The Dissertation Committee for Stuart Andrew Stanton
certifies that this is the approved version of the following dissertation:

Finite Set Control Transcription for Optimal Control Applications

Committee:

Belinda G. Marchand, Supervisor

David G. Hull

Maruthi R. Akella

Cesar A. Ocampo

Christopher N. D'Souza

Finite Set Control Transcription for Optimal Control Applications

by

Stuart Andrew Stanton, B.S., S.M.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2009

For my wife.

Acknowledgments

It is a unique blessing to have had the opportunity to study at the University of Texas at Austin for the last three years. I can't think of a better Air Force assignment. I thank God for all of the people that He has placed in my path to get me here and to allow me to succeed while I've been here. This accomplishment is the result of His strength and guidance.

I am extremely grateful to have been able to study under the world-class faculty at UT. I have learned so much from the department's professors, both inside and outside of the classroom. A special thank you must go to my advisor, Belinda Marchand, for guiding my research. It has been a truly rewarding experience to learn and discover under her leadership. I am also thankful to the members of my committee: David Hull, Maruthi Akella, Cesar Ocampo, and Chris D'Souza. Thank you for teaching me so much in the classroom, for your time outside of the classroom, and for your support as this research progressed.

Of course, I wouldn't have been able to even come to UT without the help of so many of my mentors within the Air Force community. My first mentor, Brig Gen Mike DeLorenzo, not only opened the door for me to pursue this PhD, but also (and perhaps more importantly) planted the Astronautics seed when I was 17 years old to pursue an Air Force career in Space. Col Jack Anthony, Lt Col Jerry Sellers, and Lt Col Tim Sejba have also been instrumental in shaping my career. It has been an honor to have been taken under their wings over the last 13 years. I can't express enough my appreciation for their efforts and the opportunities they have given me.

Most importantly, I must thank my family, as they are the ones that have to put up with me on a daily basis. My wife deserves all of the credit for my successes. Her support is unfailing, and her patience is unending. She continues to teach me the lessons that you can't learn in school, such as, how to be nice to people (I'm still learning!). I thank God for bringing her into my life and for convincing her to marry me. Also, I must thank my parents for their support from a distance as I continue my journey through life.

Finite Set Control Transcription for Optimal Control Applications

Stuart Andrew Stanton, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Belinda G. Marchand

An enhanced method in optimization rooted in direct collocation is formulated to treat the finite set optimal control problem. This is motivated by applications in which a hybrid dynamical system is subject to ordinary differential continuity constraints, but control variables are contained within finite spaces. Resulting solutions display control discontinuities as variables switch between one feasible value to another. Solutions derived are characterized as optimal switching schedules between feasible control values. The methodology allows control switches to be determined over a continuous spectrum, overcoming many of the limitations associated with discretized solutions. Implementation details are presented and several applications demonstrate the method's utility and capability. Simple applications highlight the effectiveness of the methodology, while complicated dynamic systems showcase its relevance. A key example considers the challenges associated with libration point formations. Extensions are proposed for broader classes of hybrid systems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Prevalent in many engineering fields are systems composed of interdependent continuous and discrete components. A physical system that describes an object's position or temperature, for example, necessarily involves continuous variables, using differential equations to express their time-varying characteristics. However, what if a digital computer or human decision-making were involved in the process? At some level, some variables are reduced to a finite set of feasible values (such as '0' and '1' for the digital computer). In some cases, their values may be controlled; otherwise, relational equations may express their values as a function of the other continuous or discrete variables.

Systems that incorporate both continuous and discrete dynamical elements are termed *hybrid control systems.*[1] These systems have gained the interest of researchers due to the challenging aspects of their dual structure. Although strictly continuous and strictly discrete control methods exist, many of these techniques are not designed to address problems involving hybrid systems, particularly regarding stability and optimality. The hybrid control problem is the focus of this investigation; specifically, a new method is developed for the determination of the continuous and discrete control variables that affect a hybrid system. In the present formulation, the objective is to determine the optimal values for discrete control variables constrained to a finite set. Dependent continuous variables are included, and extensions for independent continuous variables are also considered.

The current study presents a method that effectively treats the finite set control problem. Solutions derived using this method are characterized as optimal switching schedules, representing control histories contained in finite sets of feasible control values. The optimal control problem is transcribed into a parameter optimization problem, and the solution is subsequently determined using an existing Nonlinear Programming (NLP) algorithm, such as SNOPT[2] (hereafter, called *the optimizer*).

The *Finite Set Control Transcription (FSCT) method* is essentially a formulation of the finite set optimal control problem as a parameterized nonlinear programming problem. The structure is unique among other collocation methods, but it is flexible enough to treat a large set of problems. This study is devoted to developing the methodology for generating an appropriate formulation for any applicable hybrid control problem. In addition, the capability and utility of the method are further explored by demonstrating a range of applications. In so doing, the scope of the method is characterized, ideally inspiring additional applications outside those presented here. However, most applications presented in this investigation focus on aerospace systems, as these represent the initial motivation behind the development of the FSCT method.

1.1 Previous Work

Research in the subject of hybrid systems is still fairly new to the control and computer science communities. Although direct references are available as early as 1966,[3] the field has gained far more interest in the last twenty years. Researchers have explored system modeling, structure, stability analysis, optimality, and the associated control methods for hybrid systems.

It is difficult to categorize all of these efforts, as they seem to approach the subject from a number of different perspectives. Applications are apparent in medical diagnostics, psychology, education, economy, management, and sociology, as well as many of the various engineering disciplines.[4] Each field brings its own background, motivations, and terminology, and efforts to unify the subject are important additions to the available literature.[1]

Hybrid systems theory is sometimes employed to accurately model complex dynamical systems. These are sometimes termed *fuzzy systems* because of the ‘fuzzy’ boundaries in ‘if-then’ type relations included in the dynamics. Subsequently, studies in hybrid control have led to rule-based control methods. Other designs use artificial neural networks, genetic algorithms, or combinations thereof for analysis.[4,5] Indeed, an actual biological neural network is an example of such a hybrid system, where neurons fire electric pulses only when its inputs’ sum exceeds a threshold.

