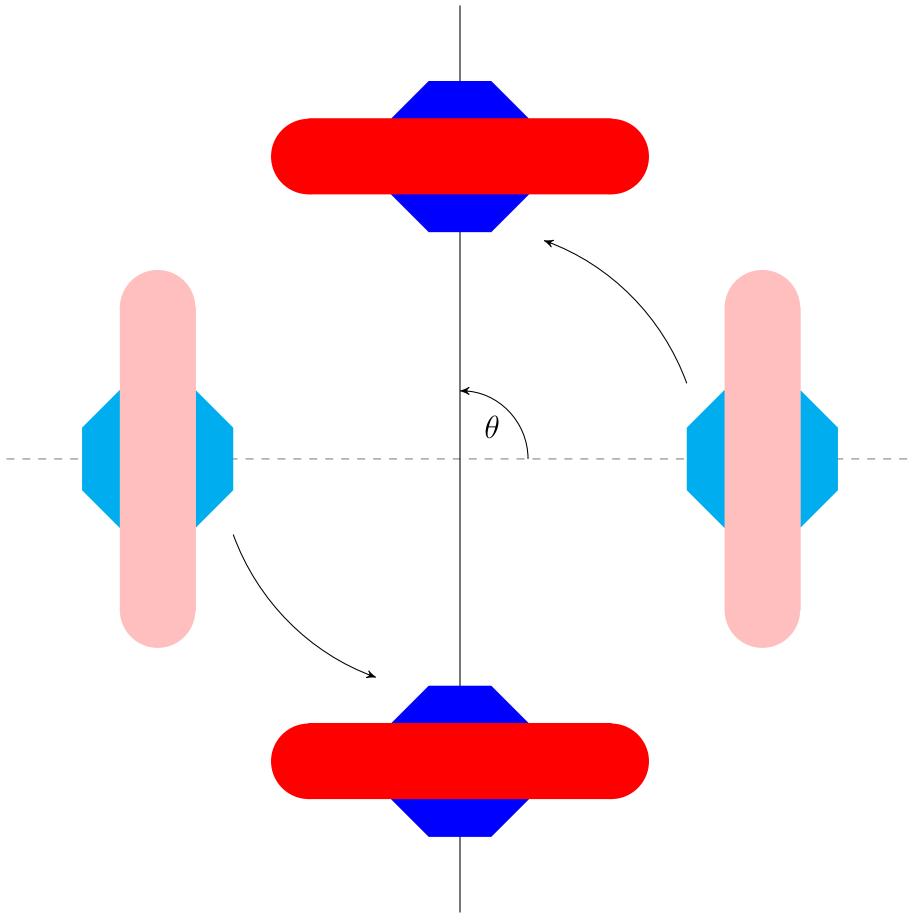


Figure 3-5: Rotating Axial RINGS Setup

The motion of this system is governed by the ordinary differential equation seen in Eq. (3.17).

$$\ddot{r} = \dot{\theta}^2 r - \frac{F_a(r,u)}{m} \tag{3.17}$$

Let L be the total angular momentum of the system and J be the inertia of the RINGS units normal to the plane of motion. The angular momentum of the system can be found using Eq. (3.18).

$$2J\dot{\theta} + 2mr^2\dot{\theta} = L \tag{3.18}$$

Since there are no external forces or torques on the system, the angular momentum of the system is conserved. Therefore, Eq. (3.18) can be manipulated in order to find the angular rate of the system as a function of separation distance.

$$\dot{\theta} = \frac{L}{2J + 2mr^2} \tag{3.19}$$

Like the static axial case, $\phi = 0$ and $\psi = 0$. When implementing the static axial case, ϕ and ψ will have to be controlled by a separate controller.

3.2.3 Planer Motion with Commanded Attitude

This case introduces two more states to the system as compared to Sections 3.2.1 and 3.2.2. Now, the RINGS are allowed to move freely in a two dimensional space. The attitude of the RINGS is assumed to be commanded. This can be accomplished with a separate attitude controller tracking commanded attitude. Section 5.2 discusses how this would be accomplished.

It is important to note that the position of one of the RINGS is simply the negative of the position of the other RINGS unit with respect to the origin. This is illustrated in Fig. 3-6. This is significant because only the position and velocity of one RINGS unit must be tracked to have full knowledge of the system, since the state of the other RINGS unit can be inferred from the first.

The equation of motion for this setup is seen in Eq. (3.20).

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \frac{F(\mathbf{r}, \mathbf{u})}{m} \tag{3.20}$$

The required control torque to overcome the torques caused by EMFF can be found for a given position using Eq. (3.2). This formulation still constrains ψ to 0

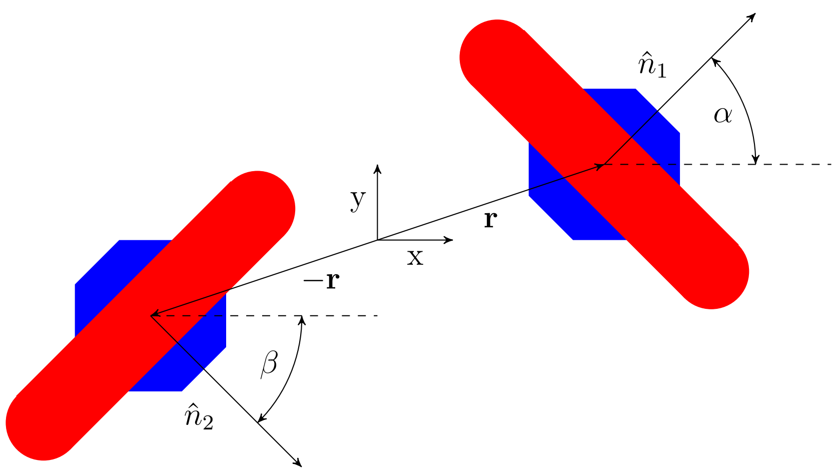


Figure 3-6: Full Planer Motion RINGS Setup

but lets ϕ be commanded instead of fixed.

3.2.4 Full Planer with Commanded Torque

This formulation as the same described in Section 3.2.3, but instead of commanding attitude, the system can only command torques. This adds four more states to the system: the two attitudes in the inertial frame α and β, as well as the associated rates of change. This formulation still requires an auxiliary controller for ψ but allows ϕ for both satellites to be controlled by the dynamic programming controller instead of an auxiliary controller.

Chapter 4

Cost-to-Go For EMFF Systems

Like Chapter 3, this chapter serves as a case study for applying dynamic programming to a physical system. Here results are presented for the static axial RINGS case, outlined in Section 3.2.1. This formulation is ideal for analysis because it only has two states, making visualizing the results more simple. The physical parameters for used are defined in Table 1.1.

4.1 Cost-to-Go Results

Using the methods described in Chapter 2, the cost to go for the RINGS axial static case were found using both aggregation and cost approximation. The results of both the aggregation method and cost approximation method are presented below for a stochastic shortest path formulation.

4.1.1 Aggregation Results

The sets of states D_i, $i = 0, ..., n$ where made by defining a set C of all potential states and then creating a uniformly distributed grid within C. The set C presented is the convex hull of the points $[r, \dot{r}] = [1.1 \pm 0.9, 0 \pm 0.4]$, where r is in meters and $\dot{r}$ in meters per second. The target box is the convex hull of $[r, \dot{r}] = [1 \pm 0.01, 0 \pm 0.01]$. The maximum time τ was set to 30 seconds. The results of this formulation is seen

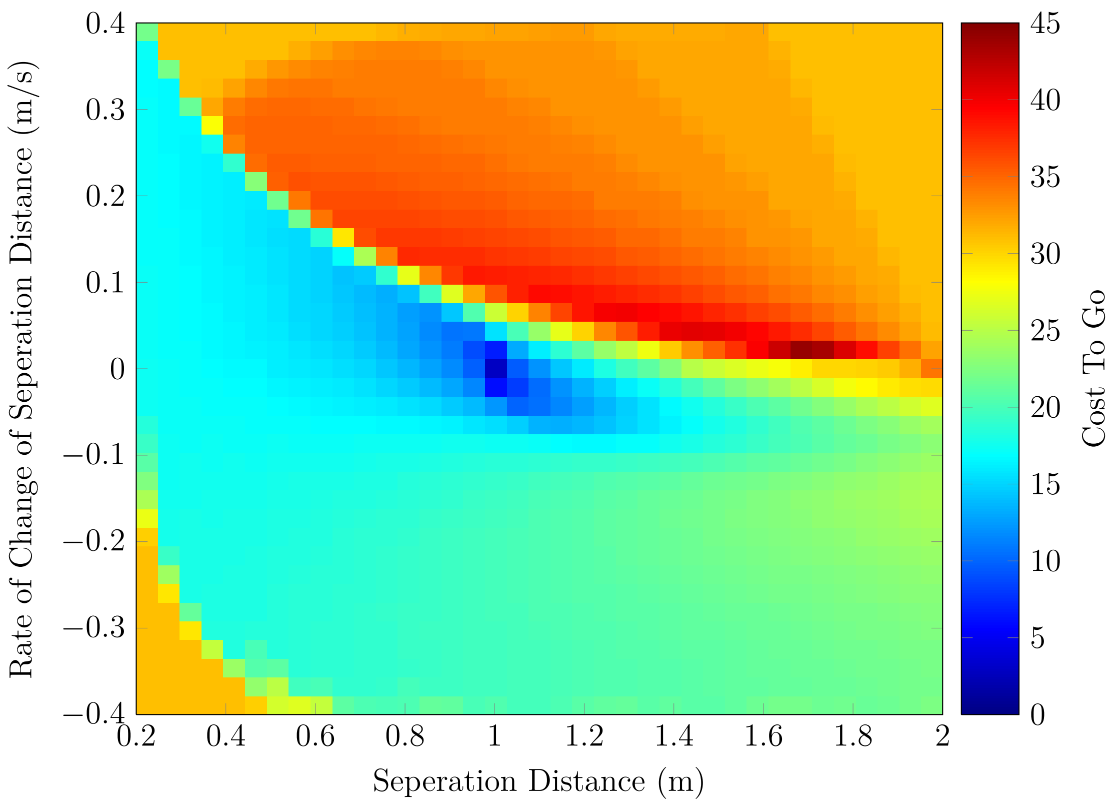


Figure 4-1: Cost To Go Using Aggregation

in Fig. 4-1.

The upper right region of high cost is the "terminal velocity" region of the state space, where the system has built up sufficient energy to reach escape velocity, hence why it never reaches the target. There are defined steps in this region because states that cannot reach the target position propagate to the "terminal velocity" states. The bottom left region of high cost is the region where no input will prevent the RINGS from crashing.

4.1.2 Cost Approximation Results

The starting separation of the RINGS was assumed to be uniformly distributed across the set $r = [0.2, 2]$. The RINGS are always assumed to be starting at rest, since this is how a real test would be conducted. The first basis function used was a two dimensional Taylor series; however, the cost approximation never converged on an optimal set of parameters with the Taylor series basis function. After several iterations, the matrix C_k would become singular, regardless of the number of terms in the Taylor series expansion. In addition to that problem, the Taylor series expansion allowed the cost function to have local minima not at the target set that would "trap" the simulation. One potential fix for this issue was only using even terms in the expansion; however, this made the rollout policy optimal to stay at the starting position for all starting positions.

After iterating over tens of basis functions, the following framework converged on an optimal set of tuning parameters. Let

$$h(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{4.1}$$

be a piecewise functions of x. Further, let $\| \bullet \|$ be the Euclidian norm operator and x_t be the target state. Using Eq. (4.1), the new basis function can be written as

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} h\left(x_r - x_r^t\right) & h\left(-\left(x_r - x_r^t\right)\right) & \|\mathbf{x} - \mathbf{x}^t\| \end{bmatrix} \tag{4.2}$$

where x_r corresponds to the state vector element relating to separation distance. The trajectories produced by this formulation can be seen in Fig. 4-2.

4.2 Analysis

Given the cost-to-go results presented in Section 4.1, the next step is the controller development process in to determine the best dynamic programming method for the system. This involves making a few design decisions in order to obtain the best cost-to-go for the system. The major decisions for the control engineer discussed here are:

- Whether to use Aggregation or Cost Approximation Cost-to-Go
- If using Aggregation, which type of interpolation to use

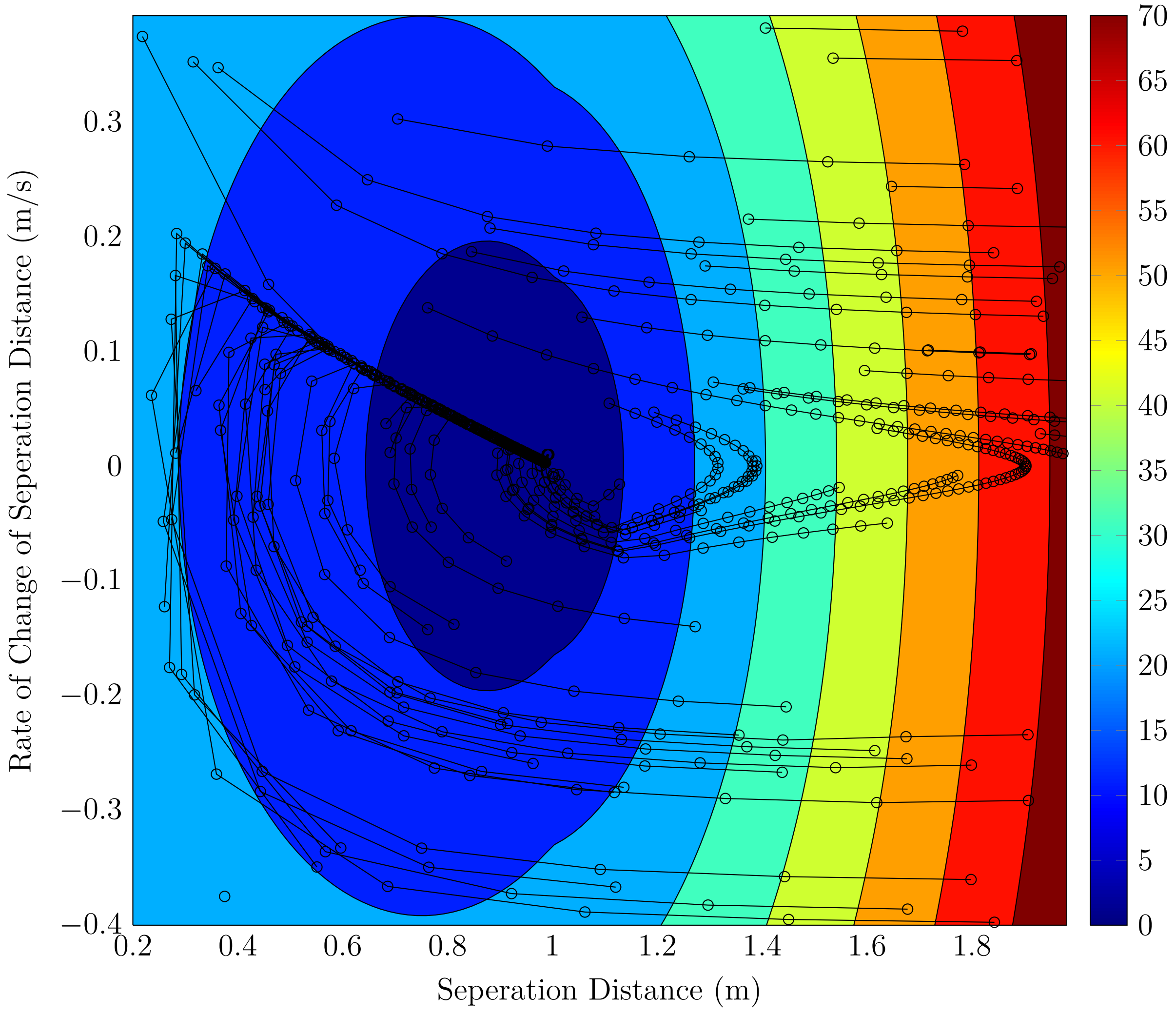


Figure 4-2: Cost To Go Using Cost Approximation With Sample Trajectories

- Whether to use Direct Input or Rollout Implementation

There are two trades that are not discussed in this section but nevertheless may be an important consideration for controller development. The first is the formulation type. A stochastic shortest path formulation was used here, but a discounted cost formulation could have also been used. If a discounted cost formulation was used, then the other trade would be the optimal cost function. While there are methods for tuning cost functions [61], [62], the choice of cost function is still up to the control engineer.

4.2.1 Aggregation Controller Performance

The results of the aggregation show that the optimal controller is essentially a "bang-bang" controller. The controller will use the maximum possible input, either positive or negative, until it reaches the switching curve, at which point the input maintains the same magnitude but switches to the opposite sign. The system then travels down the switching curve to the set of target states. This type of control agrees with the basic control theory that states a minimum time controller should take the form of a "bang-bang" controller.

In Fig. 4-1, the upper right corner of the plot appears to have distinct plateaus of constant cost. This entire region is the region that cannot reach the target state without leaving the state space. The region that leaves the state space in one control cycle regardless of input. The associates these states with the transition cost τ resulting from transitioning to the termination state. The stair effect results from other states in the terminal velocity region transitioning to the states other states that have found a path to the out-of-bounds termination state.

4.2.2 Aggregation Balance

Aggregation is a useful technique to reduce an infinite state space into a finite one. However, the level of aggregation is a balance between controller performance and storage space. To examine this trade, Algorithm 1 was run for different levels of

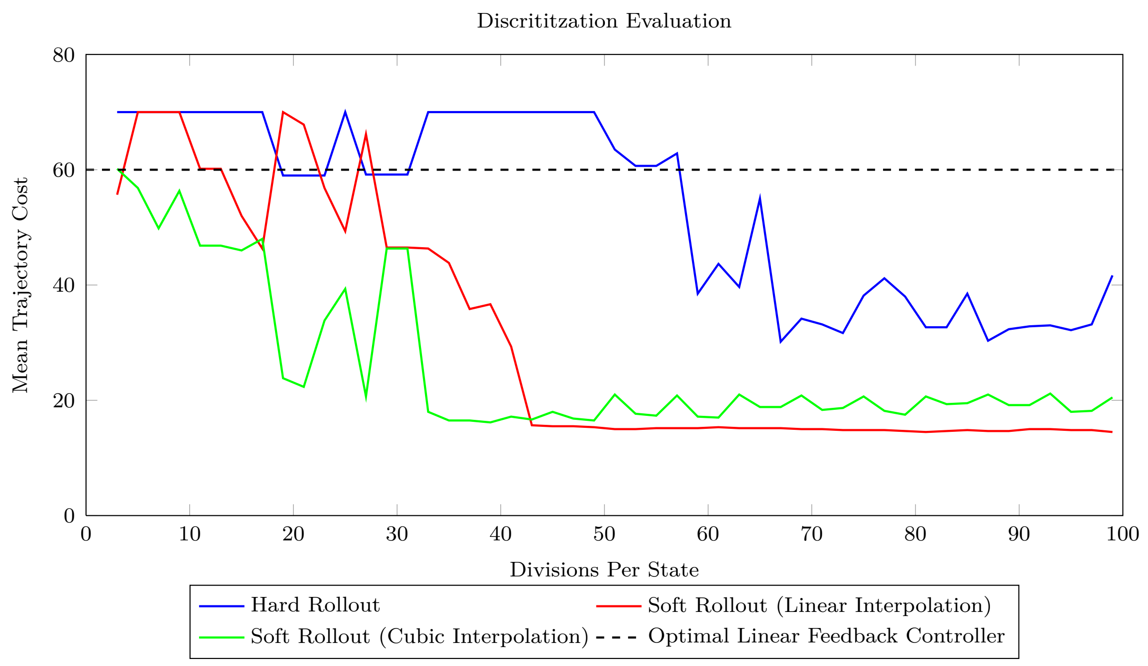


Figure 4-3: Aggregation Performance Over Differing Number of Divisions

aggregation. To compare between different levels of aggregation, the mean of the cost is used. The results are seen in Fig. 4-3. The average cost-to-go was calculated by taking the average of all of the values $J(i)$ over all i. The average rollout cost was calculated by simulating 500 trajectories that started at rest ($\dot{r} = 0$).

It is significant to note that the spikes in the data resulted from cost-to-go grids that had a local minimum that was not in the set of termination states. Certain states would fall into the local minimum and remain there until the simulation times out. A better cost metric may have been the integral sum of separation distance error.

Nevertheless, when the aggregation grid was coarse, the rollout policy seemed to perform at about the same level as the base policy, which was a hard aggregation. However, in this case, there is a point at about 50 divisions per state that yielded a significant improvement in the rollout policy. After this point, the rollout policy outperformed the base policy, which is often a positive property of rollout policies. It is also significant to notice that after a certain point, increasing the number of divisions per state yields little marginal benefit. This is significant because optimizing against this break even point can reduce the storage space required to store an aggregation policy.

Aggregation also suffers from the curse of dimensionality. All spacecraft, including SPHERES, are subject to controller size constraints. Typically these restrictions are very constrictive. While the axial case examined in this paper could be loaded and used to control RINGS, any higher dimensional cases could not be solved using aggregation. For the formulation described in Section 3.2.4, eight dimensions are required to describe the system. At ten division per dimension—which is a fairly coarse grid—at least 763 megabytes of storage space would be required. This is far to large for the SPHERES satellite. That would even stress the data uplink to the International Space Station itself.

4.2.3 Cost Approximation Performance

After a significant amount of work, Algorithm 2 was able to converge on an optimal tuning parameter vector using the basis function described by Eqs. (4.1) and (4.2). The issue of convergence most likely revolves around the choice of the basis function.

One potential fix to make the algorithm converge would be to find a basis function that still accurately describe the system without causing C_k to become singular. This will most likely involve trial and error, although basis function adaptation could also be used [63]. Ensuring convexity also helped in basis function selection. A convex (or at least quasi-convex [64]) basis function will help solve convergence issues. However, pre-describing this type of solution may remove potential basis function candidates. Also, as evidenced by the failure of the convex Taylor series approximation to converge, having a convex basis function does not guarantee convergence either.