System stability is another area of emphasis for hybrid systems. Stability analysis assists the designer in identifying stable control laws for hybrid systems, just as for continuous and discrete systems. In the case of a hybrid system, the focus is often on *switched systems*, where a switching variable is used to indicate the system mode or dynamics vector field governing the system at a given time. It is observed that in most cases, researchers consider a single switching variable when performing analysis. Using different Lyapunov-like functions for each system mode, for example, conditions for stability can be obtained, leading to switching strategies based on the Lyapunov-like functions and/or their time derivatives.[6–8] Lagrange stability is analyzed via iterated function systems theory.[9]

Cases with more than one discrete component are less common in the literature and usually appear for more specialized cases, such as those involving linear systems and/or a limited number of variables.[10,11] Some methods capitalize on similar theory as the single discrete variable case, while others extend to other methodologies. For example, power and water systems design and control may rely on a combination of genetic algorithms and linear programming.[5] Other nonlinear systems employ optimal control strategies involving a generalized Bellman equation, impulse control, and linear programming.[1,12]

Classical optimal control theory is also used to address, at least, the continuous aspects of a problem.[13,14]

The solution process presented in this study relies on numerical optimization techniques rooted in sequential quadratic programming. The associated background is presented later, in Chapter 2. However, the key element of the FSCT method is ultimately the generalized formulation developed during the course of this study.

1.2 General Problem Statement

The hybrid system under consideration for this investigation is governed by the dynamics

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \tag{1.1}$$

where the vector, $\boldsymbol{y} \in \mathbb{R}^{n_y}$, represents continuous state variables,

$$\begin{array}{rcl} \boldsymbol{y} & = & \begin{bmatrix} y_1 & \cdots & y_{n_y} \end{bmatrix}^T, \\ y_i & \in & \mathbb{C}^1, \end{array} \tag{1.2}$$

and $\boldsymbol{u}$ consists of n_u control elements limited to finite values as

$$\begin{array}{rcl} \boldsymbol{u} & = & \begin{bmatrix} u_1 & \cdots & u_{n_u} \end{bmatrix}^T, \\ u_i & \in & \mathbb{U}_i = \{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i}\}. \end{array} \tag{1.3}$$

The function, $\boldsymbol{f}$, describes the continuous variation of the states in terms of time, t, and the present values for each state and control.[1] Each control variable, u_i, can take on m_i possible values which define the set $\mathbb{U}_i$. The total number of control combinations is

$$\bar{m} = \prod_{i=1}^{n_u} m_i.$$

[1]Provided a time-explicit control, $\boldsymbol{u} = \boldsymbol{u}(t)$, and an initial state definition, $\boldsymbol{y}_0$, at some initial time, t_0, it is possible to express the states explicitly with time as $\boldsymbol{y} = \boldsymbol{y}(t)$. Likewise, the state derivatives may be expressed as $\dot{\boldsymbol{y}} = \dot{\boldsymbol{y}}(t)$. For simplicity in Equation 1.1 and throughout this work, the explicit time-dependence in $\boldsymbol{y}$, $\boldsymbol{u}$, and $\dot{\boldsymbol{y}}$ is assumed.

Thus, the control effort, at any given time, is determined by identifying one of the $\bar{m}$ control variable combinations that define $\boldsymbol{u}$.

At first glance, this formulation appears to limit the method to a specific class of hybrid systems: all states are presented as continuous and all controls are discrete. Thus, systems with discrete states or continuous controls are apparently excluded. However, the FSCT method can be tailored to include continuous state and control variables within $\boldsymbol{y}$ and, likewise, discrete states and controls in $\boldsymbol{u}$ to allow for a more general treatment of hybrid systems. The necessary adjustments, however, may be specific to the given system. These considerations are addressed in Chapter 6. It is also observed that many control variables traditionally modeled as continuous may be more accurately described by a combination of continuous dynamic states and discrete controls. This characteristic is demonstrated by example in Chapter 4. Thus, the formulation of Equations 1.1-1.3 is not necessarily restrictive. For continuity and clarity, in this study the term *state* implies a continuous variable, while *control*, a discrete one.

1.3 Applications

Although processes exist in nature that are accurately modeled with only continuously-varying dynamics, it is often the case that some level of decision-making occurs in the process. The decision is made by any number of sources, from natural to man-made technologies, but it is clear that the selection is among a discrete number of options. Thus, a hybrid system results, exhibiting continuously-varying and discretely-chosen components. It is observed in the literature that this often takes on a hierachical structure, where continuous or time-driven dynamics exist at the lower levels and discrete or event-driven dynamics exist at the higher levels.[1,13]

To motivate the following development, it is worthwhile to consider some general application areas where the FSCT method is useful. Certainly, systems involving hysteresis, collisions, or transmissions are hybrid in nature and can benefit from FSCT analysis.[1] Some additional applications are presented here.

Fixed Thrust Aerospace Systems

The genesis of this investigation is attributed to libration point interferometery missions such as TPF[15,16] and MAXIM,[17] which require precision tracking of spacecraft formations while maintaining fixed vehicle orientations. If translational control is limited to fixed thrust actuation in specified thrust directions, a discrete control problem results. In this case, decision variables exist for each translational thruster, and control is limited to two values per variable, indicating 'on' or 'off.' Spacecraft positions and velocities are continuously varying states whose dynamics are directly affected by the finite set controls.

A similarly structured problem considers spacecraft attitude control with fixed magnitude reaction jets. Again, control variables indicate the on-off status for thrusters, and spacecraft orientation and angular velocities are modeled as states with differential dynamics.

Operations Analysis

Operations analysis applications exist for optimizing task scheduling and resource allocation. As an example, consider a resource (such as a person's time) that can be applied to a number of different operations, either in series or parallel. A control variable is assigned to each operation, and its value indicates whether the resource is applied to it or not. The dynamics of each operation may be affected by each control value. In this example, the control variables can only take on a finite set of values, but the system is still continuous in time and in state dynamics.

Constrained Optimal Control Systems as Hybrid Systems

The subject of constrained optimal control is relevant to many engineering fields, from aerospace to industrial applications. Constraints may exist on both state and control variables, as point or path constraints. Often, control constraints represent actuator saturation, and these are treated by bounding control variables with a minimum and maximum value. Generally, control solutions are acceptable as long as the variables remain within the continuous spectrum between the bounds. Depending on the cost to be minimized, an optimal control history may likely exhibit 'bang-bang' control behavior, switching between minimum and maximum values. If this control behavior is assumed a priori, a continuous (but constrained) optimal control problem may be treated as a finite set control problem, where the finite set consists of each control's extremal values.