4.2.4 Aggregation vs. Cost Approximation

Given the RINGS system starts at rest ($\dot{r} = 0$), the time to the target state is seen in Fig. 4-4. For any given initial position, the controller based on aggregation methods reached the target state in fewer time steps. Since aggregation methods store more information about the cost-to-go function, it is not surprising that the aggregation

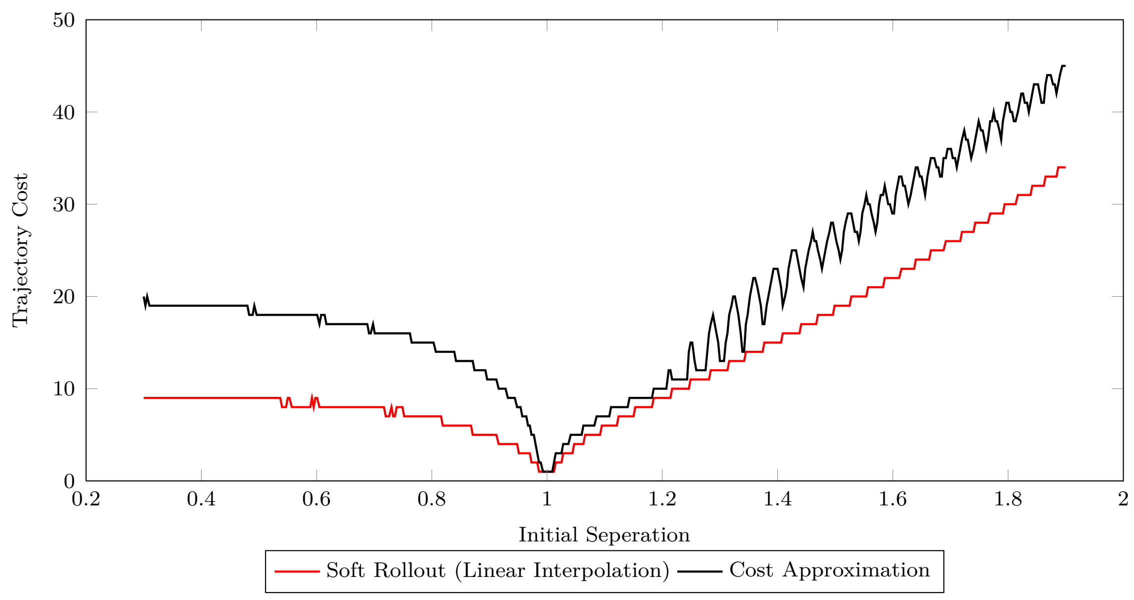


Figure 4-4: Aggregation Performance Over Differing Number of Divisions

method produced a lower time to target.

The major advantage of cost approximation over aggregation is that a cost approximation algorithm can be stored using significantly less storage space than an a controller developed using aggregation. Assuming the computational overhead of evaluating the basis function is low, an aggregation-based and a cost-approximation based controller would have the same computational overhead; however, the aggregation controller requires the lookup table be stored. As stated in Section 4.2.2, this overhead may be too much for a spacecraft to handle unless a coarse aggregation is used, which will affect the controller's performance.

Another advantage cost approximation has over aggregation is that simulation based methods can include states that the aggregation state might exclude. In this case, states with $\dot{r} < 0$ might have chosen to attract more, but that would have placed them out of the state space despite being able to rejoin the state space further down the trajectory.

The major advantage of aggregation over cost approximation is the ease in which the controller can be developed. Assuming the computer the controller is developed on has sufficient resources (processing power and memory), the aggregation controller development is a straightforward extension of finite state infinite horizon controllers. On the other hand, cost approximation requires intuition into the basis functions. No matter how much computing power is used, if the basis function do not match the problem well, then it will be impossible to find an optimal tuning parameter.

Chapter 5

Implementing A Dynamic Programming Controller

One of the impediments of using dynamic programming for controller development is implementing the controller on the system. Spacecraft are often limited in both storage space and computational ability, making implementing a dynamic programming controller difficult. Therefore, choosing the proper implementation technique is critical for the feasibility and success of the controller.

5.1 Control Design Considerations

In order do decide whether dynamic programming is an appropriate controller for a system, the general control engineering process must be considered. Using dynamic programming as a control method requires a paradigm shift in the spacecraft control field. Control engineers must consider the available computational and storage facilities on the ground for controller development, as well as the spacecraft capabilities. Such a foresight is typically not required when developing basic controllers.

In the development of any control, there is a general flow of development, which is described in Fig. 5-1. While there may be more considerations, including requirement verification and control validation, Fig. 5-1 describes the four major steps of control development. Steps 1,2, and 3 require design inputs from the control engi-

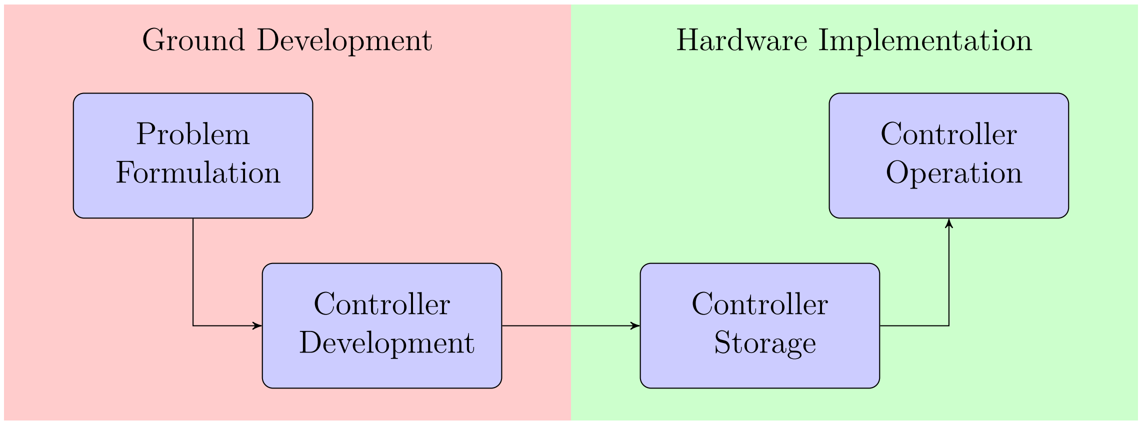


Figure 5-1: Controller Development Flow

Table 5.1: Controller Trade Space

Step	Decision	Limiting Factor
Problem Formulation	Model, States & Inputs	
Controller Development	Controller Type	Computing Resources
Controller Storage	Compression	Storage Capacity
Controller Operation		Computational Power

neer; however, steps 2,3, and 4 have constraints that may guide these decision. The decisions and limiting factors of the controller development process are summerized in Table 5.1.

5.1.1 Problem Formulation

Problem formulation involves taking a physical system and describing it with a system model. Typically this involves describing the system with states and inputs. The major trade during the problem formulation phase is the balance between accuracy and simplicity. A formulation that involves a complex model with many states may describe a system more accurately; however, controller development will be more difficult because of the complex model. On the other hand, a simple model will be easier to develop a controller for. The downside is that the controller based on the simple model may not perform well when implemented on the physical system. This is because the simplified system may not properly or fully describe the characteristics of the physical system.

Table 5.2: Problem Formulation Model Trade Space

More Simple	More Accurate
Analytic	Numerical
Linear	Non-Linear
Deterministic	Stochastic
Continuous Time	Discrete Time
Unbounded Control	Control Saturation

Typical trades during the problem formulation can be seen in Table 5.2. Table 5.2 is only meant to be an example, the trade space may differ based on the physical system.

5.1.2 Controller Development

Once a system model is properly formulated, a controller can be developed for the system based on the system model. The controller development step is typically where most of the time of the control engineer is spent. The key decision point in the controller development phase is what type of control will be used. The control engineer must trade between development time, intellectual difficulty of development, and available resources. Development time and intellectual difficulty of development will not be considered here, as they involve the engineer more than the controller itself. However, available resources—typically computational resources—may limit what type of control is used. For example, the development of linear controllers require little to no computational resources during development. Pseudo-spectral optimal control requires some computational resources. Dynamic programming, depending on the number of states, can require significant computational resources when developing the cost-to-go.

5.1.3 Controller Storage

Once a controller is developed, it must then be stored in the electronics[1] of the physical system. This typically involves storing or booting a system with the control software.

[1]Some controllers can be realized via a mechanical system instead of electronics. Those systems are not considered here.

The two considerations here are boot time and storage space. If the controller will be updated frequently, the time it takes to change the controller may be a consideration.

Where the controller is written is also an important consideration. Depending on the types of avionics used, the controller may have to be written to a section of memory that can only be written to a limited number of times. This reduces the ability to iterate controllers during operation.

The key decision point for the control engineer at this stage is the type and amount of compression used to store the controller. Data compression is a trade between the amount of storage space required to store the controller and the amount of processing required to extract the required information. This trade is largely driven by the amount of available memory for the controller and the processing ability of the avionics.

5.1.4 Controller Operation

Once the controller is stored, it is ready to be operated. At this point there are no more design decisions for the engineer. Instead, the performance of the system can be measured. The metrics of performance are not only the actual response from the system, but also the computational load of the controller. Although the controller may perform well, the controller may also strain the processor, drawing extra power and dissipating extra heat. If this is the case, then the controller may have to be redesigned.

5.1.5 Use of Dynamic Programming

Given the considerations listed in Sections 5.1.1 to 5.1.4, the advantages dynamic programming are:

- A complex model can be used (Section 5.1.1)
- The controller will be robust (Section 5.1.4)

The disadvantages of dynamic programming are:

- Controller development will require significant computational resources for system with more than a couple states (Section 5.1.2)
- The controller will require a significant amount of storage space (Section 5.1.3)

If the system needs the advantages of dynamic programming and is tolerant of the disadvantages, then dynamic programming may be an appropriate controller for the system.

5.2 Dynamic Programming Implementation

There are two general methods for implementing a dynamic programming controller:

- Storing the optimal input as a function of the state directly (Direct Input)
- Storing the cost-to-go and performing a rollout algorithm (Rollout)

The storage requirements of a direct input controller scale linearly with the number of control inputs, while the storage requirement of a rollout controller is independent of the number of inputs. However, a direct input controller requires little online computation, whereas the rollout controller may require significant online computation.

The general method for applying a dynamic programming controller to a physical system, which applies to both direct input and rollout controllers, is discussed in Section 5.2.1. The specifics of the direct input controller are described in Section 5.2.2, while the specifics of the rollout controller are described in Section 5.2.3.

5.2.1 General Architecture

The objective of a dynamic programming controller, and typically any controller, is to take an estimated state and a target state and return the relevant input(s). This architecture is described by Fig. 5-2.

The controller can be further broken down into the dynamic programming controller and other controllers. This separation may be needed because the state vector returned by the estimator may not be the same state vector used by the dynamic

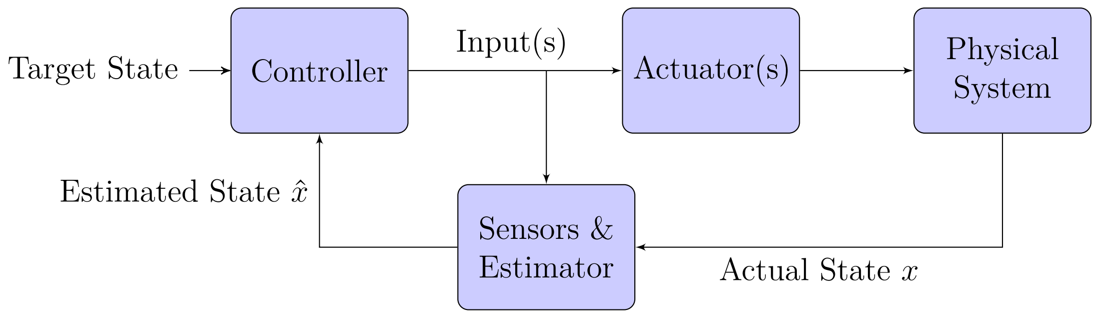


Figure 5-2: General Control Architecture

programming controller. In this case, the state vector returned from the estimator must be mapped to the states that the dynamic programming controller is expecting. Additionally, the formulation of the dynamic program may make assumption about certain states of the physical system. To ensure these assumptions are valid, other controllers may be necessary to regulate certain states of the system while the dynamic programming controller acts on other states of the system. This state breakout is illustrated in Fig. 5-3.

The first step of the controller is to take the estimated state $\hat{x}$ and map those states into the states used in the dynamic programming controller and the auxiliary controller. The dynamic programming states and auxiliary states are not necessarily mutually exclusive.

At this point, the dynamic programming controller and auxiliary controller can run simultaneously. The dynamic programming controller will take the dynamic programming target state and and the dynamic programming state to determine the appropriate control. The dynamic programming target state may have been known *a priori*, or determined on-line using the same mapping as the estimated state. At the same time, the auxiliary controller can determine the appropriate control based on the auxiliary states. Any control method can be used for the auxiliary controller, so long as the physical system is able to meet the storage and processing demands of both the dynamic programming controller and auxiliary controller.

Once both the dynamic programming controller and the auxiliary controller have determined the appropriate control, the controls must reconciled before they can

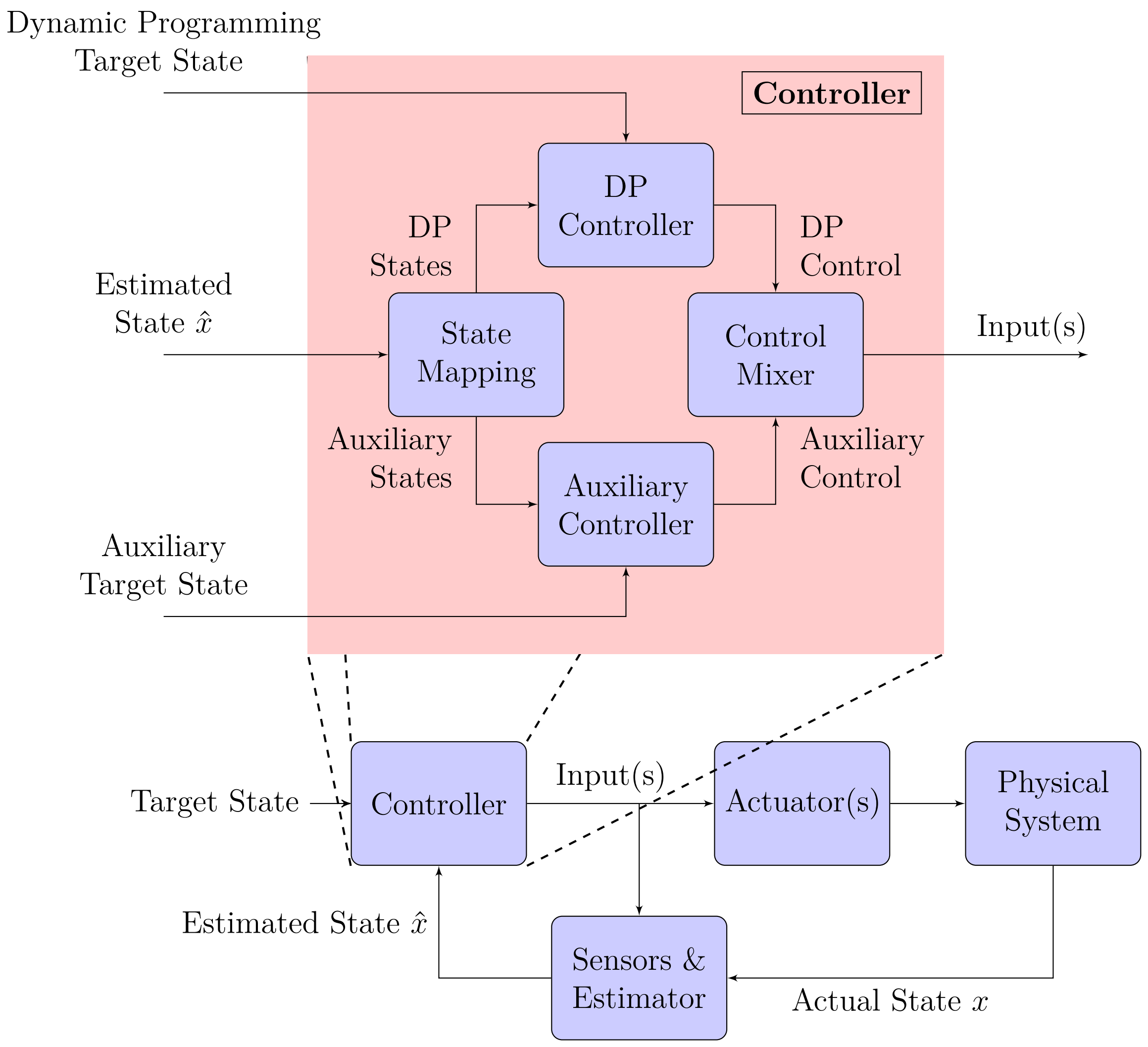


Figure 5-3: State Breakout Architecture

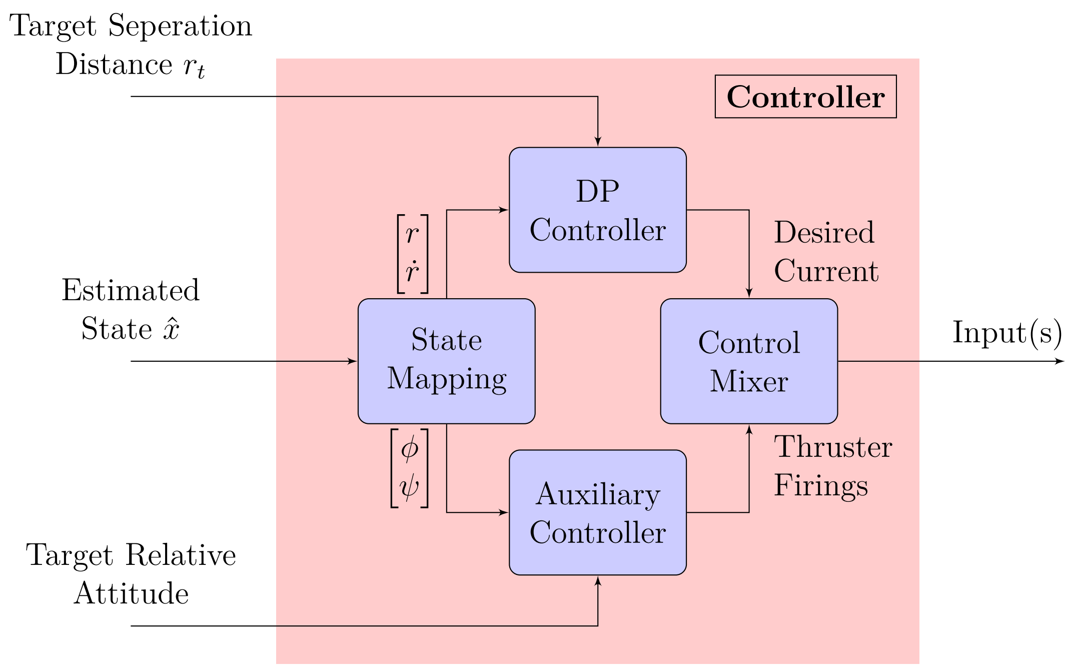

Target Seperation Distance r_t
Controller
DP Controller
$\begin{bmatrix} r \\ \dot{r} \end{bmatrix}$
Desired Current
Estimated State $\hat{x}$
State Mapping
Control Mixer
Input(s)
$\begin{bmatrix} \phi \\ \psi \end{bmatrix}$
Thruster Firings
Auxiliary Controller
Target Relative Attitude

Figure 5-4: RINGS Static Axial Breakout Architecture

be applied. If the controls are mutually exclusive, then the mixing process is very straightforward. However, if the dynamic programming controller returns a control that conflicts with a control returned by the auxiliary controller, the control has to be reconciled. This may involve choosing one over the other, choosing a convex combination of the two controls, or some other method. Once the control is reconciled, it can be passed out of the controller as an input to the actuators.

An example of Fig. 5-3 applied to the RINGS static axial case, described in Section 3.2.1, is seen in Fig. 5-4. In this example, the objective is to drive the RINGS to a target separation distance r_t using EMFF while maintaining the RINGS in an axial configuration using thrusters.

The SPHERES estimator returns a thirteen element state vector for each satellite containing the (estimated) position, velocity, attitude in quaternions, and angular rate. The state mapping would be accomplished via the method described in Section 3.1. Once the separation distance and the separation distance rate of change (r and $\dot{r}$, respectively) are known, the dynamic programming controller can then begin computing the desired current. When the relative attitude error is known, the auxiliary controller, can determine the appropriate thruster firings to regulate the attitude.

5.2.2 Direct Input Controller

The most intuitive way to implement a dynamic programming controller is to store the optimal input as a function of the state. The on-line controller would simply have to perform a lookup at every control update. To develop such a lookup table, states would have to be sampled and the input associated with that state stored.

The basic implementation in this form requires storing every input at every sampled state. During a control update, the controller finds the proper input based on the current state of the system and the stored input table. Finding the proper input could involve linear or cubic interpolation, or could be as simple as a nearest-point lookup. A block diagram of this type of architecture is seen in Fig. 5-5. Notice that the lookup table is a function of the target state(s). This means that a different lookup table may be required for different target states.