1.4 FSCT Method Overview

The Finite Set Control Transcription is a formulation of the hybrid optimal control problem as a parameter optimization problem that can be solved using a standard optimizer. The following overview is intended to provide a basic description of the methodology. The optimization techniques employed and a complete development of the methodology are presented in Chapters 2 and 3 to follow. Note that, although the method demonstrated here is rooted in direct collocation, alternative formulations exist that capitalize on the structure of indirect or direct shooting methods.

In the most basic sense, a transcription formulation seeks to convert the optimal control problem defined by an objective,

$$\text{Minimize } \mathcal{J} = \phi(t_0, \boldsymbol{y}_0, t_f, \boldsymbol{y}_f) + \int_{t_0}^{t_f} L(t, \boldsymbol{y}, \boldsymbol{u})\, dt \tag{1.4}$$

subject to Equation 1.1 and

$$\begin{array}{rcl} \mathbf{0} & = & \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0), \\ \mathbf{0} & = & \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f), \\ \mathbf{0} & = & \boldsymbol{\beta}(t, \boldsymbol{y}, \boldsymbol{u}). \end{array} \tag{1.5}$$

into an NLP problem of the form,

$$\text{Minimize } F(\boldsymbol{x}) \tag{1.6}$$

subject to

$$\boldsymbol{c}(\boldsymbol{x}) = \left[\boldsymbol{c}_y^T(\boldsymbol{x})\ \boldsymbol{c}_{\psi_0}^T(\boldsymbol{x})\ \boldsymbol{c}_{\psi_f}^T(\boldsymbol{x})\ \boldsymbol{c}_\beta^T(\boldsymbol{x})\right]^T = \mathbf{0}. \tag{1.7}$$

Ultimately, $\boldsymbol{x}$ must contain the information necessary to express $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$ for $t \in [t_0\ t_f]$. In the resulting NLP problem, an initial guess for $\boldsymbol{x}$ is iterated until arriving at a feasible and locally optimal set of values. Note that each problem has a cost function to minimize as well as constraints for the dynamics, initial and final conditions, and any path constraints imposed on the system. In the above problem definitions, all constraints are presented as equalities, however, extensions certainly exist for inequality constraints, as well. The nature of the transcription formulation dictates both the definition of the parameter vector, $\boldsymbol{x}$, and the number and forms of the constraint functions in $\boldsymbol{c}(\boldsymbol{x})$ in the resulting parameter optimization problem.

Consider the following definition of the parameter vector used for an optimization with the FSCT method.

$$\boldsymbol{x} = [\cdots\ y_{i,j,k}\ \cdots\ \cdots\ \Delta t_{i,k}\ \cdots\ t_0\ t_f]^T \tag{1.8}$$

The vector, $\boldsymbol{x}$, contains parameters that represent states, $y_{i,j,k}$, and times $\Delta t_{i,k}$, t_0, and t_f. One of the key features of this parameterization is that control variables are not among the parameters to be optimized. This is unusual: most collocation and direct shooting methods optimize parameters that directly represent control variables. However, in this case, a unique parameterization is necessary since the controls are discrete variables, while the elements of $\boldsymbol{x}$, by the nature of nonlinear programming, are necessarily treated as continuous variables (although perhaps bounded and subject to constraints). However, in this case, a control history is completely defined by the time elements in the parameter vector.

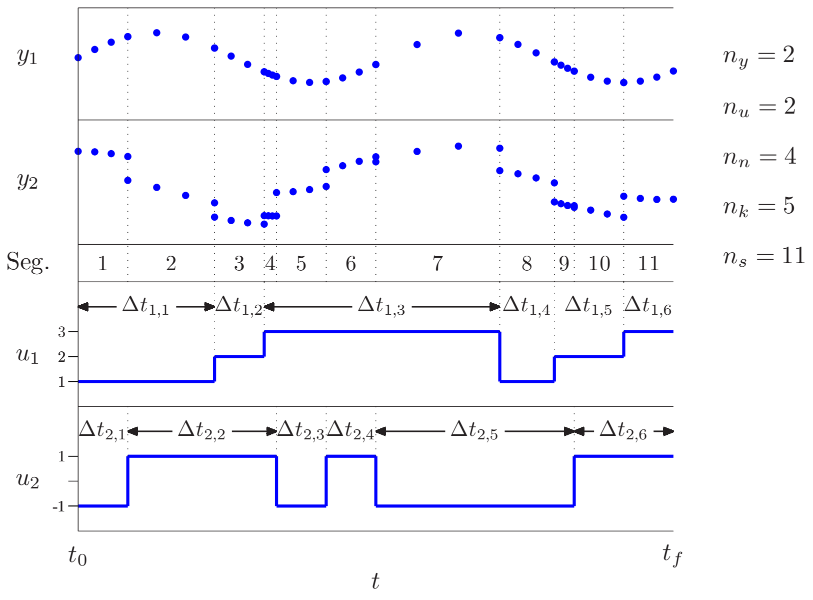
Let the trajectory defined from initial time, t_0, to final time, t_f, be broken up into n_s segments. The interior separation times between segments are termed *knots*. These represent instances of time when the discrete control variables switch from one feasible value to another. Suppose each control variable is allowed n_k switches between t_0 and t_f. The result is that $n_s = n_u n_k + 1$, and each control is held constant over each segment.

Define n_n as the number of nodes per segment. A *node* is a point in time at which the values of the state variables are contained within the parameter vector. Specifically, element $y_{i,j,k}$ in Equation 1.8 represents the i^{th} state at the j^{th} node of the k^{th} segment. Then, $\boldsymbol{x}$ contains $n_y n_n n_s$ elements pertaining to all of the states at each node. These state values are used directly in the cost and constraint Equations 1.6 and 1.7.

The elements $\Delta t_{i,k}$ in $\boldsymbol{x}$ indicate the elapsed time between two control switches for a given control variable. Specifically, $\Delta t_{i,k}$ indicates the amount of time that passes between the control switches at the $(k-1)^{\text{th}}$ and k^{th} knots for the i^{th} control variable.

The values for each u_i are *pre-specified* between each switching point. Thus, $u^*_{i,k}$ indicates the pre-specified value of the i^{th} control variable before the k^{th} knot. With a discrete number of feasible values, it is possible to set n_k large enough such that each possible control value is designated as the actual control value for some duration. During the optimization, the values of $\Delta t_{i,k}$ are determined, indicating the amount of time (possibly, zero) that each control value is maintained.

The transcription definition is best interpreted with a visualization, such as Figure 1.1. In this conceptualization, consider the hybrid control problem with $n_y = 2$ states and



y_1
y_2
Seg.
1 2 3 4 5 6 7 8 9 10 11
$\Delta t_{1,1}$ $\Delta t_{1,2}$ $\Delta t_{1,3}$ $\Delta t_{1,4}$ $\Delta t_{1,5}$ $\Delta t_{1,6}$
u_1
3
2
1
$\Delta t_{2,1}$ $\Delta t_{2,2}$ $\Delta t_{2,3}$ $\Delta t_{2,4}$ $\Delta t_{2,5}$ $\Delta t_{2,6}$
u_2
1
-1
t_0
t_f
t
$n_y = 2$
$n_u = 2$
$n_n = 4$
$n_k = 5$
$n_s = 11$

Figure 1.1: The Parameters of $\boldsymbol{x}$

$n_u = 2$ controls, where

$$\begin{aligned} \mathbb{U}_1 &= \{1, 2, 3\}, \\ \mathbb{U}_2 &= \{-1, 1\}. \end{aligned}$$

Next, assume the transcription is selected such that $n_n = 4$ nodes per segment and $n_k = 5$ switching points per control variable. Thus, the number of segments is $n_s = (2)(5)+1 = 11$ segments.

It is apparent from Figure 1.1 that each control variable may take up to $n_k + 1 = 6$ different values over the trajectory duration. Arbitrarily, the control values are pre-specified so that each control variable systematically switches between the feasible values for that variable. Note that some feasible control values may not be members of the optimal solution.

However, through iteration, the time durations between switching points are optimized. If one of the pre-specified control values is unnecessary or non-optimal, then the value of the respective time duration is reduced to zero.

Figure 1.1 further illustrates that the node distribution is not necessarily uniform over the interval $[t_0 \; t_f]$. The duration of each segment is dictated by the current values of $\Delta t_{i,k}$. The $n_n = 4$ nodes per segment are evenly distributed over a segment, but for shorter segments, this means a closer spacing between nodes. Thus, the state values contained in $\boldsymbol{x}$ may pertain to dense or sparse regions, depending on the time parameters in $\boldsymbol{x}$.

It is also important to note that two nodes are associated with a given knot: the terminal node from the preceding segment and the initial node from the following segment. Therefore, in this parameterization, two sets of state values are contained in $\boldsymbol{x}$ for the times at each knot. For a feasible solution, continuous state variables exhibit identical values at simultaneous nodes. Constraints in $\boldsymbol{c}(\boldsymbol{x})$ are included to enforce continuity across segments. Of course, these constraints are not always satisfied on intermediate iterations of the solution process. For example, in Figure 1.1, the continuity constraints for y_2 are not all met. Subsequently, this $\boldsymbol{x}$ does not represent a feasible solution. During the FSCT optimization process, elements of $\boldsymbol{x}$ are updated to ensure that, upon completion, the continuity constraints are satisfied.

Additional constraints are included in $\boldsymbol{c}(\boldsymbol{x})$ to ensure that

$$0 = t_f - t_0 - \sum_{k=1}^{n_k+1} \Delta t_{i,k}, \; i = 1, \ldots, n_u.$$

Also, at all times, $\Delta t_{i,k} \geq 0$ so that there are no negative time intervals.

By pre-specifying the control values, a collocation transcription results in which control switching times are optimized to indicate an optimal control history over all of the feasible control values. Multiple control variables are easily managed and treated independently. The control variables for a given segment subsequently affect the hybrid system dynamics, and they are included in appropriate constraint equations for that segment. As the optimizer searches for a feasible and locally optimal set of parameters, the state values are modified at each node so that, upon completion, the state and control histories represent a matching, feasible trajectory.

The total number of feasible values for a control variable, m_i, significantly affects the choice of n_k, the number of switching points allowed over the trajectory. Clearly, when $n_k \gg \max(m_i)$, it is possible to pre-specify each control value over several time durations, allowing more flexibility in the resulting NLP problem and a greater likelihood to converge on a small local minimum. However, as n_k gets larger, the sizes of $\boldsymbol{x}$ and $\boldsymbol{c}(\boldsymbol{x})$ also increase, a feature that may complicate or slow down the optimization process. This characteristic indicates the primary limitation of the FSCT method. In order to perform an optimization, a user must specify n_k, thus limiting the number of control switches to some maximum value.

1.5 Organization

This effort is organized as follows.

- **Chapter 2: Numerical Optimization Methods for Continuous Systems** The theory of optimal control and numerical optimization for continuous systems is reviewed to establish a relevant background. Some common numerical methods for solving optimal control problems are surveyed to justify the approach selected for this investigation.

- **Chapter 3: Transcription Formulations for the Finite Set Control Problem** The FSCT method is presented in detail including arguments that explain its success. The development starts from the context of continuous systems, introducing a traditional collocation transcription method first. Next, modifications and enhancements are incrementally applied to the method to treat the discrete characteristics of finite set control. Subsequently, many of the unique implementation issues are addressed, focused primarily on the considerations associated with *switching dependencies.*

- **Chapter 4: General Applications** A series of applications is presented that is designed to demonstrate the capability of the FSCT method. Results are compared with those produced using alternative hybrid control methods to articulate particular advantages or ways in which multiple methods can be used in tandem.
  - The stability of a switched linear system is considered. In each control mode, the system is stable, but certain switching structures result in instability. The FSCT method is contrasted with a technique involving *Multiple Lypunov Functions.*
  - Minimum-time and minimum-acceleration optimizations for a simple system in two dimensions is presented. An FSCT optimal solution is implemented by a real-time *Model Predictive Control* law.
  - Attitude control for a small spacecraft using inexpensive cold-gas thruster technology is explored. The FSCT method is used for trajectory tracking with fixed thrust and variable thrust dynamic schemes.
- **Chapter 5: The Libration Point Formation** Due to the extreme sensitivity of the dynamical regime near the Sun-Earth/Moon libration points, unconstrained control laws in this regime can require thrust levels *below* the limits of the technology presently available, according to several studies focused on reducing lower performance bounds on thrusters.[18–22] The combination of these conditions constrain actuators to either thrust at their lower limits or not at all. This results in control solutions characterized by a switching schedule that can be optimized. A survey is conducted to determine the feasibility of precision formation keeping during interferometry mission phases for a sample three-spacecraft formation.

- **Chapter 6: Extended Applications** The limitations imposed by the general problem statement of Section 1.2 are addressed. Processes are described for transforming different classes of hybrid control problems to conform to the FSCT formulation. Continuous controls, discrete states, and partial differential state equations are considered.