5.2.3 Rollout Controller

The other option is to implement a rollout controller. A rollout controller solves Bellman's equation at the current state using the stored cost-to-go as a heuristic. The advantage of this approach over a direct input method is two fold. First, rollout algorithms outperform the base heuristic used in the rollout [60]. Second, a rollout controller requires only storing the cost-to-go for every sampled state, while a direct input controller requires storing every input at every sample state. The amount of storage required can increase significantly ff the system has multiple inputs. The downside of rollout controllers is that they require more on-line computation than direct input controllers. A rollout controller has to compute the minimum of Bellman's equation, meaning it must predict the state at the next control cycle for a set of inputs. This requires propagating the system for every set of inputs, which can be computationally expensive. A block diagram of a rollout controller architecture is

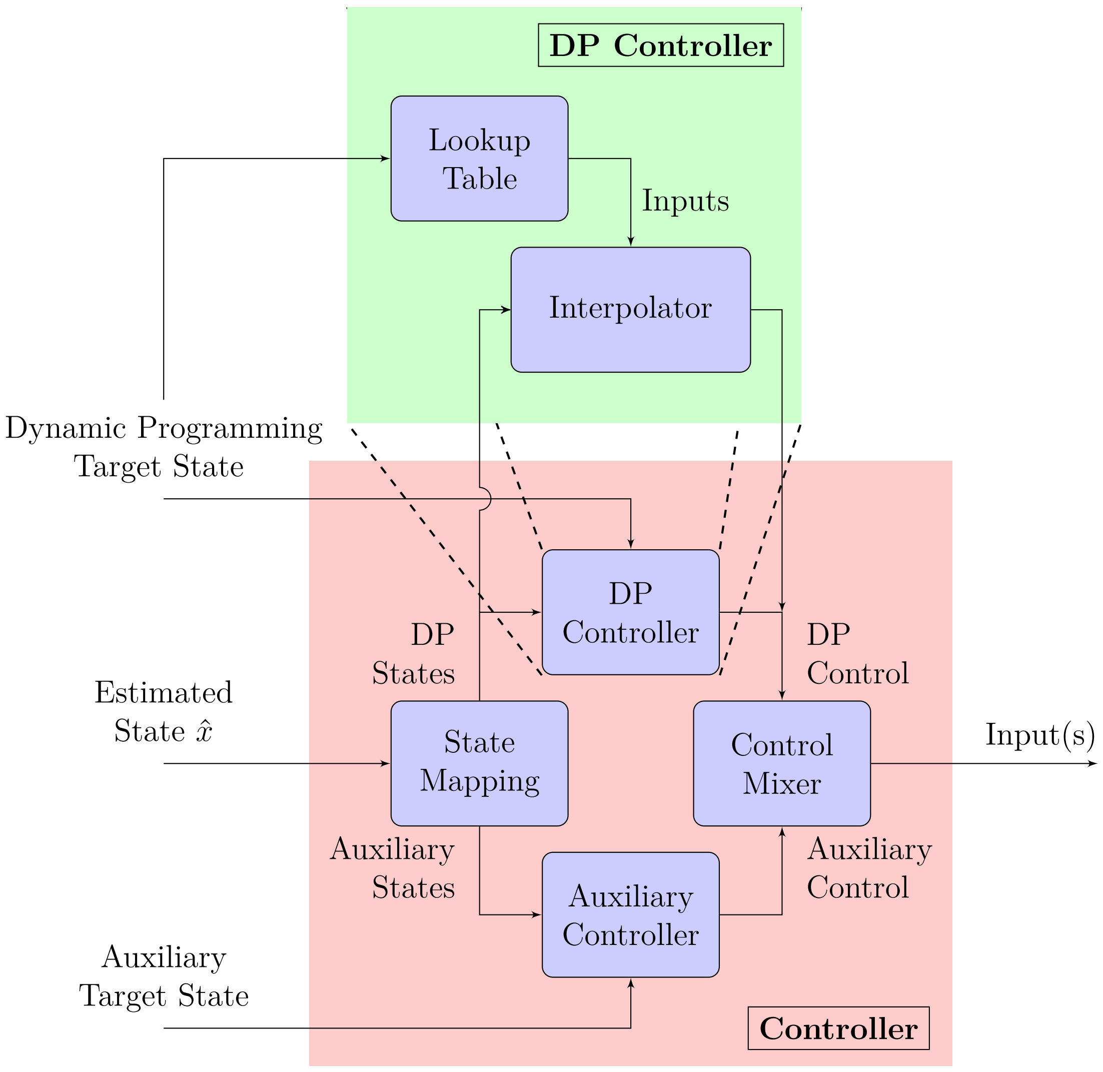


Figure 5-5: Direct Input Controller Architecture

seen in Fig. 5-6.

The propagation step of the rollout controller takes the estimated state of the system and predicts the state of the system after input u is applied. This is done for all potential inputs. If the state equation has a closed form solution, this step may be very simple. If the state must be numerically propagated, this step may be computationally intensive. In either case,

$$\hat{\mathbf{x}}_{k+1}(\mathbf{u}) = f(\hat{\mathbf{x}}_k, \mathbf{u}) \tag{5.1}$$

must be computed for all $\mathbf{u} \in U$. Once $\hat{\mathbf{x}}_{k+1}(\mathbf{u})$ is known, the cost-to-go for each input must be computed. The cost-to-go in the cost table is typically stored as a function of the state. To find cost-to-go as a function of input, the cost-to-go must be evaluated at the potential states stored in $\hat{\mathbf{x}}_{k+1}(\mathbf{u})$. This process occurs at the interpolator step.

Once the cost-to-go as a function of input is known, all that is left is to solve Bellman's equation. The optimal control is found by solving

$$u^* = \arg\min_{u \in U} g(\hat{x}_k, \hat{x}_{k+1}(u), u) + J(\hat{x}_{k+1}(u)) \tag{5.2}$$

where $g(\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{u})$ is the transition cost from state $\mathbf{x}_k$ to state $\mathbf{x}_{k+1}$ using control $\mathbf{u}$.

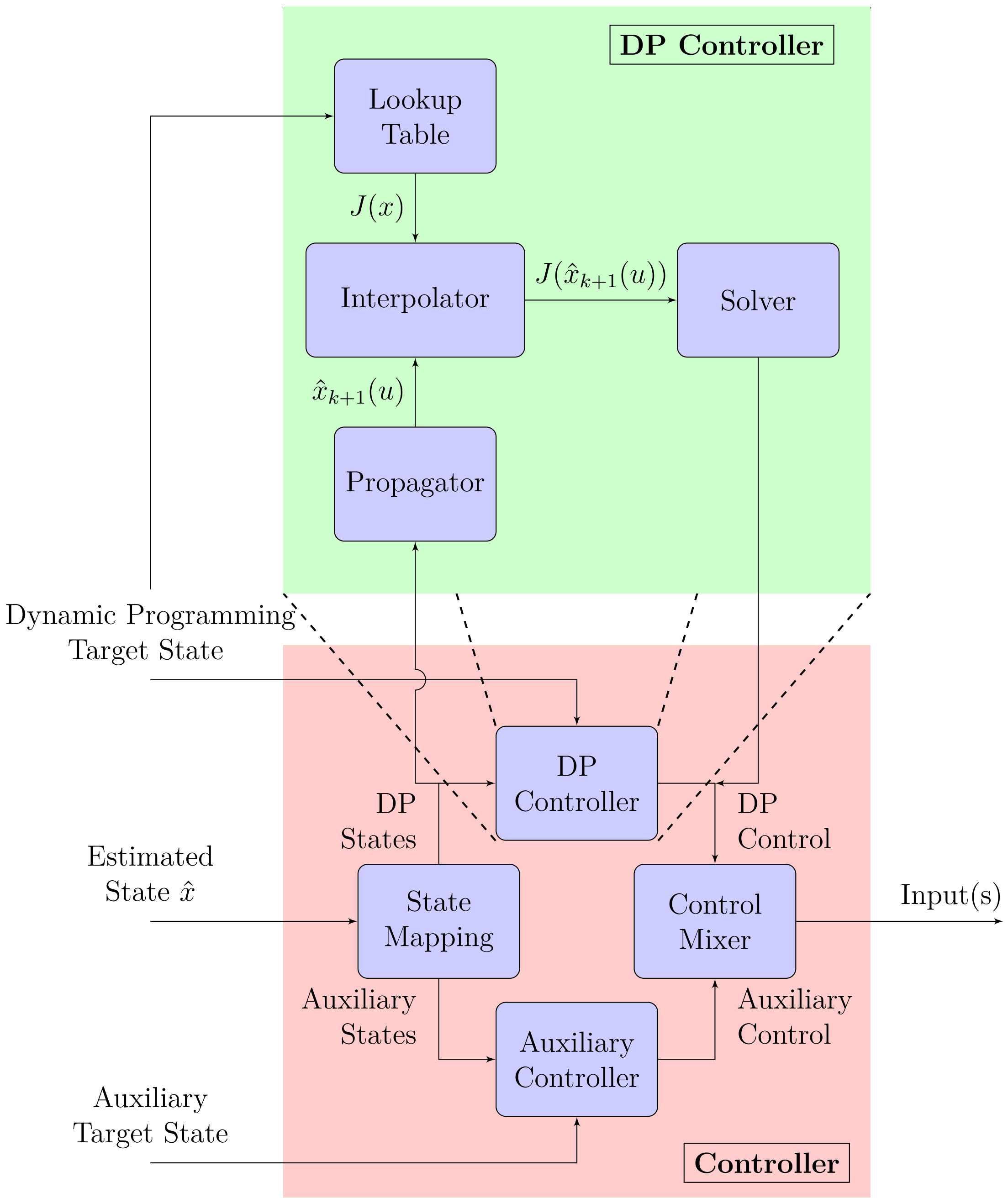


Figure 5-6: Rollout Controller Architecture

Chapter 6

Nonlinear Programming Mass Property Identification for Spacecraft

The performance of dynamic programming algorithms, like many other control methods, depends on the fidelity of the model of the system. Good estimates for the inertia and center of mass of a spacecraft are required for precision attitude and position control. When there is an error between the modeled and actual mass properties, extra propellant or energy is expelled as the controller attempts to fix its own errors. This also causes a degradation in attitude knowledge, attitude pointing ability, and the ability to maintain tight position requirements.

This chapter will describe a new method for identifying mass properties in a microgravity environment using nonlinear programming. Section 6.1 provides a summary of established on-orbit mass property identification. Section 6.2 formulates a mass property identification problem as a nonlinear program. Section 6.3 discusses how to solve for the mass properties given the problem formulation. Section 6.4 establishes the convergence guarantees for this method. Section 6.5 discusses the considerations when implementing these algorithms on a physical system.

6.1 Known Methods for Mass Identification

The previous methods for mass property identification can be grouped into two general categories: least-squares methods and filtering methods.

6.1.1 Least Squares Methods

Least squares mass identification algorithms attempt to find the mass properties contained in the vector $\mathbf{x}$ by manipulating Euler's equation of motion in order to write it in the form

$$A\mathbf{x} = \mathbf{b} \tag{6.1}$$

where A is an m by n matrix and $\mathbf{b}$ is a n by 1 vector. The mass properties can then be solved for by multiplying both sides of Eq. (6.1) by the left psuedo-inverse of A.

$$\mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{b} \tag{6.2}$$

Tanygin and Williams derived a least squares mass property identification algorithm for coasting maneuvers that was nonlinear in dynamics but linear in mass properties [65]. Wilson, Lages, and Mah developed a recursive least squares method for mass property estimation [66].

These methods require knowledge of the angular acceleration of the system. This typically requires numerical differentiation of the angular rate, which is measured using rate gyros. Process and sample noise on the rate gyros often lead to poor estimates of the angular acceleration.

6.1.2 Filtering

It is also possible to consider the mass identification problem as a hidden Markov model problem. In a hidden Markov model problem, the objective is to find $x(t)$ given $y(0), y(1), ..., y(t)$. A description of a hidden Markov process is seen in Fig. 6-1.

Filters are often used to solve hidden Markov models. Filters use the measurement at the current time step $y(t)$ as well as a sufficient statistic encapsulating

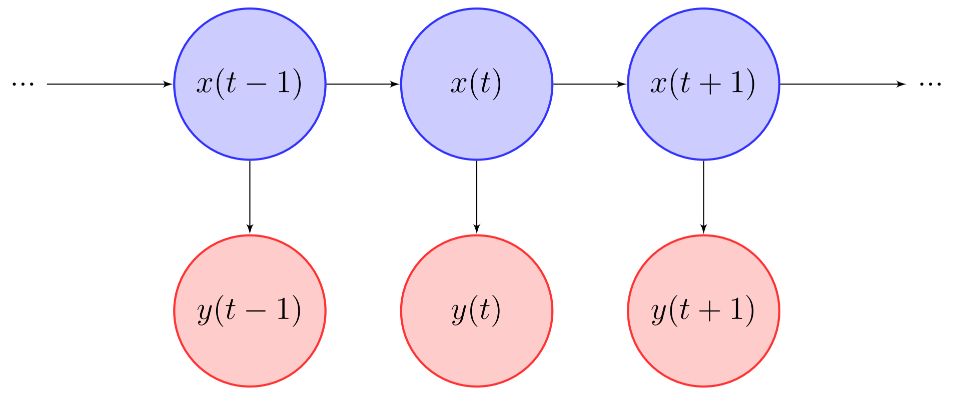


Figure 6-1: Hidden Markov Model

$y(0), y(1), ..., y(t-1)$. Filters that can be used to solve the hidden Markov problem include:

- Kalman Filter
- Extended Kalman Filter
- Unscented Kalman Filter
- Particle Filter

A main advantage of filters is that they can typically be implemented on-line. This is particularly useful if the mass properties change in a relatively short period of time where it is not feasible to compute the mass properties off-line.

Bergmann, Walker, and Levy developed a second order filter to determine the inertia of a system and Kalman filter to determine the center of gravity [67]–[69]. According to Bergmann, a good estimate for the angular acceleration is required for the filter to perform well. As discussed in Section 6.1.1, a good estimate of the angular acceleration may be difficult to compute because it requires numerical differentiation of noisy data.

6.2 Problem Formulation

Nonlinear programming has not previously been applied to the mass identification problem. A nonlinear program uses search algorithms to find the minimum of a cost function iteratively by using the gradient and, depending on the search algorithm, the Hessian of the cost function. Nonlinear search algorithms include:

- Steepest Descent Method
- Newton Method
- BFGS Method
- First-Order Methods

The overall methodology of a nonlinear programming-based mass identification algorithm can be summarized in a three step process that is run iteratively until convergence:

1. Simulate trajectories using a guess of the mass properties
2. Compare the simulated trajectory against a measured trajectory
3. Updated the mass properties guess based on the comparison of the simulated and actual trajectories using one of the methods described above.

The objective of the mass identification algorithm is to find the inertia tensor J and center of gravity in the body frame $\mathbf{c}$ of a system, where J and $\mathbf{c}$ are describe in Eqs. (6.3) and (6.4), respectively.

$$J = \begin{bmatrix} j_{xx} & j_{xy} & j_{xz} \\ j_{xy} & j_{yy} & j_{yz} \\ j_{xz} & j_{yz} & j_{zz} \end{bmatrix} \tag{6.3}$$

$$\mathbf{c} = \begin{bmatrix} c_x & c_y & c_z \end{bmatrix}^\top \tag{6.4}$$

The vector of unknown parameters $\mathbf{x}$ can then be written as

$$\mathbf{x} = \begin{bmatrix} j_{xx} & j_{yy} & j_{zz} & j_{xy} & j_{xz} & j_{yz} & c_x & c_y & c_z \end{bmatrix}^\top \tag{6.5}$$

The motion of all rigid bodies are governed by Euler's Equations of Motion, seen in Eq. (6.6).

$$J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (J\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{6.6}$$

For a sufficiently small time step, Eq. (6.6) can be written as

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \Delta t \cdot J^{-1}\left(\boldsymbol{\tau}_k - \boldsymbol{\omega}_k \times (J\boldsymbol{\omega}_k)\right) \tag{6.7}$$

where $\boldsymbol{\omega}_k$ and $\boldsymbol{\tau}_k$ is the angular rate and applied torque at time k, respectively. In anticipation of simulating angular rate trajectories based on the values of $\mathbf{x}$, Eq. (6.7) can be written in terms of $\mathbf{x}$.

$$\boldsymbol{\omega}_{k+1}(\mathbf{x}) = \boldsymbol{\omega}_k(\mathbf{x}) + \Delta t \cdot N(\mathbf{x})\left[\boldsymbol{\tau}_k(\mathbf{x}) - (\boldsymbol{\omega}_k(\mathbf{x})) \times [J(\boldsymbol{\omega}_k(\mathbf{x}))]\right] \tag{6.8}$$

where $N(\mathbf{x}) = (J(\mathbf{x}))^{-1}$.

The assumptions used in the mass identification algorithm are summarized in Table 6.1.

Table 6.1: Mass Identification Assumptions

#	Assumption
1	All torques on the system are known
2	The spacecraft is a rigid body, meaning the spacecraft dynamics can be described by Eq. (6.6). This implies the mass properties are invariant as a function of time ($\dot{J} = [0]$, $\dot{\mathbf{c}} = 0$)
3	The time between angular rate measurements is sufficiently small such that Eq. (6.7) holds

The actual dynamics of the spacecraft are assumed to be governed by the following two equations.

$$\bar{\boldsymbol{\omega}}_{k+1} = \bar{\boldsymbol{\omega}}_k + \Delta t \cdot J^{-1}\left(\boldsymbol{\tau}_k - \bar{\boldsymbol{\omega}}_k \times (J\bar{\boldsymbol{\omega}}_k)\right) + \boldsymbol{\sigma}_k \tag{6.9}$$

$$\tilde{\boldsymbol{\omega}} = \bar{\boldsymbol{\omega}} + \boldsymbol{\delta}_k \tag{6.10}$$

where $\bar{\boldsymbol{\omega}}$ and $\tilde{\boldsymbol{\omega}}$ are the true and measured values of angular rate, respectively, and $\boldsymbol{\sigma}_k$ and $\boldsymbol{\delta}_k$ are random variables associated with process noise and sample noise, respectively, at time k. However, Assumptions 1 and 2 of Table 6.1 imply that $\boldsymbol{\sigma}_k = 0$ for all k. The implications of this assumption are discussed in Section 6.5.

The error between a simulated trajectory and an actual trajectory can be measured using Eq. (6.11).

$$e_i(\mathbf{x}) = \sum_{k=i}^{i+n} \left(\boldsymbol{\omega}_k(\mathbf{x}) - \tilde{\boldsymbol{\omega}}_k + \mathbb{E}[\boldsymbol{\delta}_k]\right)^\top \left(\boldsymbol{\omega}_k(\mathbf{x}) - \tilde{\boldsymbol{\omega}}_k + \mathbb{E}[\boldsymbol{\delta}_k]\right) \tag{6.11}$$

where $\boldsymbol{\omega}_i(\mathbf{x}) = \tilde{\boldsymbol{\omega}}_i$.

The cost function described in Eq. (6.12) is the summation of many trajectory simulation.

$$h(\mathbf{x}) = \sum_{i=0}^{N} e_i(\mathbf{x}) \tag{6.12}$$

6.3 Solving the Program

Once the objective function is defined, it can then be solved using one of the different nonlinear programming solvers. Most nonlinear programming solvers require knowledge of the gradient. Since the objective function is computationally intensive, computing the Hessian of the objective function was considered infeasible. Therefore, gradient-only solvers are considered here.

6.3.1 Computing the Gradient

Let ω_k^i be the ith element of $\boldsymbol{\omega}_k$. The gradient of h can be written as

$$\nabla h = 2\sum_{i=0}^{N}\sum_{k=i}^{i+n} \frac{\partial \boldsymbol{\omega}_k}{\partial \mathbf{x}} \cdot \left(\boldsymbol{\omega}_k(\mathbf{x}) - \tilde{\boldsymbol{\omega}}_k + \mathbb{E}[\boldsymbol{\delta}]\right) \tag{6.13}$$

where

$$\frac{\partial \boldsymbol{\omega}_k}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \omega_k^1}{\partial j_{xx}} & \frac{\partial \omega_k^2}{\partial j_{xx}} & \frac{\partial \omega_k^3}{\partial j_{xx}} \\ \frac{\partial \omega_k^1}{\partial j_{yy}} & \frac{\partial \omega_k^2}{\partial j_{yy}} & \frac{\partial \omega_k^3}{\partial j_{yy}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \omega_k^1}{\partial c_z} & \frac{\partial \omega_k^2}{\partial c_z} & \frac{\partial \omega_k^3}{\partial c_z} \end{bmatrix} \tag{6.14}$$

Eq. (6.14) can be computed using Eq. (6.15).

$$\frac{\partial \boldsymbol{\omega}_k}{\partial \mathbf{x}} = \frac{\partial \boldsymbol{\omega}_{k-1}}{\partial \mathbf{x}} + \Delta t \frac{\partial}{\partial \mathbf{x}} \left(N(\mathbf{x}) \cdot \tau_k(\mathbf{x}) \right) - \Delta t \frac{\partial}{\partial \mathbf{x}} \left(N(\mathbf{x}) \left[\boldsymbol{\omega}_{k-1}(\mathbf{x}) \times \left(J(\mathbf{x}) \omega_{k-1}(\mathbf{x}) \right) \right] \right) \tag{6.15}$$

In order to solve Eq. (6.15), let

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\omega}_{k-1} \end{bmatrix} \tag{6.16}$$

The first term, $\partial \boldsymbol{\omega}_{k-1} / \partial \mathbf{x}$, is propagated from the previous time step. The second term, $\frac{\partial}{\partial \mathbf{x}} \left(N(\mathbf{x}) \cdot \tau_k(\mathbf{x}) \right)$, is straightforward to compute. The last term in Eq. (6.15), which is more difficult to compute, can be written as $\mathbf{f}(\mathbf{z})$, as seen in Eq. (6.17).

$$\mathbf{f}(\mathbf{z}) = N(\mathbf{z}) \left(\boldsymbol{\omega}_{k-1} \times \left[J(\mathbf{z}) \boldsymbol{\omega}_{k-1} \right] \right) \tag{6.17}$$

Solving for $\partial \mathbf{f} / \partial \mathbf{x}$ can be accomplished in the following manner

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \tag{6.18}$$

where

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} I & 0 \\ \frac{\partial \boldsymbol{\omega}_{k-1}}{\partial \mathbf{x}} & I \end{bmatrix} \tag{6.19}$$

Finite difference methods are another way of approximating ∇h, since implementing and evaluating Eqs. (6.13) and (6.15) is not trivial.

6.3.2 Gradient-Only Solvers

Since the Hessian of the objective function is not easily accessible, the available optimization algorithms are:

- Steepest Descent
- Quasi-Newton Methods
- First Order Methods

These methods are outlined below.

Steepest Descent Methods

The Steepest Descent Algorithm, as presented by Prof Robert Freund of MIT, is seen in Algorithm 3.

Algorithm 3 Steepest Descent

```
Given $x^0$
$k \leftarrow 0$
**repeat**
    $d^k := -\nabla f(x^k)$
    $\alpha = \arg\min_\alpha f(x^k + \alpha d^k)$
    $x^{k+1} \leftarrow x^k + \alpha d^k$
    $k \leftarrow k + 1$
**until** $d^k = 0$
```

The steepest descent method has the advantage of being one of the simplest optimization methods. It is also requires little overhead, as the only information that is saved from an iteration is the current value of x^k. However, the steepest descent method does use a line search, which can be costly depending on the function. The steepest descent method can also break down when the problem is ill-conditioned, that is, when the search path falls into a valley that is steep in one direction but shallow in another.

Quasi-Newton Methods

Quasi-Newton Methods attempt to build up an estimate of the Hessian as the search progresses by using information on the cost function and the gradient. The earliest Quasi-Newton Method is the Davidon, Fletcher, Powell, or DFP, method [70]. The most popular method nowadays is the Broyden, Fletcher, Goldfarb, Shanno, or BFGS, method [71]. The BFGS Algorithm, as presented by Prof Steven Hall of MIT, is seen in Algorithm 4.

Algorithm 4 Quasi-Newton (BFGS)

```
Given $x^0$
$B^0 := I$
$k \leftarrow 0$
**repeat**
    $d^k := -B_k \nabla f(x^k)$
    $\alpha = \arg\min_\alpha f(x^k + \alpha d^k)$
    $x^{k+1} \leftarrow x^k + \alpha d^k$
    $s_k = \alpha d_k = x^{k+1} - x^k$
    $y_k = \nabla f(x^{k+1}) - f(x^k)$
    $B^{k+1} \leftarrow \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right)^T B^k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}$
    $k \leftarrow k + 1$
**until** $d^k = 0$
```

The advantage of Quasi-Newton methods is that they do not struggle with ill-conditioned problems as much as steepest descent methods do. The downside is that additional calculations are required at each iteration. In addition, additional information, in the form of B_{k+1}, must be stored at every iteration.

First Order Methods

The Simple Gradient Scheme of the First Order Method, as presented by Prof Robert Freund of MIT, is seen in Algorithm 5.

This method is very similar to the steepest descent method in that the descent direction is simply the negative of the gradient of the function evaluated at x^k. However, one of the major assumptions of the first order method is that $f(\bullet)$ has a *Lipschitz*

Algorithm 5 First Order

```
Given $x^0$, $L$
$k \leftarrow 0$
**repeat**
    $d^k := -\nabla f(x^k)$
    $x^{k+1} \leftarrow x^k - \frac{1}{L}\nabla f(x^k)$
    $k \leftarrow k+1$
**until** $d^k = 0$
```

gradient. That is, there is a scalar L for which

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y-x\| \ \forall \ x, y \in \mathbb{R}^n \tag{6.20}$$

Unlike the steepest descent algorithm, which performs a line search at every step, the first-order method takes a fixed step governed by L. This reduces the amount of computation at each iteration, since L provides the approximate step distance the algorithm should take, whereas a steepest descent method has to search for the optimal step distance.

Since L is unknown for this application, L will be initialized at $L = 1$. At every iteration, if Eq. (6.21) holds, then the current value of L is sufficent for the search algorithm to converge.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2 \tag{6.21}$$

Should the current iteration fail this test, the algorithm is set to double the current guess of L and the step is recomputed. The algorithm that includes this change is seen in Algorithm 6.

6.4 Convergence Guarantees

A major advantage of the nonlinear programming method for mass property identification is the existence of convergence guarantees. In order to prove that the nonlinear program converges on the true mass properties, there are two major points that need

Algorithm 6 First Order (Modified)

```
Given $x^0$
$L = L^0$                                  ▷ The user must provide a guess for $L^0$
$k \leftarrow 0$
**repeat**
    $d^k := -\nabla f(x^k)$
    $x^{k+1} \leftarrow x^k - \frac{1}{L}\nabla f(x^k)$
    **if** $f(x^{k+1}) > f(x^k) + \nabla f(x^k)^T(x^{k+1} - x^{k+1}) + \frac{L}{2}\|y - x\|^2$ **then**
        $L = 2L$
        $x^{k+1} \leftarrow x^k$           ▷ This reruns the iteration with the new value for $L$
    **end if**
    $k \leftarrow k + 1$
**until** $d^k = 0$
```

to be proven:

- The error function $h(x)$ is convex
- The global minimum of $h(x)$ is achieved only when x is the vector of correct mass parameters

The former item ensures the search algorithm reaches the global minimum of $h(x)$, while the latter item ensures the global minimum is reached by the true mass properties and only the true mass properties. These two points are proven in Section 6.4.1 and Section 6.4.2, respectively.

6.4.1 Convexity of $h(x)$

Convexity of an objective function like $h(x)$ is significant because there are convergence guarantees for nonlinear programming solvers if the objective function is convex. Let H be the Hessian of $h(x)$. The objective function is convex if and only if H is positive semi-definite. A matrix is positive semi-definite if and only if the eigenvalues of the matrix are all non-negative. A matrix is also positive semi-definite if

$$x^\top Hx \geq 0 \tag{6.22}$$

for all $x \in \mathbb{R}^n$

At this point, $h(\mathbf{x})$ has not been proven convex. This is an ongoing area of research on the topic.

6.4.2 Convergence on Actual Mass Parameters

Theorem 6.4.1. *Let x^* be the vector of true mass parameters of the system. Then $\hat{x} = \arg\min h(x)$ if $\hat{x} = x^*$*

Proof. Suppose x^* is the vector of true mass parameters. If the sample trajectory is initiated at the same angular rate as the actual system (see Section 6.5), then $\omega_k(x) = \bar{\omega}_k$ for all k. This means Eq. (6.11) will reduce to

$$e_i(x^*) = \sum_{k=i}^{i+n} \left(\mathbb{E}[\delta] - \delta_k\right)^\top \left(\mathbb{E}[\delta] - \delta_k\right) \tag{6.23}$$

which is significant because

$$\frac{\partial}{\partial x} \left(\sum_{k=i}^{i+n} \left(\mathbb{E}[\delta] - \delta_k\right)^\top \left(\mathbb{E}[\delta] - \delta_k\right) \right) = \mathbf{0} \tag{6.24}$$

Therefore,

$$\nabla h = \sum_{i=0}^{N} \mathbf{0} \tag{6.25}$$

and x^* achieves the minimum of $h(x)$. □

It is significant to note that Theorem 6.4.1 did **not** read *if and only if* $\hat{\mathbf{x}} = \mathbf{x}^*$. This is because it is possible to achieve the global minimum of $h(\mathbf{x})$ with mass parameters that are not the true mass parameters. This occurs when the maneuvers used to characterize the mass properties do not provide sufficient information to characterize all mass properties. When this happens, a particular mass parameter can vary without changing the cost function, making that parameter unobservable.

The most obvious example of this occurring is when there are no torques imparted on the system. While the relation of the inertia values with respect to each other can be identified, the entire inertia matrix will be off by a scale factor. If there exist other possible solutions, the can be found by using Eq. (6.26), searching over ϵ.

$$\nabla h(\mathbf{x} + \boldsymbol{\epsilon}) = 0 \tag{6.26}$$

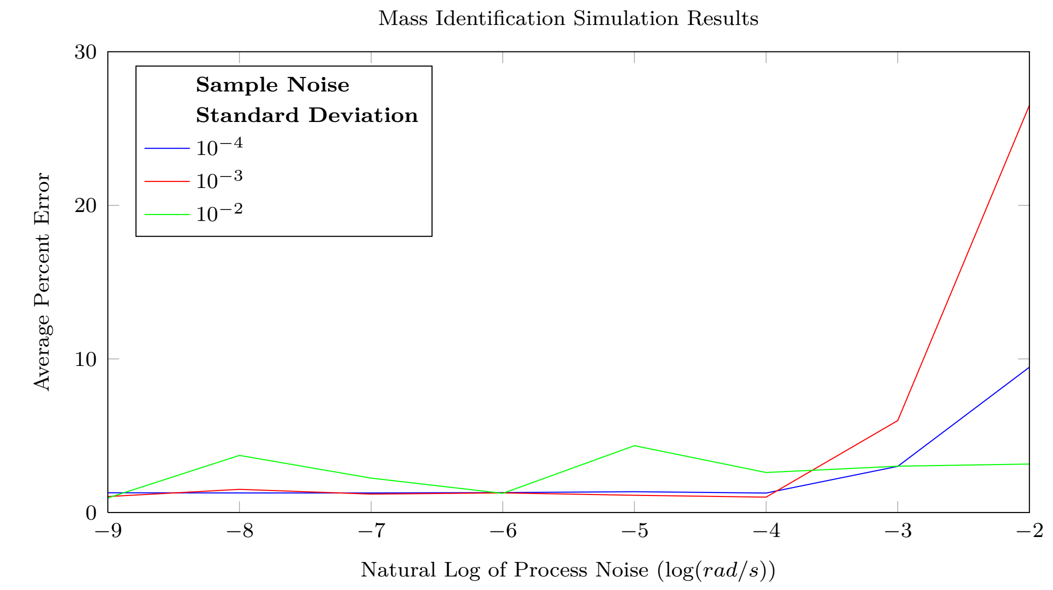
6.5 Implementation Considerations

There are two major assumptions that may not hold when this algorithm is implemented:

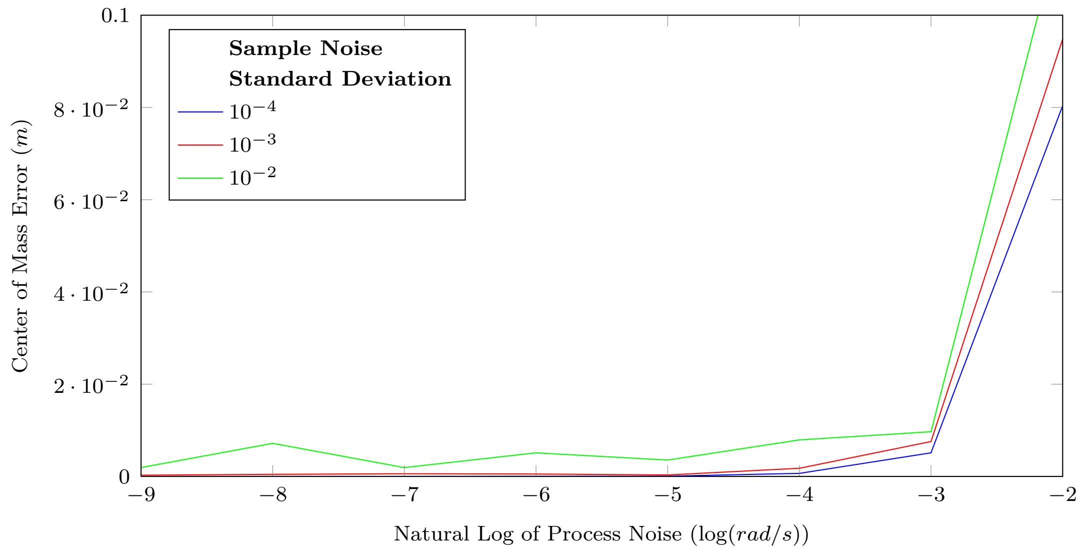
- A trajectory is started at the true angular velocity ($\boldsymbol{\omega}_i = \bar{\boldsymbol{\omega}}$)
- There is zero process noise ($\boldsymbol{\sigma}_k = 0$ for all k)

While the rest of the trajectory is tolerant of noise on the angular rate measurements, if the trajectory is started off of the true angular rate, the assumptions used to derive Theorem 6.4.1 do not hold. The best way of ensuring a trajectory is initiated at the true angular velocity is to filter the angular rate data.

To analyze the affect of effect of process and sample noise on the performance of the mass identification algorithm, a simulation was constructed that simulated the SPHERES mass property identification maneuvers, discussed in Section 7.1. Differing amounts of sample and process noise were injected into the system before the mass property identification algorithm was applied. The results are seen in Fig. 6-2. In Fig. 6-2(a), the calculated major moment of inertia was compared to the actual major moment of inertia. The plot shows the average present error as a function of process noise for three different sample noise characteristics. When the process noise is low, there is a small error, most likely caused by the violation of Assumption 3 in Table 6.1. As the process noise increases, so does the average error. In Fig. 6-2(b), a similar analysis was conducted for the center of mass prediction, with the same result: increasing process noise increases the error.



(a) Major Moment of Inertia Error.



(b) Center of Mass Error.

Figure 6-2: Mass Property Identification Simulation

Chapter 7

System Characterization for Thruster-Based Spacecraft

In a 1-G environment, it is difficult to fully characterize a spacecraft's mass and thruster properties. A spacecraft may not be able to support itself in a 1-G environment. The satellite may also change its configuration in space in such a manner that is difficult to replicate on the ground. Finally, the disturbance forces on the ground, such as airflow and friction, are much greater than the disturbance forces seen on orbit. For these reasons, a system characterization on orbit may greatly help the control algorithms of a spacecraft.

The SPHERES system is an ideal testbed for practicing on-orbit system characterizations. Video streams are available during a SPHERES test session, allowing for a visual feedback on the characterization. It is also a risk-tolerant way to practice system characterizations. If the characterization does not perform as expected, the satellite is at no risk of breaking or becoming lost in space. Therefore, the SPHERES testbed is an excellent place to mature system characterization tests.

The two main system characterizations considered here are thruster force characterization and mass property identification. With the addition of the SPHERES expansion port in January of 2012, the thruster force and mass properties of the system will change as new sensors and actuators are added to the system. These additions provide opportunities to demonstrate system characterization for thruster-

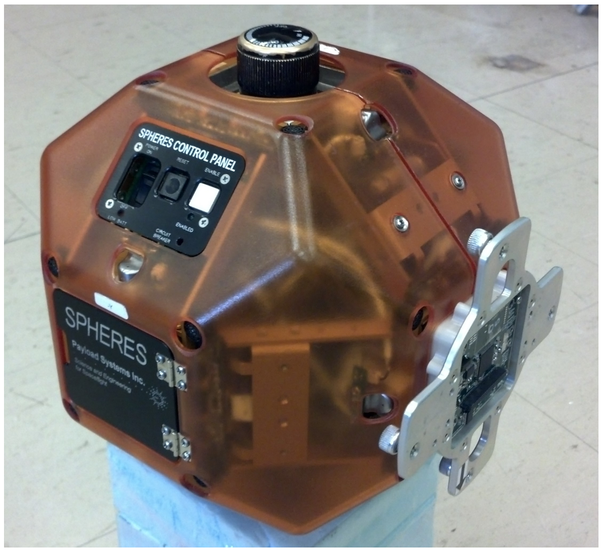


Figure 7-1: SPHERES With Expansion Port

based spacecraft.

7.1 SPHERES With Expansion Port

In January of 2012, an expansion port was added to the SPHERES testbed. This expansion port allows for additional actuators and sensors to be attached to the SPHERES both mechanically and electrically. A picture of the SPHERES with the expansion port is seen in Fig. 7-1.

While the new expansion port did not obstruct any of the thrusters, it did change the mass properties of the system. Therefore, a mass property characterization needed to be performed.

7.1.1 Predicted Changes

The mass properties of the SPHERES system without the expansion port had been found a number of different ways. A Computer Aided Design (CAD) model can predict the mass of a system through finite element methods. This method is highly dependent on the fidelity of the CAD model. If the mass of the modeled parts are not correct the CAD model predictions for the mass properties can be significantly

Table 7.1: SPHERES CAD Mass Property Predictions

Values	Full Tank	Empty Tank	Units
j_{xx}	2.30×10^{-2}	2.19×10^{-2}	$\mathrm{kg\,m^2}$
j_{yy}	2.42×10^{-2}	2.31×10^{-2}	$\mathrm{kg\,m^2}$
j_{zz}	2.14×10^{-2}	2.13×10^{-2}	$\mathrm{kg\,m^2}$
j_{xy}	9.90×10^{-5}	9.90×10^{-5}	$\mathrm{kg\,m^2}$
j_{xz}	-2.95×10^{-4}	-2.95×10^{-4}	$\mathrm{kg\,m^2}$
j_{yz}	-2.54×10^{-5}	-2.54×10^{-5}	$\mathrm{kg\,m^2}$
c_x	0.48	0.49	mm
c_y	−1.19	−1.24	mm
c_z	1.08	3.98	mm

Reference Frame: Geometric Center

Table 7.2: SPHERES Parallelogram Mass Property Results

Values	Full Tank	Empty Tank	Units
j_{xx}	2.57×10^{-2}	2.45×10^{-2}	$\mathrm{kg\,m^2}$
j_{yy}	2.25×10^{-2}	2.15×10^{-2}	$\mathrm{kg\,m^2}$
j_{zz}	2.03×10^{-2}	2.03×10^{-2}	$\mathrm{kg\,m^2}$

Reference Frame: Geometric Center

off. An easy way to verify the fidelity of the model is to compare the predicted mass of the CAD model against the actual mass of the object.

Another method for inertia estimation is using a swinging parallelogram test. In this test, the frequency of a swinging parallelogram is used to find the inertia tensor [72]. Finally, mass property identification methods described in Chapter 6 can also be used.

The mass properties as determined by the SPHERES CAD model, the swinging parallelogram test [73], and a least squares mass identification test [66] are seen in Tables 7.1 to 7.3.

The expansion port weighs 110 g±0.25 g. The center of gravity of the expansion port is located approximately 10 cm in the +X direction from the geometric center of the SPHERES. For the purposes of predicting mass property changes, the expansion port was assumed to be a point mass. Using the parallel axis theorem, the expansion port was determined to add $1.1 \times 10^{-3}\,\mathrm{N\,m^2}$ to the inertia about the Y and Z axes.

As for the center of gravity, the SPHERES system with a full tank weighs 4.6 kg and empty weighs 4.43 kg. With the addition of the expansion port, the center of

Table 7.3: SPHERES KC-135 Mass Property Results

Values	Empty Tank	Units
j_{xx}	2.20×10^{-2}	$\mathrm{kg\,m^2}$
j_{yy}	1.97×10^{-2}	$\mathrm{kg\,m^2}$
j_{zz}	1.82×10^{-2}	$\mathrm{kg\,m^2}$
j_{xy}	1.96×10^{-4}	$\mathrm{kg\,m^2}$
j_{xz}	-5.5×10^{-5}	$\mathrm{kg\,m^2}$
j_{yz}	-2.15×10^{-4}	$\mathrm{kg\,m^2}$
c_x	-0.016	mm
c_y	-0.821	mm
c_z	3.082	mm

Reference Frame: Center of Mass

gravity should shift approximately 3 mm in the +X direction.

7.1.2 Mass Characterization Test

In order to characterize the SPHERES mass properties with the expansion port attached, a test was conducted during the 37th SPHERES test session, which occurred on 26 February 2013. This test consisted a thruster characterization maneuver and two mass property identification maneuvers.

The thruster characterization maneuver was conducted in order to estimate the force produced by a single thruster. The thruster force is infered using data from the accelerometers. The recorded acceleration of the acceleromter is described in Eq. (7.1).

$$\hat{a}_r = a_i - \omega \times (\omega \times r) - \dot{\omega} \times r + \gamma \tag{7.1}$$

where a_r is the acceleration in the rotating reference frame (which is the acceleration of interest), a_i is the acceleration in the inertial reference frame (which is what the accelerometers measure), ω is the angular velocity of the satellite, r is the position vector from the center of mass to the accelerometer, and γ is the sample noise of the acceleromter. Since the acceleromter is not moving with respect to the center of gravity, the Coriolis effect was not included in Eq. (7.1).

If the angular rate and angular acceleration of the satellite is kept sufficiently low,

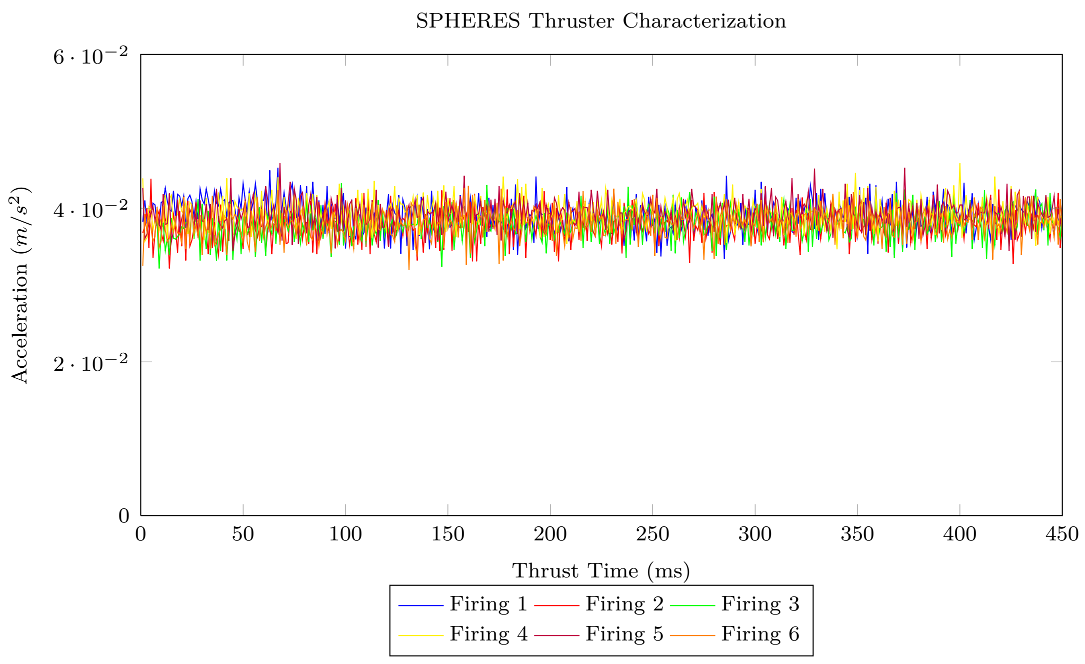


Figure 7-2: Results of Thruster Characterization Test

Eq. (7.1) can be reduced to

$$\hat{a}_r = a_i + \gamma \tag{7.2}$$

If $\mathbb{E}[\gamma] = 0$ and many samples of the acceleration are taken, Eq. (7.2) can then be approximated as

$$a_r = \sum_{i=1}^{n} \hat{a}_r^i \tag{7.3}$$

where n is the number of accelerometer samples and $\hat{a}_r^i$ is the ith accelerometer sample.