- **Chapter 7: Conclusions** The results of the investigation are summarized. Suggestions are provided for areas in which future work may expand the results obtained in the current effort.

Additional references are included in the form of an appendix.

- **Appendix A: Model Predictive Control for the Hybrid System** A more complete development of a traditional model predictive controller is presented to demonstrate how the concepts can be applied to the finite set control problem.

- **Appendix B: Previous Work Towards Libration Point Formations** A summary of other investigations into the feasibility of libration point formations provides context for Chapter 5.

Chapter 2

Numerical Optimization Methods for Continuous Systems

Although the focus of this investigation is on hybrid control problems, the methodology presented here results from extending the existing methodologies for solving continuous optimal control problems. This comes from the observation that many hybrid systems can be formulated as constrained continuous systems. Thus, the fundamental background for this research is the theory and numerical methods of continuous optimal control. This chapter, then, focuses strictly on continuous systems. Extensions to hybrid control are presented in the next chapter.

The methodology employed in this study is rooted in direct collocation, one of the many numerical optimization methods of transcribing an optimal control problem into a parameter optimization problem. This chapter develops the general methods employed to arrive at the solutions to optimization problems. This provides context for the collocation fundamentals, with the intent of justifying its employment over some of its more popular competitors. To that end, this chapter is broken into four parts. A brief review of the analytical theory of optimal control is first presented, as it is necessary for the rest of the discussion. However, numerical optimization techniques are ultimately necessary due to the generalized complexity of most problems of interest, complexities that often render the analytical approach infeasible. The theory of numerical optimimization is addressed in the second part of this chapter. The third part examines the various methods of transforming the optimal control problem into a form suitable for implementation in a numerical approach. In this section, the major advantages and disadvantages of various optimization methods are compared. Finally, a classical example illustrates the methods available and their salient characteristics. This demonstration leads to the conclusion that direct collocation serves as the best framework for the remainder of this investigation.

2.1 Review of Optimal Control Theory Fundamentals

The history and theory of optimal control is treated in many places, such as Hull,[23] Bryson and Ho,[24] and Kirk.[25] The scope of this work does not include an in-depth development of the theory, so it is assumed that the reader is at least familiar with the subject. However, since the fundamentals of the theory are the foundation of the present analysis, the basic concepts are reviewed briefly here.

Consider a general optimal control problem presented in the form of Bolza.[23] The objective is to determine the control history, $\boldsymbol{u}(t)$, which minimizes the cost function,

$$\mathcal{J} = \phi(t_0, \boldsymbol{y}_0, t_f, \boldsymbol{y}_f) + \int_{t_0}^{t_f} L(t, \boldsymbol{y}, \boldsymbol{u})\, dt, \tag{1.4}$$

where t_0 is the initial time, t_f is the final time, $\boldsymbol{y} \in \mathbb{R}^{n_y}$ denotes the state vector, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ represents the control vector, $\boldsymbol{y}_0 = \boldsymbol{y}(t_0)$, and $\boldsymbol{y}_f = \boldsymbol{y}(t_f)$. The optimal solution must satisfy the vector dynamical constraint,

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \tag{1.1}$$

as well as the constraints,

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0), \\ \mathbf{0} &= \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f), \\ \mathbf{0} &= \boldsymbol{\beta}(t, \boldsymbol{y}, \boldsymbol{u}). \end{aligned} \tag{1.5}$$

Here, $\boldsymbol{\psi}_0 \in \mathbb{R}^{n_{\psi_0}}$ is a vector of initial conditions, $\boldsymbol{\psi}_f \in \mathbb{R}^{n_{\psi_f}}$ denotes the terminal constraints, and $\boldsymbol{\beta} \in \mathbb{R}^{n_\beta}$ represents the path constraints imposed over all $t \in [t_0\ t_f]$. Note, though all the constraints presented here are formulated as equality constraints, that is not a necessary restriction. Extensions are available to accommodate inequality constraints as well. For example, any inequality constraint, $\tilde{c} \leq 0$, can be converted into an equality constraint,

$c = 0$, with the use of a slack variable:[23]

$$c = \tilde{c} + \alpha^2 = 0.$$

In this case, the slack variable, α, is treated as a control parameter. If $\tilde{c}$ is satisfied along the boundary, $\alpha = 0$.

Many variations of the Bolza problem exist. For example, by augmenting the state vector, the problem may be manipulated into Mayer form, where the cost is only a function of ϕ ($L = 0$). In addition, it is possible to include constant parameters that must also be determined in the solution. However, the form presented in Equation 1.4 sufficiently envelopes the problems addressed in this work, and thus it is employed from here on.

2.1.1 First-Order Optimality Conditions

The conditions of optimality for the problem described above are derived via the Method of Lagrange. Simply stated, the constraints are adjoined to the cost function with Lagrange multipliers. An extremum is found by setting to zero the differential of the resulting augmented cost function. This describes a point or arc where the gradient of the cost function is normal to the constraints imposed. The augmented cost function, described by $\tilde{\mathcal{J}}$, is defined as

$$\tilde{\mathcal{J}} = G(t_0, \boldsymbol{y}_0, t_f, \boldsymbol{y}_f, \boldsymbol{\nu}_0, \boldsymbol{\nu}_f) + \int_{t_0}^{t_f} \left(H(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) - \boldsymbol{\lambda}^T \dot{\boldsymbol{y}} \right) dt, \tag{2.1}$$

where

$$\begin{aligned} G(t_0, \boldsymbol{y}_0, t_f, \boldsymbol{y}_f, \boldsymbol{\nu}_0, \boldsymbol{\nu}_f) &= \phi(t_0, \boldsymbol{y}_0, t_f, \boldsymbol{y}_f) + \boldsymbol{\nu}_0^T \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) + \boldsymbol{\nu}_f^T \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f), && (2.2) \\ H(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= L(t, \boldsymbol{y}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) + \boldsymbol{\mu}^T \boldsymbol{\beta}(t, \boldsymbol{y}, \boldsymbol{u}), && (2.3) \end{aligned}$$

and $\boldsymbol{\nu}_0$, $\boldsymbol{\nu}_f$, $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$ are *Lagrange multipliers* associated with the initial and final endpoint constraints, differential constraints, and path constraints, respectively. The set $\boldsymbol{\lambda}$, associated with the differential state equations, are also termed the *costates*, as each time varying costate corresponds to a state. Note that the Lagrange multipliers associated with point constraints are constant, but those associated with the time-varying differential and path constraints are time-varying. Along any feasible trajectory (that is, a trajectory where the constraints are satisfied), the cost function and the augmented cost function are identical.