For the SPHERES thruster characterization test, the thruster pairs were chosen to impart as little angular acceleration as possible. Two thrusters were fired at a time, with the assumptions that the force produced by the thruster is constant and is the same for all thrusters. The results of the thruster firing test are seen in Fig. 7-2

The mean acceleration during the tests was $0.0385\,\mathrm{m/s^2}$. The maximum thruster force F_t is found by

$$F_t \cdot (1 - \epsilon)^{n-1} = \frac{m \cdot a_r}{n} \tag{7.4}$$

where n is the number of thrusters open at a time and ϵ is the reduction of thrust due to an additional thruster being fired. The thrust reduction has been characterized to be approximately $\epsilon = 0.06$[74]. For the case of SPHERES thruster characterization, $n = 2$.

The first mass property identification maneuver consisted of a set of single thruster firings. In this maneuver, one thruster at a time was fired for 0.6 s, then the system would coast for 0.4 s. This was repeated until all twelve SPHERES thrusters had fired. The second mass identification maneuver was a high speed spin without thruster firings. The satellite was spun up about a non-body axis and then allowed to spin for twelve seconds. The angular rates produced by these maneuvers are seen in Fig. 7-3.

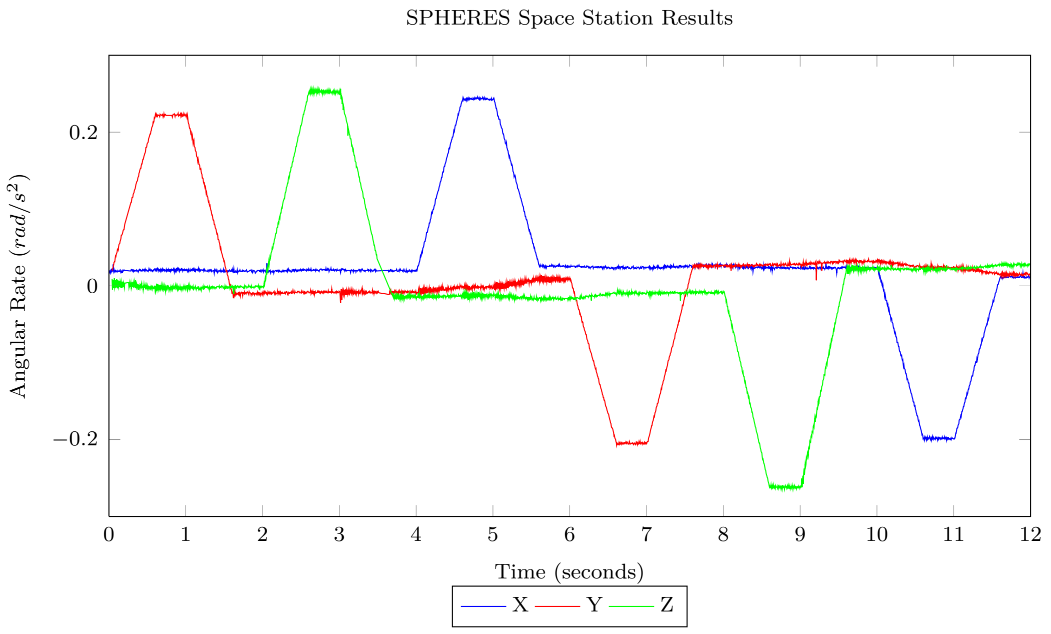
7.1.3 Results

Using the nonlinear programming method described in Chapter 6, the mass properties of the SPHERES satellite with the expansion port were found using data presented in Section 7.1.2. Since the exact mass and thruster reduction was not known, a Monte-Carlo based method was employed to estimate the expected value of the mass properties as well as the standard deviation of those values. The mass and thruser reduction were assumed to take the from of a Gaussian distribution. For the simulation, a mass and thruster reduction were sampled from the respective distribution. If the values were out of bounds, the values were resampled until a set of values that were in bounds were found. The parameters for the Monte-Carlo simulation are seen in Table 7.4.

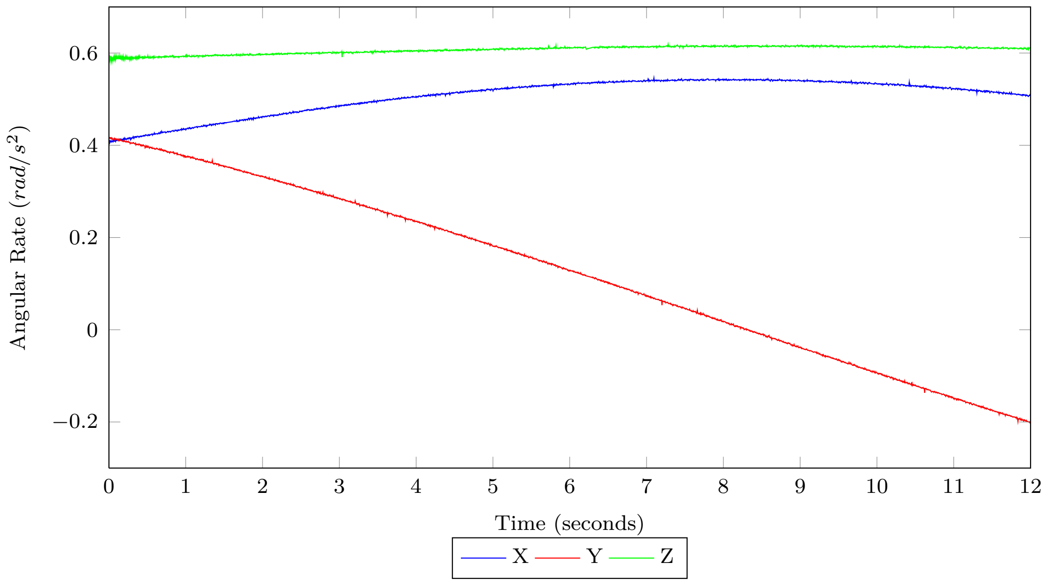
Table 7.4: Monte-Carlo Setup for SPHERES Expansion Port Mass Identification

Values	Mean	Std Dev	Min	Max	Units
Mass (m)	4.487	0.0567	4.43	4.6	kg
Thruster Reduction (ϵ)	0.06	0.01	0	0.12	Unitless

The results of the Monte-Carlo simulation are seen in Table 7.5.



(a) Single Thruster Firings.



(b) High Speed Spin.

Figure 7-3: Results of Thruster Characterization Test

Table 7.5: Monte-Carlo Results for SPHERES Expansion Port Mass Identification

Values	Mean	Std Dev	Units
j_{xx}	2.41×10^{-2}	3.24×10^{-4}	$\mathrm{kg\,m^2}$
j_{yy}	2.34×10^{-2}	3.15×10^{-4}	$\mathrm{kg\,m^2}$
j_{zz}	2.01×10^{-2}	2.70×10^{-4}	$\mathrm{kg\,m^2}$
j_{xy}	-1.30×10^{-4}	1.78×10^{-6}	$\mathrm{kg\,m^2}$
j_{xz}	-1.42×10^{-4}	1.90×10^{-6}	$\mathrm{kg\,m^2}$
j_{yz}	5.74×10^{-5}	8.05×10^{-7}	$\mathrm{kg\,m^2}$
c_x	3.88	1.12×10^{-3}	mm
c_y	-1.49	1.65×10^{-3}	mm
c_z	1.92	7.44×10^{-3}	mm

As expected, the center of mass moved approximately 3 mm in the +X direction. As for inertia, the inertia values fell in the general range of predicted changes, but the previous inertia predictions were not consistent enough to provide a good baseline.

7.2 VERTIGO

In November of 2012 the Visual Estimation for Relative Tracking and Inspection of Generic Objects (VERTIGO) system was launched to the International Space Station. VERTIGO consists of a stereo vision camera and an avionics box containing a small computer. The VERTIGO system is meant to be an additional sensor for SPHERES, enabling vision based navigation and tracking. The VERTIGO system, seen with astronaut Tom Marshburn, is seen in Fig. 7-4.

The VERTIGO system attaches to SPHERES on the expansion port. The system is connected electrically through a 40 pin connector and mechanically via the two thumb screws on the expansion port which thread into holes on the VERTIGO avionics box. Attaching VERTIGO to SPHERES obviously changes the mass properties of SPHERES, but the VERTIGO system may also impinge the thrusters on the +X face of SPHERES. Therefore, a full system characterization of the combined

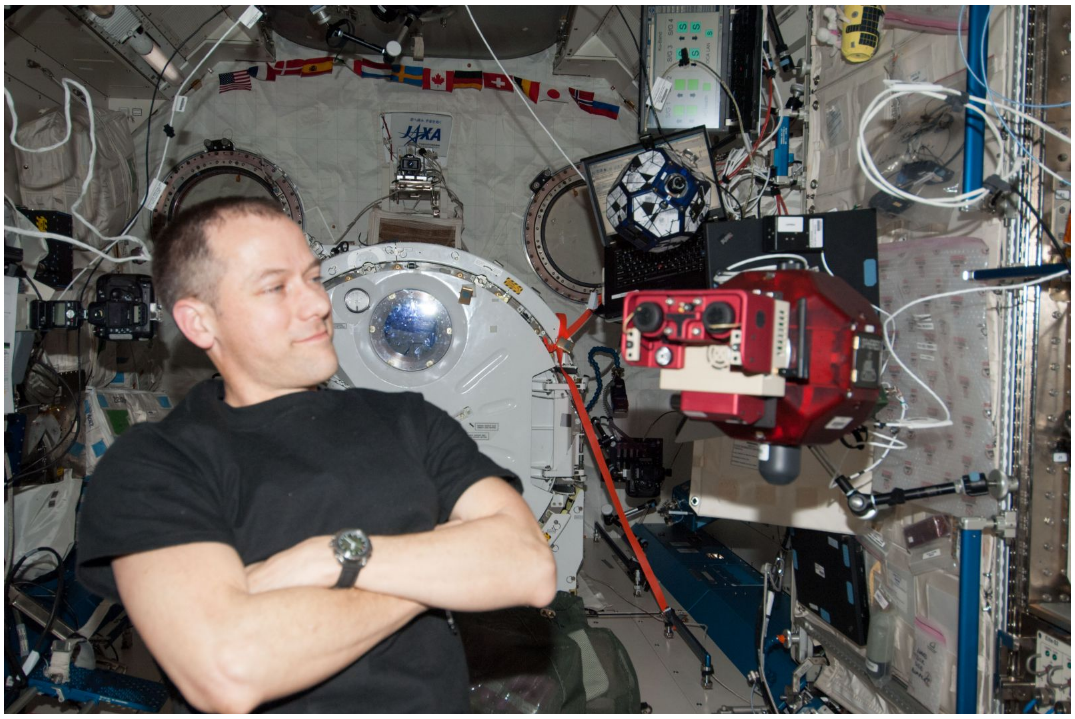


Figure 7-4: The VERTIGO System with NASA Astronaut Tom Marshburn [75]

SPHERES/VERTIGO system was performed.

7.2.1 Thruster Impingement

To test for thruster impingement, the acceleration of the spacecraft when unobstructed thrusters fire will be compared against the acceleration of the spacecraft when the potentially obstructed thrusters are fired. The VERTIGO system attached to the +X face of SPHERES, meaning the center of gravity of the combined system shifts significantly in the +X direction as compared to the SPHERES themselves. The center of mass of the VERTIGO system is approximately at the geometric center, meaning the center of mass of the combined system does not shift significantly in the Y or Z direction when compared to the SPHERES center of gravity.

To investigate the thruster impingement, a SPHERES test was developed that alternates between firing the +X and -X thrusters. The -X thrusters will serve as the control since they are unobstructed. The +X thrusters will assume to be equally obstructed. Performing an individual thruster characterization would require using

Eq. (7.1); however, calculating the center of mass requires knowing the thruster force, which is derived here. Therefore, the thrusters will have to be assumed as having the same impingement in order to derive the thruster force.

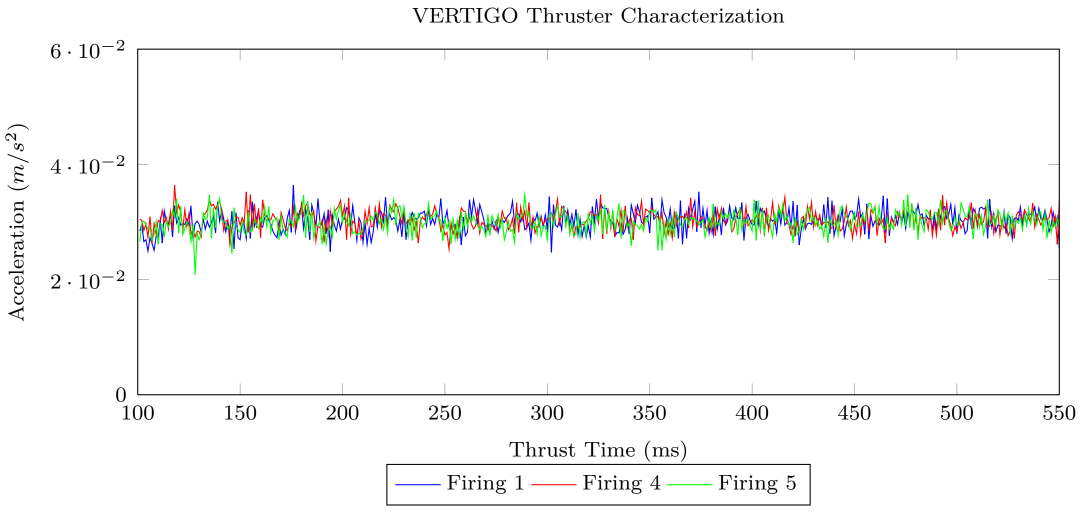
On March 12th, 2013 the SPHERES/VERTIGO test with the thruster characterization was performed on orbit. The measured accelerations are seen in Fig. 7-5. Firings 1,4, and 5 involved firing the unobstructed -X thrusters while firings 2,3, and 6 involved firing the potentially obstructed +X thrusters. The data for thruster firing 6 was disregarded because the angular rates of the satellite were too great for the assumptions used to derive Eq. (7.2) to hold.

The mean acceleration of the satellite was $0.0302\,\mathrm{m/s^2}$ for the -X thrusters and $0.211\,\mathrm{m/s^2}$ for the +X thrusters. However, this test violated the assumption that there were no external forces exerted on the SPHERES. The thruster characterization test involved two SPHERES, which were aligned along the X axis. In firings 1,4, and 5, the SPHERES with VERTIGO attached thrusted in such a manner that impeded the motion of secondary SPHERES. In firings 3,4, and 6, the secondary SPHERES fired in such a manner that impeded the motion of the SPHERES with VERTIGO attached.

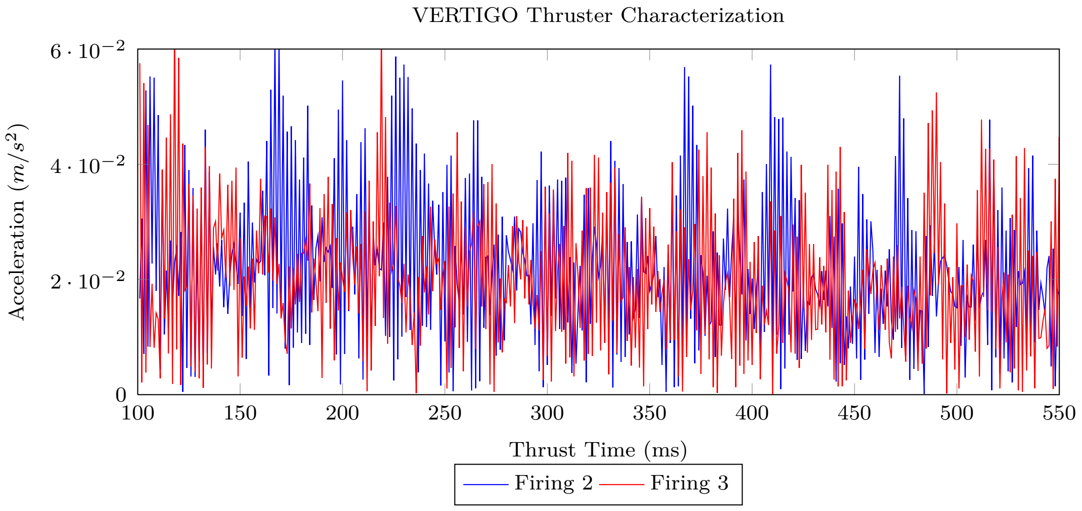
To adjust for this difference, a sum of equations can be used to remove the disturbing force from the measurement. Let F_{drag} be the drag force induced by the other SPHERES firing its thrusters. The adjusted acceleration can then be found in Eq. (7.5)

$$a_r = \sum_{i=1}^{n} \hat{a}_r^i - \frac{\mathrm{F}_{drag}}{m} \tag{7.5}$$

The drag force F_{drag} can be found by comparing the performance of the secondary SPHERES against the performance described in Section 7.1.2. The average acceleration measured for the SPHERES expansion port checkout during firings 2,3, and 5 was $0.0379\,\mathrm{m/s^2}$. In the VERTIGO test, the non-VERTIGO SPHERES had an average acceleration of $0.0387\,\mathrm{m/s^2}$. This is a difference of 2.2%, which can be attributed a slightly different mass or tank pressure. In contrast, the average acceleration of the non-VERTIGO SPHERES when the VERTIGO SPHERES thrusted against the



(a) Un-impinged Thrusters



(b) Impinged Thrusters

Figure 7-5: VERTIGO Thruster Characterization Results

Table 7.6: Monte-Carlo Setup for SPHERES/VERTIGO Mass Identification

Values	Mean	Variance	Min	Max	Units
Mass	6.346	0.02	5.5	7.0	kg
Thruster Reduction	0.06	0.01	0	0.12	Unitless

non-VERTIGO SPHERES was 0.0305 m/s^2. Assuming the mass was approximately the same between the two tests, the drag force is therefore approximately 0.0373 N.

The VERTIGO system has a mass of 1.746 kg$\pm$0.0025 kg. Using the combined SPHERES/VERTIGO mass along with Eq. (7.5), the adjusted accelerations for the -X firings become 0.0267 m/s^2. Since the thruster force scales linearly with the measured acceleration, this indicates that the VERTIGO system causes an 11.3% reduction in thruster force.

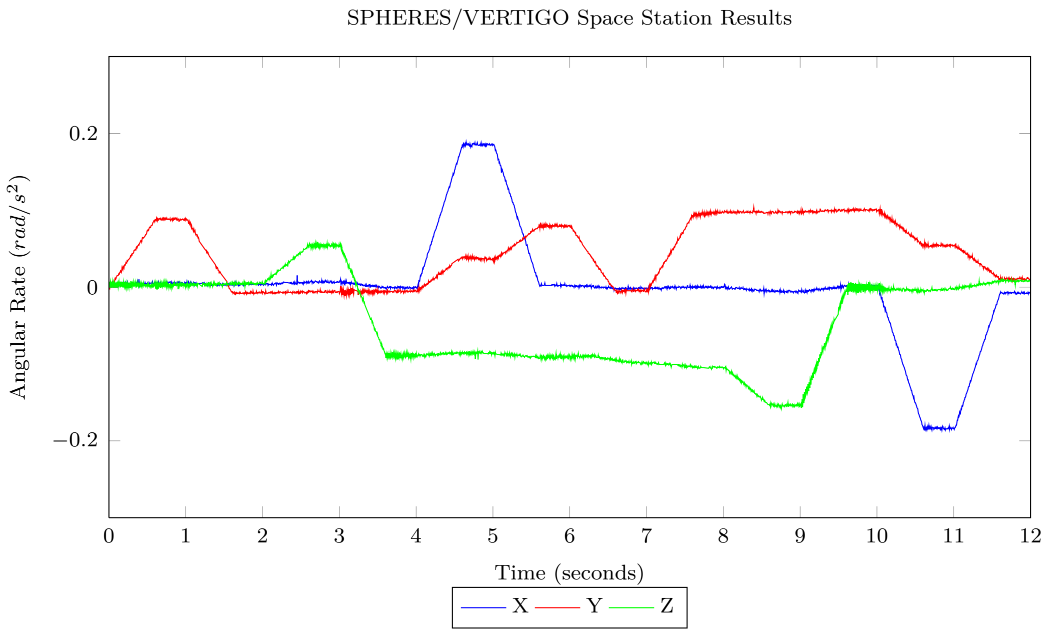
7.2.2 Mass Property Identification

In the same March 12th test, the same two mass identification maneuvers described in Section 7.1.2 were performed with the VERTIGO system attached to SPHERES. The angular rates produced by this test are seen in Fig. 7-6.