The first differential of the augmented performance index is

$$\begin{aligned} d\tilde{\mathcal{J}} &= G_{t_0}dt_0 + G_{\boldsymbol{y}_0}d\boldsymbol{y}_0 + G_{t_f}dt_f + G_{\boldsymbol{y}_f}d\boldsymbol{y}_f + G_{\boldsymbol{\nu}_0}d\boldsymbol{\nu}_0 + G_{\boldsymbol{\nu}_f}d\boldsymbol{\nu}_f + \left[\left(H - \boldsymbol{\lambda}^T\dot{\boldsymbol{y}}\right)dt\right]_{t_0}^{t_f} \\ &\quad + \int_{t_0}^{t_f}\left(H_{\boldsymbol{y}}\delta\boldsymbol{y} + H_{\boldsymbol{u}}\delta\boldsymbol{u} + H_{\boldsymbol{\lambda}}\delta\boldsymbol{\lambda} + H_{\boldsymbol{\mu}}\delta\boldsymbol{\mu} - \delta\boldsymbol{\lambda}^T\dot{\boldsymbol{y}} - \boldsymbol{\lambda}^T\delta\dot{\boldsymbol{y}}\right)dt. \qquad (2.4) \end{aligned}$$

In Equation 2.4, the subscripts of the form $(\cdot)_x$ are shorthand for the partial derivatives, $\frac{\partial(\cdot)}{\partial x}$. The differential is simplified by observing that

$$\begin{aligned} G_{\boldsymbol{\nu}_0} &= \boldsymbol{\psi}_0^T = \boldsymbol{0}^T, \\ G_{\boldsymbol{\nu}_f} &= \boldsymbol{\psi}_f^T = \boldsymbol{0}^T, \\ H_{\boldsymbol{\lambda}} &= \boldsymbol{f}^T, \\ H_{\boldsymbol{\mu}} &= \boldsymbol{\beta}^T = \boldsymbol{0}^T, \end{aligned}$$

resulting in several cancelations. In addition, integration by parts is performed on the term

$$\int -\boldsymbol{\lambda}^T\delta\dot{\boldsymbol{y}}\,dt = \left[-\boldsymbol{\lambda}^T\delta\boldsymbol{y}\right] + \int \dot{\boldsymbol{\lambda}}^T\delta\boldsymbol{y}\,dt$$

to substitute $\delta\dot{\boldsymbol{y}}$ with $\delta\boldsymbol{y}$ throughout. The resulting form of the differential is,

$$\begin{aligned} d\tilde{J} &= G_{t_0}dt_0 + G_{\boldsymbol{y}_0}d\boldsymbol{y}_0 + G_{t_f}dt_f + G_{\boldsymbol{y}_f}d\boldsymbol{y}_f + \left[\left(H - \boldsymbol{\lambda}^T\dot{\boldsymbol{y}}\right)dt\right]_{t_0}^{t_f} - \left[\boldsymbol{\lambda}^T\delta\boldsymbol{y}\right]_{t_0}^{t_f} \\ &\quad + \int_{t_0}^{t_f}\left(\left(H_{\boldsymbol{y}} + \dot{\boldsymbol{\lambda}}^T\right)\delta\boldsymbol{y} + H_{\boldsymbol{u}}\delta\boldsymbol{u}\right)\,dt \qquad (2.5) \\ &= \left(G_{t_0} - H_0\right)dt_0 + \left(G_{\boldsymbol{y}_0} + \boldsymbol{\lambda}_0^T\right)d\boldsymbol{y}_0 + \left(G_{t_f} + H_f\right)dt_f + \left(G_{\boldsymbol{y}_f} - \boldsymbol{\lambda}_f^T\right)d\boldsymbol{y}_f \\ &\quad + \int_{t_0}^{t_f}\left(\left(H_{\boldsymbol{y}} + \dot{\boldsymbol{\lambda}}^T\right)\delta\boldsymbol{y} + H_{\boldsymbol{u}}\delta\boldsymbol{u}\right)\,dt. \qquad (2.6) \end{aligned}$$

The steps taken between Equations 2.5 and 2.6 involve the relationship between the variation and the partial

$$d\boldsymbol{y} = \delta\boldsymbol{y} + \dot{\boldsymbol{y}}dt.$$

At an extremum, the differential in Equation 2.6 must equal zero. Thus, the coefficients in front of each independent differential or variation must be zero. Because of the path constraint, $\boldsymbol{\beta}$, though, the variations $\delta\boldsymbol{u}$ are not all independent. However, it is possible to choose the values of $\boldsymbol{\mu}$ such that the coefficients in front of the dependent variations of $\boldsymbol{u}$ equal zero. Then, all of the coefficients in Equation 2.6 must be zero, and this leads to the set of first-order differential equations for an extremum, the Euler-Lagrange equations,

$$\begin{array}{rcl} \dot{\boldsymbol{y}} & = & \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \\ \dot{\boldsymbol{\lambda}} & = & -H_{\boldsymbol{y}}^T(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \\ \boldsymbol{0} & = & H_{\boldsymbol{u}}^T(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \end{array} \tag{2.7}$$

and the boundary conditions,

$$\begin{array}{rrr} \boldsymbol{\psi}_0 = \boldsymbol{0}, & \boldsymbol{\psi}_f = \boldsymbol{0}, & \boldsymbol{\beta} = \boldsymbol{0}, \\ H_0 = G_{t_0}, & H_f = -G_{t_f}, & \\ \boldsymbol{\lambda}_0 = -G_{\boldsymbol{y}_0}^T, & \boldsymbol{\lambda}_f = G_{\boldsymbol{y}_f}^T. & \end{array} \tag{2.8}$$

The conditions presented in Equations 2.7 and 2.8 completely describe the optimal solution to the Bolza problem. Unfortunately, these equations are mostly useful in determining analytical solutions to only the simplest of optimization problems. For moderate to complex problems, numerical optimization methods are often the most common approach in the identification of an optimal solution.

For the discussion that follows, assume a distinction between ‘techniques’ and ‘methods’. The term *technique* is used in this context to indicate a process for solving a *function minimization* problem. Note, however, that the optimal control problem is one of *functional minimization*, as the cost function in Equation 1.4 depends on other functions. The term *method*, then, implies the process of converting the functional minimization problem into a function minimization problem. Numerical methods are the focus of Section 2.3 and beyond. First, however, consider the techniques available for optimizing a function.

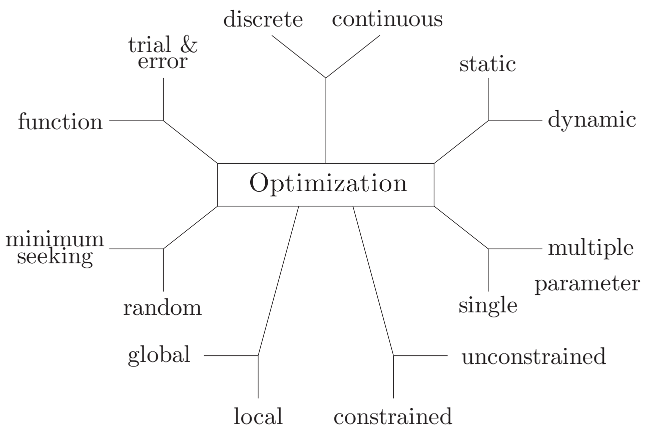


Figure 2.1: Categories of Optimization Algorithms[26]

2.2 Some Numerical Optimization Techniques

The process of identifying a solution to an optimization problem may benefit from a variety of existing techniques. It is outside the scope of this study to examine all of them closely. Even classifying them is a challenge, as there are so many characteristics to consider that identify any technique. Figure 2.2 illustrates one possible classification, where various optimization algorithms are broken into non-mutually exclusive branches.[26] Each of the branches indicates a significant characteristic, and the techniques available to solve each category vary accordingly.

Consider, for example, the limb that branches to either discrete or continuous parameter optimization. In the context of optimal control of hybrid systems, this is perhaps the most important distinction to make among the various techniques. Without techniques that can treat both discrete and continuous parameters in a simple, concise manner, numerical optimization of hybrid systems must begin from either a discrete or continuous technique. The greatest subsequent effort is in determining how to treat the other type of element once a specific branch of techniques is selected (i.e. how to treat discrete variables using a continuous technique, or vice versa).

This investigation, again, begins from the continuous control problem presented in Section 2.1. Consequently, a continuous technique is selected. However, it is important to note that discrete techniques, such as *integer programming*, *combinatorial* algorithms,[27,28] or *dynamic programming*,[29] have all been employed by others to treat the discrete components of hybrid problems.

In addition, it is reasonable to limit consideration to those techniques that minimize tractable functions with multiple static parameters. Assume that the function minimization problem can always be expressed as a cost in the form

$$\mathcal{J} = F(\boldsymbol{x}). \tag{2.9}$$

The parameters, $\boldsymbol{x} \in \mathbb{R}^n$, are constants to be determined, and the solution is a static point in n-dimensional space. These limitations reduce the pool of numerical techniques significantly, leaving two more branches in Figure 2.2 to consider.

One branch distinguishes global techniques from local techniques. For example, an *exhaustive search* is a global technique that requires a sampling of the cost function over the entire domain. To evaluate a sufficient sampling of the solution space in most cases is extremely time consuming and certainly inefficient. *Genetic algorithms*,[26] modeling the natural selection process, have the ability to jump out of local minima in favor of a global minimum. Formulations for continuous or discrete parameters are possible, and genetic algorithms have been used previously on hybrid systems.[4,5]

Alternatively, this analysis favors local techniques, which generally outperform genetic algorithms when the analytic cost function is well-behaved. Local techniques examine the vicinity of a current iterative point to determine how to select the next point. Some common local techniques for unconstrained minimization are listed below.

- *Quadratic programming* algorithms are effective in determining the minimum for convex quadradic cost functions.
- Non-derivative techniques, like the *simplex technique*, seek to determine a 'downhill' direction without the evaluation of derivatives. For example, by examining the values of the cost function at the nearby vertices of an $(n+1)$-gon, an update is made by replacing the highest value vertex with a new point.
- First order derivative techniques require smooth functions (in order to evaluate derivatives) and move a nonoptimal point downhill until it reaches a local minimum. The *gradient* or *steepest descent* technique simply evaluates the slope of the cost function at the current point and moves to the next point along the path where the slope is steepest.
- Local techniques using second order derivatives are quite popular. For example, the *Newton-Raphson*[30] technique applies the properties of a quadratic function to determine an update direction to move to the next point. It is applicable for all smooth nonlinear functions and is guaranteed to reach the minimum of a quadratic function in n iterations.

The efficiency, robustness, and maturity of the derivative techniques motivate their use in solving optimal control problems from here on. They can easily be implemented in the case of unconstrained optimization, and well-developed algorithms are available for their implementation with constraints formulated as the functions,

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{x}) &= \boldsymbol{0}, \\ \tilde{\boldsymbol{c}}(\boldsymbol{x}) &\leq \boldsymbol{0}. \end{aligned}$$

The discussion continues with a review of both first and second derivative techniques, with formulations for addressing constraints in function minimization.

2.2.1 Derivative Techniques

Consider a general unconstrained parameter optimization problem, where Equation 2.9 is the nonlinear cost function to be minimized. The optimal values of $\boldsymbol{x}$ are those than minimize $F(\boldsymbol{x})$. The basic optimization process is to perform an iteration on $\boldsymbol{x}$ until a locally optimal point is determined. Thus, from any arbitrary set of values for $\boldsymbol{x}$, the objective is to determine the update, $\Delta\boldsymbol{x}$, such that

$$F(\boldsymbol{x} + \Delta\boldsymbol{x}) < F(\boldsymbol{x}).$$

Therefore, on each iteration, $\boldsymbol{x}$ is updated and the cost is reduced. On the p^{th} iteration, $\boldsymbol{x}_{p+1} = \boldsymbol{x}_p + \Delta\boldsymbol{x}_p$. The process continues to iterate on $\boldsymbol{x}$ until either $|F(\boldsymbol{x}_{p+1}) - F(\boldsymbol{x}_p)| < \epsilon$ or $\|\boldsymbol{x}_{p+1} - \boldsymbol{x}_p\| = \|\Delta\boldsymbol{x}_p\| < \epsilon$ for some sufficiently small $\epsilon > 0$.