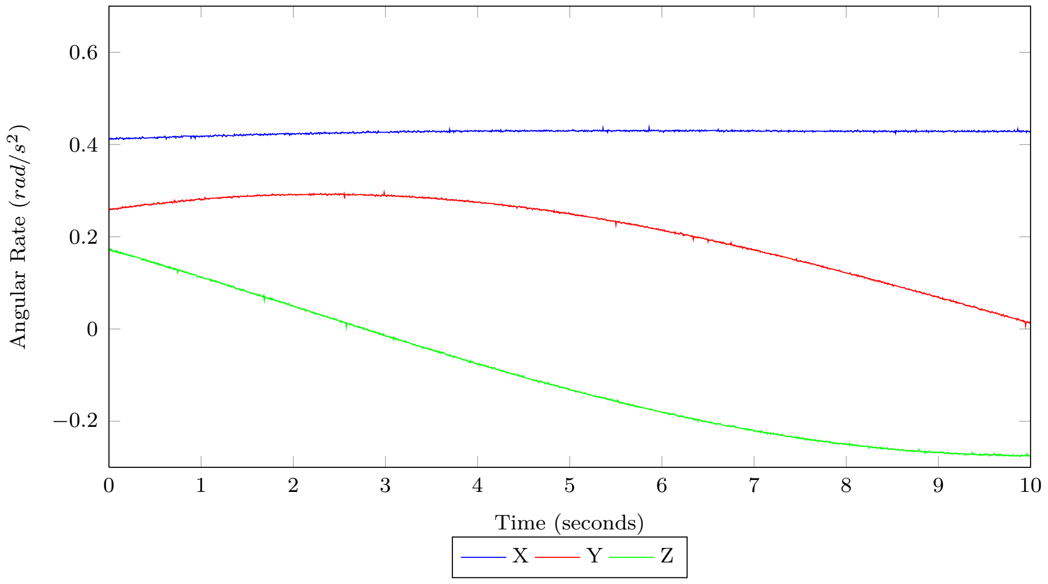
The same method described in Section 7.1.3 was used to calculate the SPHERES/VERTIGO mass properties. Unlike the SPHERES expansion port mass identification test, the SPHERES CO2 tank was replaced just prior to the test run, meaning the mass of the system was known with near-certainty. The parameters used for the Monte-Carlo simulation are seen in Table 7.6. The results of the Monte-Carlo simulation are seen in Table 7.7.

7.3 RINGS

In order to prepare for operations in the International Space Station, the engineering RINGS units were taken on a Reduced Gravity Aircraft (RGA) flight. During the four day campaign a number of tests were performed, one of which was a system characterization, which included both a thruster impingement test and a mass characterization test.



(a) Single Thruster Firings.



(b) High Speed Spin.

Figure 7-6: Results of VERTIGO Mass Identification Maneuvers

Table 7.7: Monte-Carlo Setup for SPHERES/VERTIGO Mass Identification

Values	Mean	Std Dev	Units
j_{xx}	3.19×10^{-2}	3.59×10^{-4}	$\mathrm{kg\,m^2}$
j_{yy}	6.12×10^{-2}	6.90×10^{-4}	$\mathrm{kg\,m^2}$
j_{zz}	5.85×10^{-2}	6.60×10^{-4}	$\mathrm{kg\,m^2}$
j_{xy}	-3.51×10^{-4}	4.05×10^{-6}	$\mathrm{kg\,m^2}$
j_{xz}	-2.00×10^{-4}	2.24×10^{-5}	$\mathrm{kg\,m^2}$
j_{yz}	7.18×10^{-5}	7.29×10^{-7}	$\mathrm{kg\,m^2}$
c_x	47.1	3.69×10^{-3}	mm
c_y	-0.781	5.18×10^{-4}	mm
c_z	1.66	9.79×10^{-4}	mm

7.3.1 Expected Results

The addition of the RINGS system changes the mass properties of the SPHERES system much more significantly than the expansion port or VERTIGO. In order to ensure the RGA flights would produce the desired data, analysis was performed prior to the flight in order to ensure the data collected would contain the necessary resolution for a thruster characterization and mass identification.

Thruster Characterization

For the thruster characterization test, thruster pairs will be fired in order to measure the acceleration of the system without inducing significant angular rates. The SPHERES thrusters nominally produce about 0.1 N of thrust. The combined SPHERES and RINGS system has a mass of 17.1 kg. If the thrusters are not impinged in any manner, then the system should accelerate at a rate of $11.7 \times 10^{-3}\,\mathrm{m/s^2}$ when a thruster pair is fired. The noise of the accelerometers has a measured standard deviation of $1.9 \times 10^{-3}\,\mathrm{m/s^2}$, meaning the acceleration of the RINGS system will be discernible from the accelerometer data.

The amount of expected impingement varies based on the thruster direction. The thrusters pointed in the positive or negative X direction will be severely impinged, as they point directly at the RINGS coil housing. The positive and negative Y thrusters are not physically obstructed and should not show any impingement. The positive and negative Z thrusters are generally unobstructed but may interact slightly with

Table 7.8: RINGS Center of Mass Ground Testing Results

Axis	Value	Uncertainty	Units
X	-2.5	± 0.5	cm
Z	0.19	± 0.5	cm

the outer coil housing.

Mass Property Identification

In order to estimate the mass properties of the combined SPHERES/RINGS system, analysis was performed on both the center of mass and the expected inertia. In order to find the center of mass of the system on the ground, a test setup seen in Fig. 7-7.

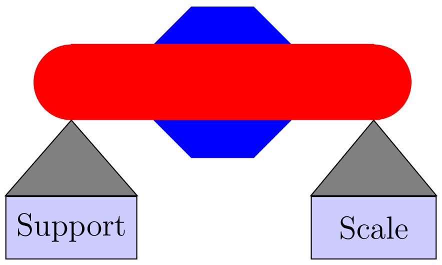


Figure 7-7: RINGS Center of Gravity Ground Test

If the distance between supports d is known, the distance of the center of mass from the support can be found by using Eq. (7.6)

$$c = \frac{d \cdot s}{m} \tag{7.6}$$

where s is the scale readout, m is the mass of the system, and g is the acceleration due to gravity. This setup assumes that the system is perfectly level. For the RINGS system, this setup can be used to measure the location of center of mass in the SPHERES body frame. The results of these test are seen in Table 7.8.

The parallel axis theorem was used to estimate the inertia of the RINGS system. First, the major components of RINGS where weighed. Next, basic inertia approximates were found for each component. The inertia of each component was computed, as well as the inertia caused by the center of mass being offset from the total center

Table 7.9: RINGS Inertia Prediction Calculations

Component	Mass (kg)	Approximation	CoM Offset (m)	Total Inertia ($\mathrm{kg\,m^2}$)
Coil	3.52	Hoop	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.180\\0.360\\0.180\end{bmatrix}$
Shell	1.936	Hoop	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.099\\0.198\\0.099\end{bmatrix}$
4 × Cuff	0.151	Point Mass	$\begin{bmatrix}\pm0.228\\0\\\pm0.228\end{bmatrix}$	$\begin{bmatrix}0.031\\0.062\\0.031\end{bmatrix}$
2 × Battery	0.673	Point Mass	$\begin{bmatrix}-0.171\\0\\\pm0.089\end{bmatrix}$	$\begin{bmatrix}0.011\\0.050\\0.039\end{bmatrix}$
2 × Battery Holder	0.236	Point Mass	$\begin{bmatrix}-0.171\\0\\\pm0.089\end{bmatrix}$	$\begin{bmatrix}0.004\\0.018\\0.014\end{bmatrix}$
Avionics	1.125	Point Mass	$\begin{bmatrix}0.248\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.0\\0.069\\0.069\end{bmatrix}$

Table 7.10: RINGS Inertia Prediction

Axis	J ($\mathrm{kg\,m^2}$)
X	0.325
Y	0.757
Z	0.432

of mass. Table 7.9 contains the list of components evaluated and their associated inertia.

From Table 7.9, the estimated inertia of the RINGS system is seen in Table 7.10. These are inertia will be an underestimate of the actual inertia, since several components, such as the fans, bolts, and SPHERES cuff, were not included. This numbers are to serve as a baseline to ensure the inertia found from the RGA flight were the same order of magnitude.

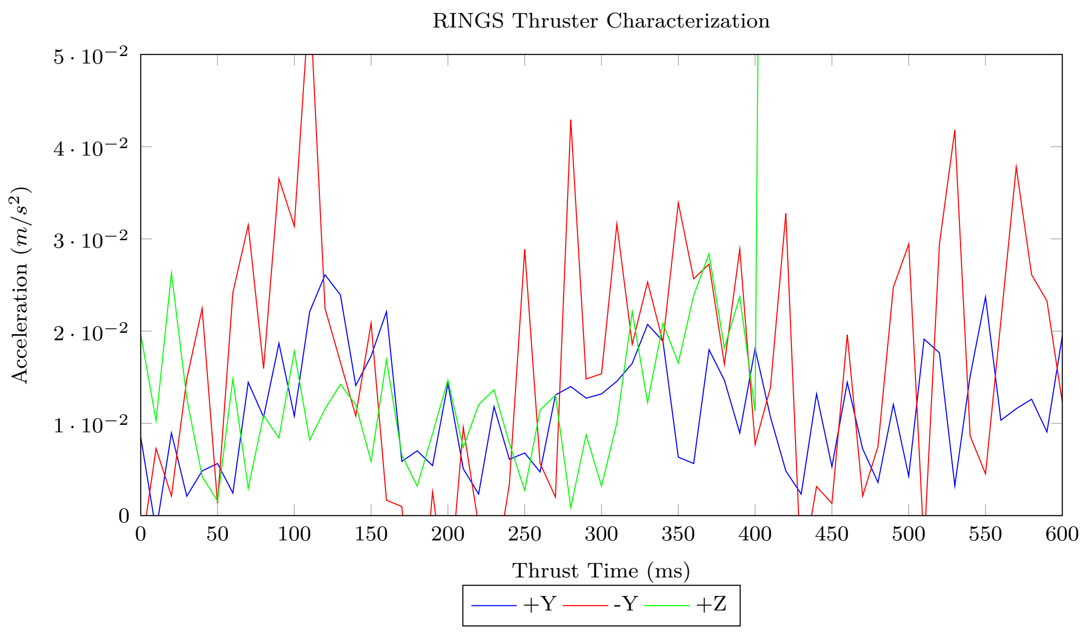


Figure 7-8: RINGS Thruster Characterization Results

7.3.2 Thruster Impingement

The RINGS units severely obstruct several of the thrusters. In order to characterize the thurster impingement, a test was written that fires pairs of thrusters corresponding to each thruster direction (+X,-X,+Y,-Y,+Z,-Z). The center of mass of the RINGS unit is approximately the same as the SPHERES center of mass, so the assumptions used to derive Eq. (7.2) hold.

The acceleration in the positive and negative X direction was indescribable from the noise. This indicates that the associated thrusters are severely impinged. This agrees with the prediction. The results of the +Y,-Y, and +Z thrusts are seen in Fig. 7-8. The satellite was disturbed before it could complete the -Z thruster firing.

The results indicate that the Y and Z axis thrusters are not impinged by RINGS.

7.3.3 Mass Property Identification

The same mass identification maneuvers performed in Sections 7.1.2 and 7.2.2 were performed on the RGA flight. However, despite running the test several times, the satellite was disturbed before it could finish the test. Also, the significant increase

Table 7.11: RINGS MassID RGA Test Results

	$\Delta\omega$ (rad/s)	
Run Number	X	Z
1	0.1399	0.0838
2	0.1357	0.0867
3	0.1371	0.1044
4		0.0945
Average	0.1376	0.0923
Std Dev	0.0021	0.0092

Table 7.12: RINGS Inertia Experimental Results

Axis	$\bar{\alpha}$ (rad/s^2)	J (kg m^2)
X	0.0459±0.0014	0.3742 $^{+0.0118}_{-0.0110}$
Z	0.0308±0.0061	0.5577 $^{+0.1377}_{-0.0922}$

in inertia of the system made distinguishing induced motion from noise difficult. Therefore, long duration torques were executed in order to find the major moments of inertia.

Given that the tests were three seconds long, the average angular acceleration $\bar{\alpha}$ can be computed. To compute the error bounds, the 95% confidence interval was used. The average angular acceleration is documented in Table 7.12. Given that the angular acceleration of the system is governed by

$$\tau = J \cdot \alpha \tag{7.7}$$

where τ is the torque on the system, the inertia J about that axis can be computed. The torque τ was computed by crossing the force vector with the moment arm. The magnitude of the force vector was assumed to be 0.089 N. Since the thruster characterization of RINGS did not provide enough data for a qualitative force analysis, the thruster value was derived from the analysis in Section 7.1.2. As for the moment arm, the RINGS center of mass was assumed to be at the body center. The moment arm is therefore 0.0965 m. Using Eq. (7.7), the inertia about the X and Z axis can be computed and are shown in Table 7.12.

The results in Table 7.12 can then be compared to the predictions in Table 7.10 in order to estimate the inertia about the Y axis. The experimental results indicate that the actual inertia was 15.1% and 29.1% greater than the prediction for the X and Z axis, respectively. Therefore, the Y axis can be estimated as 22.1% (the average of the two other axes) greater than the prediction, making the predicted Y axis inertia approximately $0.924\,\mathrm{kg\,m^2}$. To compute the error bounds, the smallest possible scale factor and largest possible scale factors possible, given the uncertainty in Table 7.12. These values are 7.8% and 61.0%, respectively, and are driven by the uncertainty in the Z axis inertia. Therefore, the Y axis inertia is estimated to be between $0.816\,\mathrm{kg\,m^2}$ and $1.219\,\mathrm{kg\,m^2}$. While the uncertainty for this axis is rather large, EMFF forces will produce an insignificant amount of torque about the Y axis, meaning the only motion about the Y axis will be induced from either the initial release of RINGS or thrusters. Given the severe thruster impingement of the thrusters required to create a torque about the Y axis and the significant amount of inertia about this axis, little can be done to change the angular velocity of RINGS about the Y axis, making the inertia about the Y axis of little use.

Chapter 8

Model-Free State Estimation Using Line-of-Sight Transmitters

Model-based estimation and control has become common practice in control engineering; however, the success of these estimators and controllers depends on the accuracy of the model used. To validate a model, the model is typically compared against real-world measurements. The catch is that the model cannot be used when generating the real-world measurements; otherwise the measurements are biased by the unproven model.

The SPHERES testbed uses a model-based filter for state estimation. When new models, such as the RINGS, need to be validated against the real system, the model-based filter cannot be used for reasons stated previously. This chapter will present a method for estimating the satellite state without using a model of the satellite. Instead, the SPHERES metrology system, which relies on line-of-sight ultrasound measurements, will be used to generate the system state. While SPHERES is used as a case study, this process can be applied to any system uses line-of-sight measurements, so long as it satisfies the assumptions listed in Section 8.2.

8.1 Motivation

The performance of a dynamic programming controller is heavily dependent on the accuracy of the model used to generate the cost-to-go. Therefore, model validation is essential when applying dynamic programming to a physical system. A dynamic programming controller is "doomed to succeed" when applied in simulation, since the simulation will most likely use the same model to simulate the trajectories as the one used to generate the cost-to-go. If the model does not accurately represent the system, then the controller will perform poorly on the system despite performing well in simulation.

Validating the model of the system may not be a trivial process either. Model validation typically involves inferring some performance over the envelope of potential states of the system. This requires knowledge of the system state. Model-based filters, such as the Kalman Filter, Extended Kalman Filter, and Particle Filter have become the standard for state estimation; however, estimators cannot be used during model validation for two reasons:

- Using a model to predict the state vector would bias the state vector, meaning the model validation would be biased towards the model used in the estimator.

- If a model has not been validated, it is inadvisable to use it in the filter anyway.

Given these reasons, when performing model validation, it is necessary to use a model-free method when obtaining the state vector for model validation.

While some model validation for spacecraft can be performed on the ground, it may be necessary to move to a microgravity environment for a full model validation. For example, the RINGS system produces small amounts of oscillating force. These forces are very hard to measure on the ground because of the mass of the system causes the electromagnetic forces to show up in the noise of any force measurements. Precise accelerometers are needed to capture the intra-vehicular forces of RINGS, but these same accelerometers saturate in the presence of gravity. Therefore, a micro-gravity environment is the best environment for RINGS model validation.

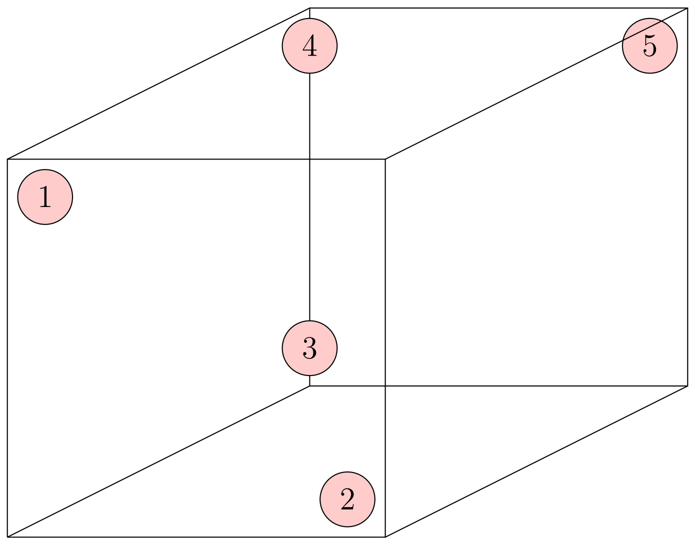


Figure 8-1: Example Transmitter Setup

For spacecraft, the state vector typically consists of position, attitude, and their associated rates of change. In a micro-gravity testbed environment, such as a Reduced Gravity Aircraft (RGA) or on-board the International Space Station, there are a few options for producing a state measurement, such as stereoscopic vision or LIDAR. However, these methods are complex, require significant power, and are expensive. A simple and cheap method of measuring position and attitude is using a line-of-sight transmitter/receiver system. While there has been previous work on how to find position and attitude using line-of-sight measurements [76], these methods rely on models of the system. This chapter will discuss how to use such a system to estimate the position and attitude of a system without using a system model.

8.2 Problem Formulation

A typical transmitter/receiver system involves several transmitters placed at various locations in the area where the system will be operating. Since the system is assumed to require line-of-sight from the transmitter to receiver in order to receive a measurement, the transmitters are usually placed at the extent of the working volume pointed inward. Fig. 8-1 illustrates the locations of transmitters used by the SPHERES testbed onboard the International Space Station.

The system being estimated is assumed to have multiple receivers, places in various positions and orientations on the system. This analysis will be using the SPHERES receiver setup, illustrated in Fig. 8-2. The location of each receiver is listed in Table 8.1.

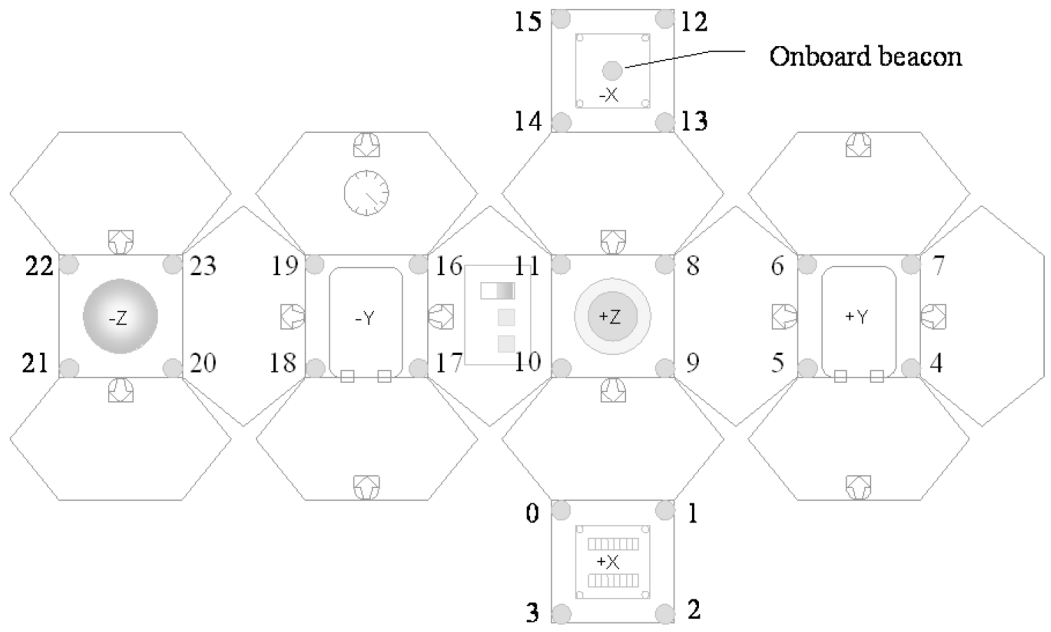


Figure 8-2: SPHERES Receiver Locations [77]

The assumptions used for this analysis are summarized in Table 8.2.

8.3 Position Determination

Although it may seem counter-intuitive, it is actually advantageous to decouple the position and attitude estimation. An estimate of the relative attitude of the spacecraft to the transmitter can be inferred from the beacon measurements, decoupling the attitude and position estimation. Once the position of the spacecraft is estimated, a more accurate estimation of the attitude can be made.

Table 8.1: SPHERES Receiver Locations [77]

Face	Receiver Number	Location (cm) X	Y	Z
+X	0	10.23	-3.92	3.94
	1	10.23	3.92	3.94
	2	10.23	3.92	-3.94
	3	10.23	-3.92	-3.94
+Y	4	3.94	10.23	-3.92
	5	3.94	10.23	3.92
	6	-3.94	10.23	3.92
	7	-3.94	10.23	-3.92
+Z	8	-3.92	3.94	10.26
	9	3.92	3.94	10.26
	10	3.92	-3.94	10.26
	11	-3.92	-3.94	10.26
-X	12	-10.23	3.92	-3.94
	13	-10.23	3.92	3.94
	14	-10.23	-3.92	3.94
	15	-10.23	-3.92	-3.94
-Y	16	-3.94	-10.23	3.92
	17	3.94	-10.23	3.92
	18	3.94	-10.23	-3.92
	19	-3.94	-10.23	-3.92
-Z	20	3.92	-3.94	-10.23
	21	3.92	3.94	-10.23
	22	-3.92	3.94	-10.23
	23	-3.92	-3.94	-10.23

Table 8.2: Model-Free State Assumptions

#	Assumption
1	There are measurements available from (at least) four unique transmitters
2	The measurement received by the system for each receiver is proportional to the distance between the receiver and the transmitter
3	The proportion listed in Assumption 1 is the same for all receivers and transmitters
4	The location of the receivers are fixed in the body frame
5	The location of the transmitters are fixed relative to the global frame

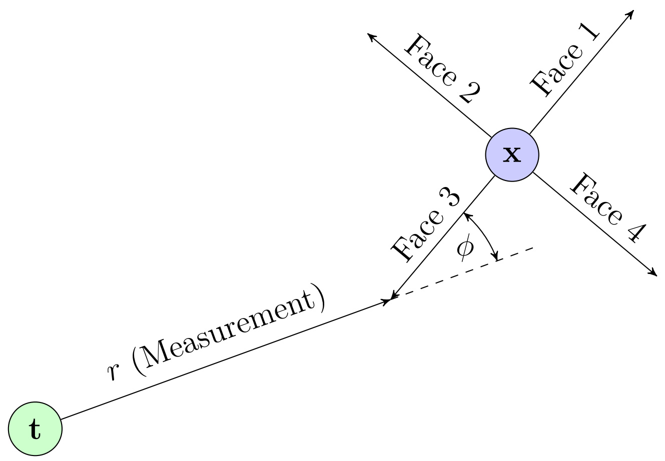


Figure 8-3: Transmitter Distance Illustration

8.3.1 Derivation

The SPHERES are designed such that no transmitter will have a bearing angle ϕ of more than 54.7° with a face of the satellite. Although this assumption may not hold for all systems, this analysis can be adapted for a system with different maximum bearing angles. Let $\mathbf{t}$ be the location of the transmitter and $\mathbf{x}$ be the location of the geometric center of the system. An illustration of the bearing angle concept (in two dimensions) is seen in Fig. 8-3.

Let f be the distance from the geometric center of the system to a face. The distance between between $\mathbf{t}$ and $\mathbf{x}$, defined at d is governed by Eq. (8.1).

$$d = r + f \cdot \cos(\phi) \tag{8.1}$$

For SPHERES, the center of a face is always 10.23 cm from the geometric center of the satellite.

Since, at this point, attitude is assumed to be random. Therefore, a probability distribution function of ϕ can be developed. The euler angles, used to describe the attitude of the satellite, are assumed to be random variables with a uniform probability distribution between 0 and 2π. The resulting probability distribution function for ϕ is seen in Fig. 8-4(a). Using Fig. 8-4(a), the probability distribution function for the $\cos\phi$ term of Eq. (8.1) can also be computed. The resulting probability distribution function for the $\cos\phi$ term is seen in Fig. 8-4(b). From Fig. 8-4(b), it holds that $\mathbb{E}[\cos(\phi)] = 0.8576$.

There are two general ways for proceeding given Fig. 8-4(b). The first would be the use the expected value of Eq. (8.1), seen in Eq. (8.2), in further analysis.

$$\mathbb{E}[d] = r + 0.8576f \tag{8.2}$$

This would be advantageous for an on-line solution, where accuracy may need to be traded for computational efficiency. The other option is to compute a Monte-Carlo simulation of the position analysis. This not only produces the expected value of the position, but also gives error bounds. This method requires more computational power, and is not ideal for on-line implementations.

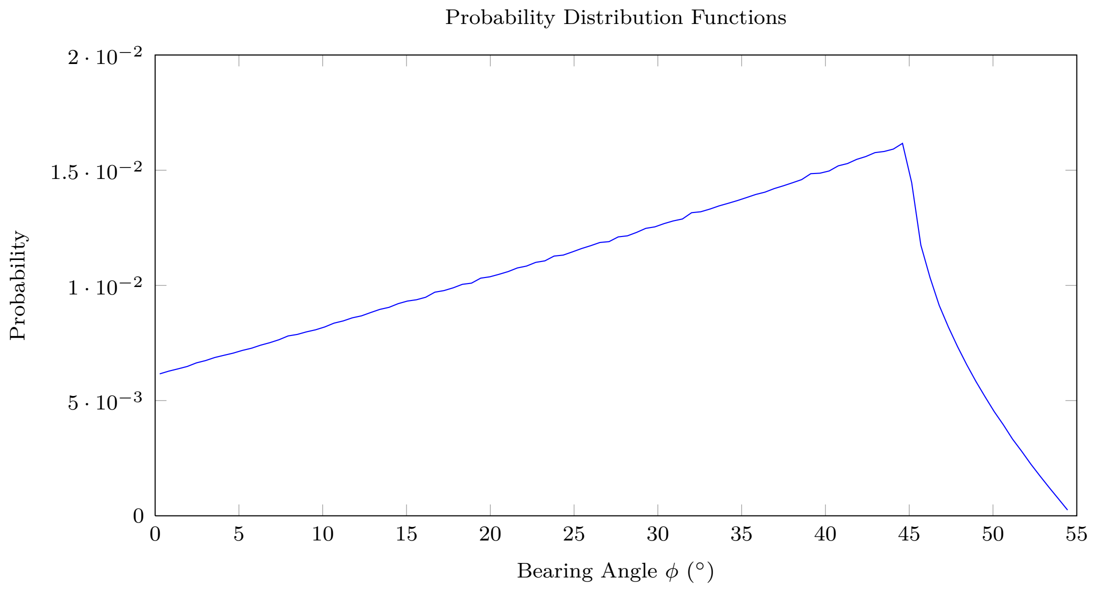
The measurement r from the receiver most likely does not return the exact distance from the transmitter to the receiver. Instead, it will return a value *proportional to* the distance between the transmitter and receiver. The relation between r and the measurement m is seen in Eq. (8.3)

$$r = s \cdot m \tag{8.3}$$

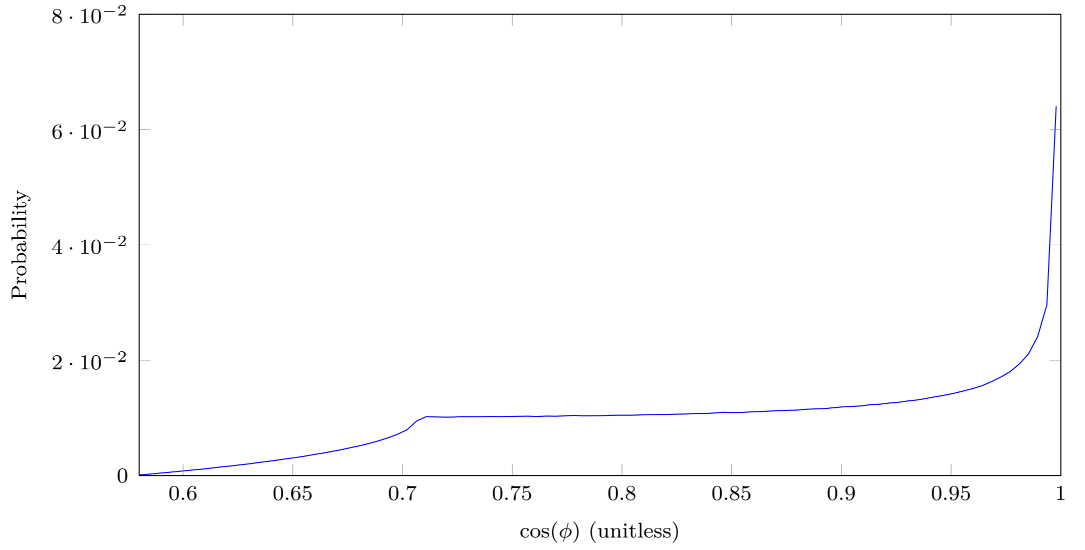
where s is the scale factor. The scale factor may depend on a number of factors. For example, the SPHERES receiver measurement depends on the processor clock period and the speed of sound. While the processor clock period may be known, the speed of sound is dependent on temperature, which may not be known at the time. However, Assumption 2 of Table 8.2 states that this scale factor must be the same for all transmitters.

Let $\mathbf{t}_i$ be the location of transmitter i in the global frame. The system center of mass $\mathbf{x}$ and scale factor s are then determined by the system of equations described by Eqs. (8.4a) to (8.4d). While there a multitude of methods for solving nonlinear equation systems [78], a basic Newton method will be presented in Section 8.3.2.

$$(s \cdot m_1 + f \cdot \mathbb{E}[\cos\phi])^2 = (\mathbf{x} - \mathbf{t}_1)^\top (\mathbf{x} - \mathbf{t}_1) \tag{8.4a}$$



(a) Bearing Angle Probability Distribution Function.



(b) Added Distance Probability Distribution Function.

Figure 8-4: Bearing Angle and Range Adjustment Probability Distribution Functions

$$(s \cdot m_2 + f \cdot \mathbb{E}[\cos\phi])^2 = (\mathbf{x} - \mathbf{t}_2)^\top(\mathbf{x} - \mathbf{t}_2) \tag{8.4b}$$

$$(s \cdot m_3 + f \cdot \mathbb{E}[\cos\phi])^2 = (\mathbf{x} - \mathbf{t}_3)^\top(\mathbf{x} - \mathbf{t}_3) \tag{8.4c}$$

$$(s \cdot m_4 + f \cdot \mathbb{E}[\cos\phi])^2 = (\mathbf{x} - \mathbf{t}_4)^\top(\mathbf{x} - \mathbf{t}_4) \tag{8.4d}$$

If a Monte Carlo simulation was desired in order to more precisely predict the position, then $\mathbb{E}[\cos\phi]$ in Eqs. (8.4a) to (8.4d) would be replaced by a random variable α. For the Monte Carlo analysis, α would be randomly sampled from the $\cos\phi$ probability distribution defined in Fig. 8-4(b).

When solving the system of equations defined by Eqs. (8.4a) to (8.4d), there will be four answers. In order to choose the proper answer, an estimate of the proper scale factor is required. Let $\hat{s}$ be the estimated scale factor and $\tilde{s}$ be the set of solutions for s, with $\tilde{s}_i$ being the ith solution. The proper s, defined as s^* can be found using Eq. (8.5).

$$s^* = \min_{i \in \{1,2,3,4\}} (\tilde{s}_i - \hat{s})^2 \tag{8.5}$$

If there are more than four transmitters providing measurements, then the system is over-determined. If the system is over-determined, there are two general options for computing the position estimate:

- Use measurements that correspond to the best four transmitter readings
- Permute the transmitters used

Both methods attempt to use the extra measurements to reduce the uncertainty in the measurement.

8.3.2 Solving for Position

The algorithm for determining the best face measurement is seen in Algorithm 7. It is important to note that because the receivers are symmetrical distributed about a face, the measurement for the face is the mean of the receiver measurements. While this algorithm is designed for SPHERES, it can easily be adapted for other systems.

Algorithm 7 Four Transmitter Position Determination

```
Given $m_{ij}$, $i = \{1, ..., N\}$, $j = \{1, ..., 24\}$        ▷ $N$ is number of transmitters
for $i = 1, ..., N$ do                                          ▷ Loop through the four transmitters
    for $j = 0, ..., 5$ do                                      ▷ Loop through all faces
        $\beta_j = \frac{1}{4}\sum_{k=4j}^{4j+3} m_{ij}$         ▷ Find measurement for face $j$
    end for
    $\hat{m}_i = \min_{j=\{0,1,2,3,4,5\}} \beta_j$               ▷ Find the best face measurement
end for
Return $\hat{m}_i$, $i = \{1, ..., N\}$
```

With the face measurements determined, they can be used to solve for $\mathbf{x}$ and s using Eqs. (8.4a) to (8.4d). Let the search variable z be defined as follows:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ s \end{bmatrix} \tag{8.6}$$

In order to solve for z, Eqs. (8.4a) to (8.4d) must be set equal to zero. The updated equations are seen in Eqs. (8.7a) to (8.7d).

$$(s \cdot m_1 + f \cdot \mathbb{E}[\cos\phi])^2 - (\mathbf{x} - \mathbf{t}_1)^\top(\mathbf{x} - \mathbf{t}_1) = 0 \tag{8.7a}$$

$$(s \cdot m_2 + f \cdot \mathbb{E}[\cos\phi])^2 - (\mathbf{x} - \mathbf{t}_2)^\top(\mathbf{x} - \mathbf{t}_2) = 0 \tag{8.7b}$$

$$(s \cdot m_3 + f \cdot \mathbb{E}[\cos\phi])^2 - (\mathbf{x} - \mathbf{t}_3)^\top(\mathbf{x} - \mathbf{t}_3) = 0 \tag{8.7c}$$

$$(s \cdot m_4 + f \cdot \mathbb{E}[\cos\phi])^2 - (\mathbf{x} - \mathbf{t}_4)^\top(\mathbf{x} - \mathbf{t}_4) = 0 \tag{8.7d}$$

Eqs. (8.7a) to (8.7d) can then be compacted to the form

$$\mathbf{f}(\mathbf{z}) = 0 \tag{8.8}$$

where $\mathbf{f}(\bullet)$ is the vector function comprised of Eqs. (8.7a) to (8.7d). The Jacobian of f, written as $\partial\mathbf{f}/\partial\mathbf{z}$, is defined in Eq. (8.9).

$$\frac{\partial\mathbf{f}}{\partial\mathbf{z}} = 2\begin{bmatrix} -(\mathbf{x} - \mathbf{t}_1) & -(\mathbf{x} - \mathbf{t}_2) & -(\mathbf{x} - \mathbf{t}_3) & -(\mathbf{x} - \mathbf{t}_4) \\ s & s & s & s \end{bmatrix} \tag{8.9}$$

Using the Jacobian defined in Eq. (8.9), it is possible to iteratively solve for z using Newton's Method. A basic form of Newton's Method applied to the position determination problem is seen in Algorithm 8.

Algorithm 8 Newton's Method for Position Determination

```
Initialize $\mathbf{z}_1 = \begin{bmatrix} \mathbf{0} \\ \hat{s} \end{bmatrix}$
$i \leftarrow 0$
**repeat**
    $i = i + 1$
    $\mathbf{z}_{i+1} = \mathbf{z}_i - \left[\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right]^{-1} \mathbf{f}(\mathbf{z}_i)$
**until** $\|\mathbf{z}_{i+1} - \mathbf{z}_i\| < \delta$                    ▷ $\delta$ is the convergence criteria
```

While Newton's Method is simple, it is not perfect. If $\partial \mathbf{f}/\partial \mathbf{z}$ becomes singular, it will no longer be invertable and the method will break. There are other methods that can circumvent this problem [71], Newton's Method is arguably the most simple.

8.3.3 Simulations

In order to analyze the performance of this method, a Monte Carlo simulation was conducted to characterize the system. The parameters of the system are seen in Table 8.3. The results of the simulation are seen in Fig. 8-5(a). As expected, the error of the system increases as the amount of noise in the measurements increases.

Table 8.3: Model-Free State Assumptions

Parameter	Value		
	X	Y	Z
Transmitter Locations	-1.181	-0.792	-0.833
	1.181	0.792	0.833
	1.181	-0.792	0.833
	1.181	0.792	-0.833
	-1.181	0.792	0.833
Mean Position	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$ m		
Position Variance	1.0 m		

In Fig. 8-5(b), an over-determined system is compared four-transmitter system. The four-transmitter system had an average error of 3.63 cm, while the over-determined system had an average error of 2.33 cm, meaning the over-determined system had a 36% lower average error.

8.4 Attitude Determination

With the position of the system determined, the next step is determining the attitude of the system. For this analysis, the attitude, which has three degrees of freedom, is assumed to be over-determined. The position analysis also determined the appropriate scale factor for the measurements, making determining the attitude easier.

8.4.1 Derivation

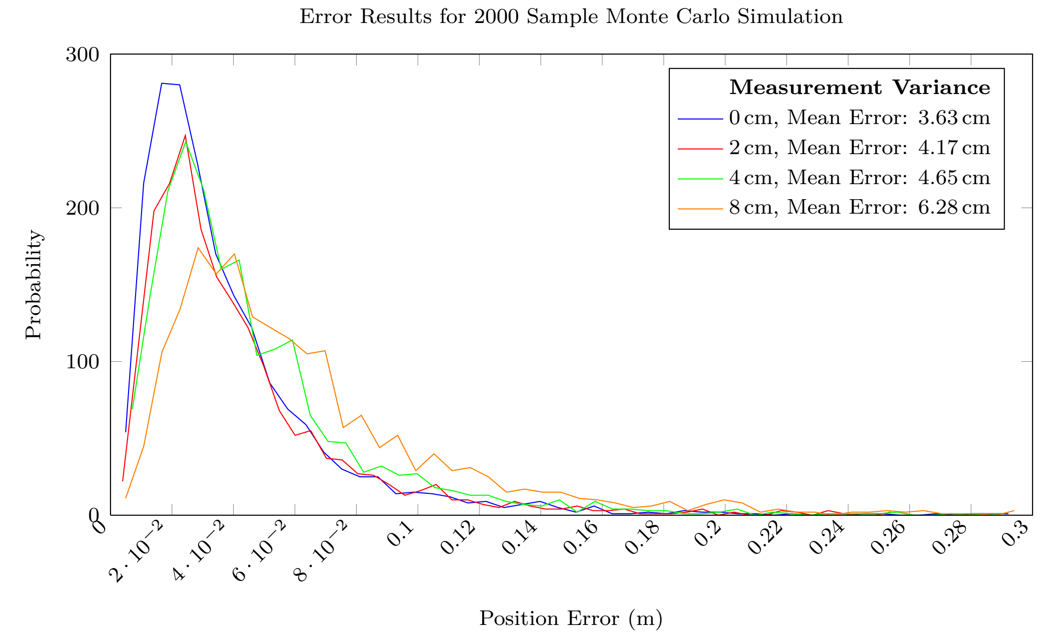
Let $\boldsymbol{\theta}$ be the vector of Euler angles describing the system attitude in the global frame, and θ_i be the ith element of $\boldsymbol{\theta}$. Further, let c_i, $i = 1, 2, 3$ be the cosine of θ_i and s_i, $i = 1, 2, 3$ be the sine of θ_i. The rotation matrix $R(\boldsymbol{\theta})$ can be found by using the 1-2-3 rotation sequence about the X, Z, and X axes, denoted in Eq. (8.10).

$$R(\boldsymbol{\theta}) = \begin{bmatrix} c_2 & -c_3 s_2 & s_2 s_3 \\ c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 \\ s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix} \tag{8.10}$$

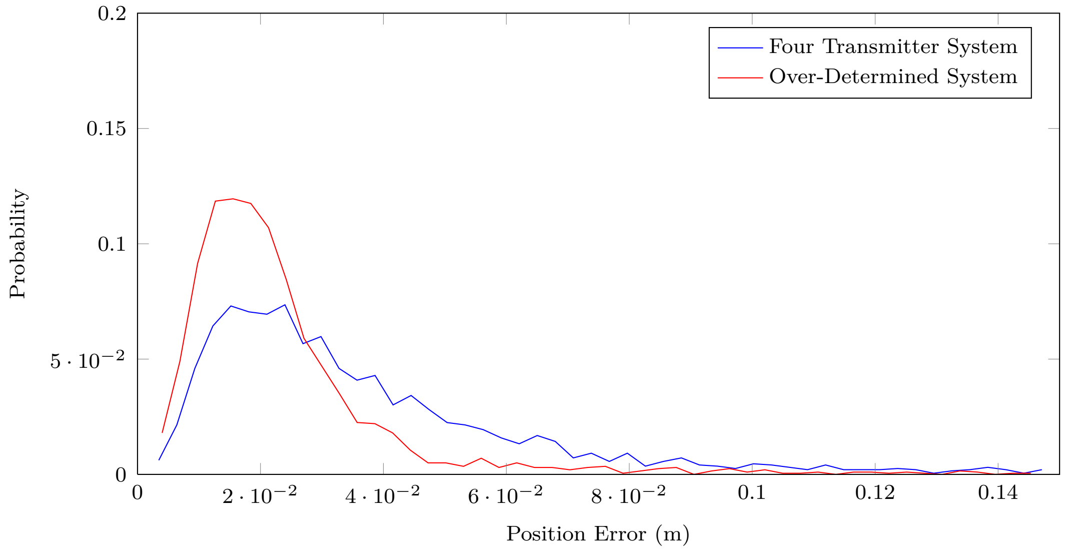
Further, let $\mathbf{g}_{ij}$ be distance from transmitter i to receiver j, while $\mathbf{b}_j$ is the vector from the system center to receiver j. The value of $\mathbf{g}_{ij}$ can be computed using Eq. (8.11).

$$\mathbf{g}_{ij}(\boldsymbol{\theta}) = \mathbf{x} + R(\boldsymbol{\theta}) \cdot \mathbf{b}_j - \mathbf{t}_i \tag{8.11}$$

Since the attitude is assumed to be over-determined, there is no explicit function that can be used to determine the attitude. Instead, a least-squares fit of the beacon measurements can be used to determine the attitude. The least-squares approach attempts to minimize the difference between the predicted and measured distances



(a) Position Estimate Error.



(b) Over-Determined System Comparison.

Figure 8-5: Position Estimate Error Analysis

between transmitters and receivers. Let N be the number of transmitters and M be the number of receivers. The error function is seen in Eq. (8.12)

$$f(\boldsymbol{\theta}) = \sum_{i=0}^{N}\sum_{j=0}^{M}\left(\mathbf{g}_{ij}^{\top}(\boldsymbol{\theta})\mathbf{g}_{ij}(\boldsymbol{\theta}) - (s \cdot m_{ij})^2\right)^2 \tag{8.12}$$

In order to solve for $\boldsymbol{\theta}$, it is first necessary to compute the gradient of $f(\boldsymbol{\theta})$.

$$\nabla_{\boldsymbol{\theta}} f = 4\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{\partial \mathbf{g}_{ij}}{\partial \boldsymbol{\theta}}\mathbf{g}_{ij}(\theta)\left(\mathbf{g}_{ij}^{\top}(\boldsymbol{\theta})\mathbf{g}_{ij}(\boldsymbol{\theta}) - (s \cdot m_{ij})^2\right) \tag{8.13}$$

The term $\partial \mathbf{g}_{ij}/\partial \boldsymbol{\theta}$ is found using Eq. (8.14).

$$\frac{\partial \mathbf{g}_{ij}}{\partial \boldsymbol{\theta}} = \begin{bmatrix}\frac{\partial R}{\partial \theta_1}\cdot \mathbf{b}_j & \frac{\partial R}{\partial \theta_2}\cdot \mathbf{b}_j & \frac{\partial R}{\partial \theta_3}\cdot \mathbf{b}_j\end{bmatrix}^{\top} \tag{8.14}$$

The terms $\partial R/\partial\theta_1$, $\partial R/\partial\theta_2$, and $\partial R/\partial\theta_3$ of Eq. (8.14) are defined in Eqs. (8.15a) to (8.15c), respectively.

$$\frac{\partial R}{\partial \theta_1} = \begin{bmatrix} 0 & 0 & 0 \\ -s_1 s_2 & -s_1 c_2 c_3 - c_1 s_3 & -c_1 c_3 + c_2 s_1 s_3 \\ c_1 s_2 & -s_2 s_3 + c_1 c_2 c_3 & -c_3 s_1 - c_1 c_2 s_3 \end{bmatrix} \tag{8.15a}$$

$$\frac{\partial R}{\partial \theta_2} = \begin{bmatrix} -s_2 & -c_2 c_3 & 0 \\ -s_1 s_2 & -s_1 c_2 c_3 - c_1 s_3 & -c_1 c_3 + c_2 s_1 s_3 \\ c_1 s_2 & -s_2 s_3 + c_1 c_2 c_3 & -c_3 s_1 - c_1 c_2 s_3 \end{bmatrix} \tag{8.15b}$$

$$\frac{\partial R}{\partial \theta_3} = \begin{bmatrix} 0 & s_2 s_3 & c_3 s_2 \\ 0 & -c_1 c_2 s_3 - s_1 c_3 & s_3 s_1 - c_1 c_2 c_3 \\ 0 & c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 \end{bmatrix} \tag{8.15c}$$

Eq. (8.12) can then be solved using Eq. (8.13) combined with conventional gradient-based nonlinear programming solvers, such as those discussed in Section 6.3.2.

Chapter 9

Conclusion

Spacecraft formation flight has the potential to revolutionize urban planning, farming, hydrology, among other applications [79]. However, there are technical challenges constraining the future of spacecraft formation flight. EMFF seeks to enable spacecraft formation flight by removing propellant from the mission architecture. Before spacecraft engineers are ready to accept EMFF as a true enabling technology, it must first be proven. RINGS will take a step in proving EMFF as a viable technology, yet demonstrating EMFF via RINGS requires advanced control methods.

Dynamic programming has been used in the fields of computer science [80], economics [81], biology [82], and chemistry [83], yet is has yet to be significantly used for spacecraft control. The large storage space required for dynamic programming solutions have made control engineers shy away from dynamic programming in favor of linear control and other optimal control methods. However, the increasing complexity of spacecraft mission architectures, such as EMFF, and their associated control problems may justify the use of dynamic programming for spacecraft.

9.1 Novel Contributions

The major novel contribution of this thesis is presenting a framework for applying dynamic programming to satellite control. Chapter 5 presents a way of analyzing control problems that shows whether dynamic programming is an appropriate control strategy for any embedded system. This paradigm shift is important when considering whether to use dynamic programming for control. Chapter 5 also presents two ways of implementing a dynamic programming controller on an embedded system. This gives some flexibility to the control engineer to adapt the dynamic programming implementation to the constraints of the system. Finally, a method for resolving the difference between physical states and dynamic programming states was presented.

In Chapter 3, novel formulations of the RINGS control problem were developed as a case study for formulating a dynamic programming system. These formulations do not have to be limited to the application of dynamic programming; state reduction is useful for nearly every control methodology. Another method, such as a Gauss Pseudospectral method, could be applied to these same system. In Chapter 4, the cost-to-go for one formulation was examined in order to serve as an example of the type of analysis required when developing the cost-to-go for a spacecraft.

In Chapter 6, a novel mass property identification method that uses nonlinear programming was presented. This method was presented because of the importance of having a good spacecraft model, since the performance of the dynamic programming controller is dependent on the model accuracy. The method presented in Chapter 6 was applied in Chapter 7 to three different spacecraft systems, with the results of each presented. In Chapter 8, a novel, model-free method for determining the position and attitude of a system using line-of-sight measurement devices was presented. This method is intended to be used for model validation, which is important for dynamic programming.

9.2 Future Work

The biggest step forward for dynamic programming for spacecraft is actually implementing a dynamic programming controller on an actual satellite. This will show show the feasibility of a dynamic programming controller. This will also give flight heritage to the new control scheme. The most obvious opportunity for the application of dynamic programming is the RINGS system. The axial cost-to-go is already developed and could be implemented. The cost-to-go for other formulations still need to be developed.

As for mass property identification, there still remains some theoretical work to be done. The convexity of the error function has yet to be proven. Also, the lack of uniqueness of a global minima for certain data sets needs to be further addressed.

The model-free state estimation discussed in Chapter 8 has significant room for additional work. Another method that could be implemented is a Kalman filter that determines both the position and the scale factor. This is different then the current SPHERES estimator that has position and attitude coupled in the estimator [84]. While this method would be model based, it would have to model the acceleration of the system as a random process. This may not be as accurate as an estimator that includes the model of the actuators, but an estimator-free model is still useful for model validation.

Another step for Chapter 8 will be validating the system. This would involve implementing it alongside an already validated estimator and comparing the results. This could easily be performed on the SPHERES testbed.

9.3 Concluding Remarks

Dynamic programming is a potentially viable control option that has yet to become prevalent in the spacecraft control field. Although it has only been applied to a limited scope in this thesis, the methodology presented here can potentially be applied to a wide variety of control problems for spacecraft. This thesis lays the groundwork for dynamic programming as a option for spacecraft control engineers. Given the dependence of the performance of dynamic programming results on the model used to derive them, model validation is important when using dynamic programming. This thesis presents methods for characterizing the mass properties as well as providing state estimates for actuator validation.

Bibliography

[1] M. Neave and R. Sedwick, “Dynamic and thermal control of an electromagnetic formation flight testbed,” Master’s thesis, Massachusetts Institute of Technology, 2005.

[2] R. Sedwick. (2013). Resonant inductive near-field generation system (rings), [Online]. Available: `http://www.energy.umd.edu/projects/smartgrid-08`.

[3] J. J. Sellers, *Understanding Space*, D. Krkpatrick, Ed. McGraw-Hill, 2005.

[4] J. Leitner, “Formation flying - the future of remote sensing from space,” in *Symposium on Space Flight Dynamics*, 2004.

[5] S. DŠAmico, O. Montenbruck, C. Arbinger, and H. Fiedler, “Formation flying concept for close remote sensing satellites,” in *15th AAS/AIAA Space Flight Mechanics Conference*, 2005.

[6] O. Brown and P. Eremenko, “Fractionated space architectures: a vision for responsive space,” Defence Advanced Research Projects Agency, Tech. Rep., 2006.

[7] J. Everist, K. Mogharei, H. Suri, N. Ranasinghe, B. Khoschnevis, P. Will, and W.-M. Shen, “A system for in-space assembly,” in *Procceedings of 2004 IEEE/RSJ Interntation Conference on Intelligent Robots and Systems*, 2004.

[8] A. N. Parmar, M. Arnaud, X. Barcons, J. M. Bleeker, G. Hasinger, H. Inoue, G. Palumbo, and M. J. Turner, “Science with xeus: the x-ray evolving universe spectroscopy mission,” in *Astronomical Telescopes and Instrumentation*, International Society for Optics and Photonics, 2004, pp. 388–393.

[9] R Licata, M Parisch, I. Ruiz Urien, M De Bartolomei, G Grisoni, and F Didot, “Robotic assembly of large space structures: application to xeus,” in *7th ESA Workshop on. Advanced Space Technologies for Robotics and Automation*, 2002.

[10] W.-M. Shen, P. Will, and B. Khoschnevis, “Self-assembly in space via self-reconfigurable robots,” *Robotics and Automation*, vol. 2, pp. 2516–2521, 2003.

[11] P. Staritz, S. Skaff, C. Urmson, and W. Whittaker, “Skyworker: a robot for assembly, inspection and maintenance of large scale orbital facilities,” in *IEEE International Conference on Robotics & Automation*, 2001, pp. 4180–4185.

[12] J. C. Mankins, “Sps-alpha: the first practical solar power satellite via arbitrarily large phased array,” NASA Innovative Advanced Concepts, Tech. Rep., 2012.

[13] L. B. King, G. G. Parker, S. Deshmukh, and J.-H. Chong, "Study of interspacecraft coulomb forces and implications for formation flying," *Journal of Propulsion and Power*, vol. 19, no. 3, pp. 497–505, 2003.

[14] Y. K. Bae, "A contamination-free ultrahigh precision formation flying method for micro-, nano-, and pico-satellites with nanometer accuracy," in *AIP Conference Proceedings*, vol. 813, 2006, p. 1213.

[15] E. M. Kong, D. W. Kwon, S. A. Schweighart, L. M. Elias, R. J. Sedwick, and D. W. Miller, "Electromagnetic formation flight for multisatellite arrays," *Journal of Spacecraft and Rockets*, vol. 41, no. 4, pp. 659–666, 2004.

[16] D. R. Coulter, "Nasa's terrestrial planet finder mission: the search for habitable planets," in *Earths: DARWIN/TPF and the Search for Extrasolar Terrestrial Planets*, vol. 539, 2003, pp. 47–54.

[17] K. A. Polzin, E. Y. Choueiri, P. Gurfil, and N. J. Kasdin, "Plasma propulsion options for multiple terrestrial planet finder architectures," *Journal of spacecraft and rockets*, vol. 39, no. 3, pp. 347–356, 2002.

[18] U. Ahsun, L. Rodgers, and D. W. Miller, "Comparison between structurally connected propellant formation flying and electromagnetic formation flying spacecraft configurations for gen-x mission," in *Optics & Photonics 2005*, International Society for Optics and Photonics, 2005, pp. 58990M–58990M.

[19] D. W. Kwon and R. J. Sedwick, "Cryogenic heat pipe for cooling high temperature superconductors," *Cryogenics*, vol. 49, no. 9, pp. 514–523, 2009.

[20] D. W. Kwon and R Sedwick, "Cryogenic heat pipe for cooling high temperature superconductors with application to electromagnetic formation flight satellites," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2009.

[21] D. W. Kwon, "Electromagnetic formation flight system design," in *Small Satellites for Earth Observation*, Springer, 2008, pp. 221–229.

[22] D. M. Fleetwood, P. S. Winokur, and P. E. Dodd, "An overview of radiation effects on electronics in the space telecommunications environment," *Microelectronics Reliability*, vol. 40, no. 1, pp. 17–26, 2000.

[23] L. Townsend, "Overview of active methods for shielding spacecraft from energetic space radiation," *Physica Medica*, vol. 17, pp. 84–85, 2001.

[24] D. Miller, A Saenz-Otero, J Wertz, A Chen, G Berkowski, C Brodel, S Carlson, D Carpenter, S Chen, S Cheng, *et al.*, "Spheres: a testbed for long duration satellite formation flying in micro-gravity conditions," in *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, Clearwater, Florida, January, 2000, pp. 167–179.

[25] J. Enright, M. Hilstad, A. Saenz-Otero, and D. Miller, "The spheres guest scientist program: collaborative science on the iss," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, IEEE, vol. 1, 2004.

[26] E. M. Kong, A. Saenz-Otero, S. Nolet, D. S. Berkovitz, D. W. Miller, and S. W. Sell, “Spheres as a formation flight algorithm development and validation testbed: current progress and beyond,” in *2nd International Symposium on Formation Flying Missions and Technologies*, 2004, pp. 14–16.

[27] A. J. Buck, “Path planning and position control and of an underactuated electromagnetic formation flight satellite system in the near field,” Master’s thesis, Massachusetts Institute of Technology, 2013.

[28] M. J. Kochenderfer and J. Chryssanthacopoulos, “Robust airborne collision avoidance through dynamic programming,” *Massachusetts Institute of Technology Lincoln Laboratory. Project Report ATC-371*, 2011.

[29] R. Enns and J. Si, “Helicopter trimming and tracking control using direct neural dynamic programming,” *Neural Networks, IEEE Transactions on*, vol. 14, no. 4, pp. 929–939, 2003.

[30] ——, “Apache helicopter stabilization using neural dynamic programming,” *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 19–25, 2002.

[31] ——, “Helicopter tracking control using direct neural dynamic programming,” in *Neural Networks, 2001. Proceedings. IJCNN’01. International Joint Conference on*, IEEE, vol. 2, 2001, pp. 1019–1024.

[32] T. Hashimoto, S.-i. Sakai, K. Ninomiya, K. Maeda, and T. Saitoh, “Formation flight control using super-conducting magnets,” in *International Symposium on Formation Flying Missions and Technologies, Centre National d’Etudes Spatiales, Touluse Space Center, France*, 2002.

[33] T. Hashimoto, S. Sakai, K. Ninomiya, K. Maeda, and T. Saitoh, *Formation Flight using Super-Conducting Magnets*, F. Toulouse, Ed. International Symposium of Formation Flying Missions and Technologies, 2002.

[34] D. W. Kwon, “Electromagnetic formation flight of satellite arrays,” Master’s thesis, Massachusetts Institute of Technology, 2005.

[35] E. M.-C. Kong, “Spacecraft formation flight exploiting potential fields,” PhD thesis, Massachusetts Institute of Technology, 2002.

[36] D. W. Miller, R. J. Sedwick, E. M. Kong, and S. Schweighart, “Electromagnetic formation flight for sparse aperture telescopes,” in *Aerospace Conference Proceedings, 2002. IEEE*, IEEE, vol. 2, 2002, pp. 2–729.

[37] U. Ehsun, “Using electromagnetic formation flying for remote sensing applications,” in *Advances in Space Technologies, 2008. ICAST 2008. 2nd International Conference on*, IEEE, 2008, pp. 118–123.

[38] D. W. Kwon, “Propellantless formation flight applications using electromagnetic satellite formations,” *Acta Astronautica*, vol. 67, no. 9, pp. 1189–1201, 2010.

[39] S. A. Schweighart and R. J. Sedwick, “Propellantless formation flight operations using electromagnetic formation flight,” in *SpaceOps Conference*, 2006.

[40] L. M. Elias, “Dynamics of multi-body space interferometers including reaction wheel gyroscopic stiffening effects,” PhD thesis, Massachusetts Institute of Technology, 2004.

[41] S. Schweighart and R. Sedwick, “Electromagnetic formation flight dipole solution planning,” PhD thesis, Massachusetts Institute of Technology, 2005.

[42] S. A. Schweighart and R. J. Sedwick, “Dynamics of an electromagnetically flown formation of spacecraft within the earth’s magnetic field,” in *Optical Science and Technology, SPIE’s 48th Annual Meeting*, International Society for Optics and Photonics, 2004, pp. 86–97.

[43] S. S. P. Development, “Emfforce design appendix,” Massachusets Institute of Technology Department of Aeronautics and Astronautics, Tech. Rep., 2003.

[44] D. W. Kwon, R. J. Sedwick, S.-I. Lee, and J. Ramirez-Riberos, “Electromagnetic formation flight testbed using superconducting coils,” *Journal of Spacecraft and Rockets*, vol. 48, no. 1, pp. 124–134, 2011.

[45] A. Sakaguchi, “Micro-electromagnetic formation flight of satellite systems,” Master’s thesis, Massachusetts Institute of Technology, 2007.

[46] U. Ahsun, “Dynamics and control of electromagnetic satellite formations,” PhD thesis, Massachusetts Institute of Technology, 2007.

[47] U. Ahsun and D. W. Miller, “Dynamics and control of electromagnetic satellite formations,” in *American Control Conference, 2006*, IEEE, 2006, 6–pp.

[48] D. W. Miller, U. Ahsun, and J. L. Ramirez-Riberos, “Control of electromagnetic satellite formations in near-earth orbits,” *Journal of guidance, control, and dynamics*, vol. 33, no. 6, pp. 1883–1891, 2010.

[49] U. Ahsun, “Dynamics and control of electromagnetic satellite formations in low earth orbits,” in *AIAA Guidance, Navigation, and Control Conference*, 2006.

[50] J. L. Ramirez Riberos, “New decentralized algorithms for spacecraft formation control based on a cyclic approach,” PhD thesis, Massachusetts Institute of Technology, 2010.

[51] R. J. Sedwick and S. A. Schweighart, “Propellantless spin-up of tethered or electromagnetically coupled sparse apertures,” in *Astronomical Telescopes and Instrumentation*, International Society for Optics and Photonics, 2002, pp. 193–204.

[52] S. A. Schweighart and R. J. Sedwick, “Explicit dipole trajectory solution for electromagnetically controlled spacecraft clusters,” *Journal of guidance, control, and dynamics*, vol. 33, no. 4, pp. 1225–1235, 2010.

[53] W.-w. Cai, L.-p. Yang, Y.-w. Zhu, and Y.-w. Zhang, “Optimal satellite formation reconfiguration actuated by inter-satellite electromagnetic forces,” *Acta Astronautica*, 2013.

[54] R. Wawrzaszek and M. Banaszkiewicz, “Control and reconfiguration of satellite formations by electromagnetic forces,” *Proceedings of 2nd Microwave & Radar Week in Poland, STW-2006, Krakow, Poland*, 2007.

[55] R Wawrzaszek and M Banaszkiewicz, "Dynamics of 2 and 3 satellite formations controlled by electromagnetic forces," in *36th COSPAR Scientific Assembly*, vol. 36, 2006, p. 3028.

[56] Y.-w. Zhang, L.-p. Yang, Y.-w. Zhu, H. Huang, and W.-w. Cai, "Nonlinear 6-dof control of spacecraft docking with inter-satellite electromagnetic force," *Acta Astronautica*, vol. 77, pp. 97–108, 2012.

[57] Y.-W. Zhang, L.-P. Yang, Y.-W. Zhu, X.-H. Ren, and H. Huang, "Self-docking capability and control strategy of electromagnetic docking technology," *Acta Astronautica*, vol. 69, no. 11, pp. 1073–1081, 2011.

[58] G. Zeng and M. Hu, "Finite-time control for electromagnetic satellite formations," *Acta Astronautica*, vol. 74, pp. 120–130, 2012.

[59] R. Bellman, *Dynamic programming and modern control theory*, R. E. Kalaba, Ed. Academic Press, 1965.

[60] D. P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, 2005, vol. 2.

[61] G. J. Eslinger and A. Saenz-Otero, "Electromagnetic formation flight control using dynamic programming," in *American Astronautical Society Rocky Mountain Chapter, 2013. Proceedings.*.

[62] D. M. Asmar and G. J. Eslinger, "Nonlinear programming approach to filter tuning," 2012.

[63] I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals of Operations Research*, vol. 134, no. 1, pp. 215–238, 2005.

[64] R. Freund, "Analysis of convex sets and functions," Massachusetts Institute of Technology, Tech. Rep., 2012.

[65] S. Tanygin and T. Williams, "Mass property estimation using coasting maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 20, pp. 625–632, 1997.

[66] E. Wilson, C. Lages, and R. Mah, "On-line, gryo-based, mass-property identification for thruster-controlled spacecraft using recursive least squares," 2002.

[67] E. Bergmann, B. Walker, and D. Levy, "Mass property estimation for control of asymmetrical satellites," *Journal of Guidance, Control, and Dynamics*, vol. 10, pp. 483–491, 1987.

[68] R. F. Richfield, B. K. Walker, and E. V. Bergmann; "Input selection for a second-order mass property estimator," *Journal of Guidance, Control, and Dynamics*, vol. 11, pp. 207–212, 1988.

[69] J Bergmann E; Dzielski, "Spacecraft mass property identification with torque-generating control," *Journal of Guidance, Control, and Dynamics*, vol. 13, pp. 99–103, 1990.

[70] R. Fletcher and M. J. Powell, “A rapidly convergent descent method for minimization,” *The Computer Journal*, vol. 6, no. 2, pp. 163–168, 1963.

[71] C. G. Broyden, “A class of methods for solving nonlinear simultaneous equations,” *Mathematics of computation*, vol. 19, no. 92, pp. 577–593, 1965.

[72] C. Schedlinsk and M. Link, “A survey of current inertia parameter identification methods,” *Mechanical Systems and Signal Processing*, vol. 15, pp. 189–211, 2001.

[73] D. Berkovitz, “System characterization and online mass property identification of the spheres formation flight testbed,” Master’s thesis, Massachusetts Institute of Technology, 2007.

[74] A. Chen, “Propulsion system characterization for the spheres formation flight and docking testbed,” Master’s thesis, Massachusetts Institute of Technology, 2002.

[75] NASA. (2013). Synchronized position, hold, engage, reorient, experimental satellites - vertigo (spheres-vertigo), [Online]. Available: `http://www.nasa.gov/mission_pages/station/research/experiments/869.html`.

[76] J. L. Crassidis, R. Alonso, and J. L. Junkins, “Optimal attitude and position determination from line-of-sight measurements.,” *Journal of Astronautical Sciences*, vol. 48, no. 2, pp. 391–408, 2000.

[77] M. O. Hilstad, J. P. Enright, and A. G. Richards, “The spheres guest scientist program,” *Space Systems Laboratory, Massachusetts Institute of Technology*, 2003.

[78] W. C. Rheinboldt, *Methods for Solving Systems of Nonlinear Equations.* Society for Industrial and Applied Mathematics, 1987, vol. 70.

[79] M. D. Graziano, “Overview of distributed missions,” *Distributed Space Missions for Earth System Monitoring*, vol. 31, p. 375, 2012.

[80] J. Y. Zien, *Hierarchical string matching using multi-path dynamic programming*, US Patent 6,556,984, 2003.

[81] J. Rust, “Numerical dynamic programming in economics,” *Handbook of computational economics*, vol. 1, pp. 619–729, 1996.

[82] E. Rivas and S. R. Eddy, “A dynamic programming algorithm for rna structure prediction including pseudoknots,” *Journal of molecular biology*, vol. 285, no. 5, pp. 2053–2068, 1999.

[83] D. Bylund, R. Danielsson, G. Malmquist, and K. E. Markides, “Chromatographic alignment by warping and dynamic programming as a pre-processing tool for parafac modelling of liquid chromatography–mass spectrometry data,” *Journal of Chromatography A*, vol. 961, no. 2, pp. 237–244, 2002.

[84] S. Nolet, “Development of a guidance, navigation and control architecture and validation process enabling autonomous docking to a tumbling satellite,” PhD thesis, Massachusetts Institute of Technology, 2007.