Assuming that $n > 1$, the update, $\Delta\boldsymbol{x} = \alpha\boldsymbol{d}$, consists of two parts: a direction and a magnitude. From the current value of $\boldsymbol{x}$, a candidate search direction, $\boldsymbol{d}$, is determined, and a positive value for α is computed. Thus, the point $\boldsymbol{x}$ is updated, during each iteration, by computing the change in $\boldsymbol{x}$ as a positve step α along the search direction $\boldsymbol{d}$. Techniques are primarily identified by how $\boldsymbol{d}$ is determined. Once $\boldsymbol{d}$ is known, the optimal value of α is determined. The search for the optimal α may benefit from any number of simple one-dimensional search algorithms. Although the various line searches available for this operation are not discussed here, in a general sense $F(\boldsymbol{x} + \alpha\boldsymbol{d})$ is evaluated for different values of α until the minimizing value (to tolerance) is determined.

It is relevant, however, to develop some techniques for determining the update direction, $\boldsymbol{d}$. The Gradient and the Newton-Raphson techniques represent first and second order derivative techniques, respectively, and are both discussed below.

2.2.1.1 Gradient Technique for Unconstrained Problems

The Gradient technique, as its name implies, defines $\Delta\boldsymbol{x}$ according to the gradient of the function to be minimized. It is a first order method, as it is derived from the first order approximation of the Taylor series.

$$F(\boldsymbol{x}+d\boldsymbol{x}) = F(\boldsymbol{x}) + F_{\boldsymbol{x}}(\boldsymbol{x})d\boldsymbol{x} + ... \tag{2.10}$$

Consider a *minor* optimization problem, solved on each iteration to determine $\Delta\boldsymbol{x}$ to maximize improvement. For this technique, the minor problem originates from the linear function defined by truncating Equation 2.10 after the second term.

$$\mathscr{J} = F(\boldsymbol{x}+\Delta\boldsymbol{x}) - F(\boldsymbol{x}) = F_{\boldsymbol{x}}(\boldsymbol{x})\Delta\boldsymbol{x}$$

Note that the greatest improvement on $\boldsymbol{x}$ results when $\mathscr{J}$ is minimized with respect to $\Delta\boldsymbol{x}$. For the linear function in $\Delta\boldsymbol{x}$, as long as the update direction is along a negative slope, then the larger the magnitude, the smaller value of $\mathscr{J}$. Since the purpose of the linearization is to determine the search direction for the nonlinear function, let $\alpha = 1$ temporarily, and fix the magnitude of the search direction such that $\|\boldsymbol{d}\| = a > 0$. This becomes

$$\mathscr{J} = F(\boldsymbol{x}+\boldsymbol{d}) - F(\boldsymbol{x}) = F_{\boldsymbol{x}}(\boldsymbol{x})\boldsymbol{d},$$

subject to

$$\boldsymbol{d}^T\boldsymbol{d} = a^2.$$

Minimizing $\mathscr{J}$ with respect to $\boldsymbol{d}$ can be interpreted as the best improvement on the linearized approximation for a given step size, a. If the step size is added as a constraint, then the augmented minor cost function is

$$\tilde{\mathscr{J}} = F_{\boldsymbol{x}}(\boldsymbol{x})\boldsymbol{d} + \lambda\left(\boldsymbol{d}^T\boldsymbol{d} - a^2\right).$$

By setting $\tilde{\mathscr{J}}_{\boldsymbol{d}} = \mathbf{0}^T$ and $\tilde{\mathscr{J}}_{\lambda} = 0$, the update is derived:

$$\boldsymbol{d} = -a\frac{F_{\boldsymbol{x}}^T(\boldsymbol{x})}{\|F_{\boldsymbol{x}}^T(\boldsymbol{x})\|}.$$

Since a is arbitrary, let it be equal to $\|F_{\boldsymbol{x}}^T(\boldsymbol{x})\|$ so that

$$\begin{aligned} \boldsymbol{d} &= -F_{\boldsymbol{x}}^T(\boldsymbol{x}), \\ \Delta\boldsymbol{x} &= \alpha\boldsymbol{d} = -\alpha F_{\boldsymbol{x}}^T(\boldsymbol{x}). \end{aligned}$$

Thus, the gradient $F_{\boldsymbol{x}}^T(\boldsymbol{x})$ indicates the *step direction* and α the *step size.* The step direction is defined by the gradient of the function at the current point, and it represents a move in the direction of *steepest descent.* Note for the linear approximation, the step is guaranteed to be downhill. That is,

$$\mathscr{J} = -\alpha F_{\boldsymbol{x}}(\boldsymbol{x})F_{\boldsymbol{x}}^T(\boldsymbol{x}) \leq 0$$

for $\alpha > 0$. Therefore, if α is small enough such that the linearization of Equation 2.10 is accurate, then the update is guaranteed to improve the current cost.

A demonstration of the Gradient technique is illustrated in Figure 2.2, which depicts a contour of $F(\boldsymbol{x})$ along with the path of $\boldsymbol{x}$ after several updates. At a given point, the update moves the point along the direction of steepest descent as far as possible. Where the line of the update direction is tangent to the contour of $F(\boldsymbol{x})$, α is optimized. For the next iteration, the update direction is necessarily perpendicular to the previous update direction. Therefore, the Gradient technique guides the point to the minimum through a series of 90 degree turns along the contours until tolerance is achieved.

2.2.1.2 Newton-Raphson Technique and Beyond

The Newton-Raphson technique uses the Taylor Series approximation to the cost function with three terms.

$$F(\boldsymbol{x} + d\boldsymbol{x}) = F(\boldsymbol{x}) + F_{\boldsymbol{x}}(\boldsymbol{x})d\boldsymbol{x} + \frac{1}{2}d\boldsymbol{x}^T F_{\boldsymbol{xx}}(\boldsymbol{x})d\boldsymbol{x} + ...$$

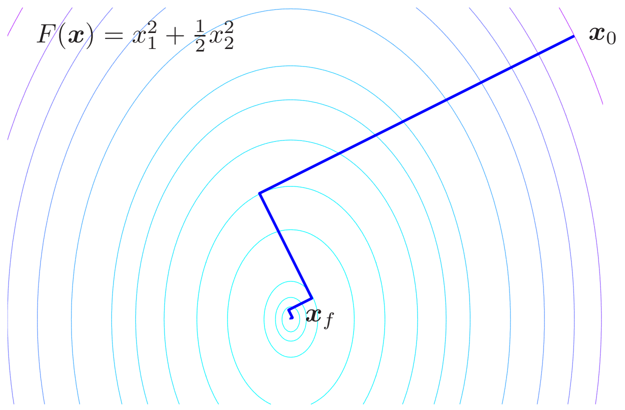


Figure 2.2: Iterative Path from Initial Guess, $\boldsymbol{x}_0$, to Final Solution, $\boldsymbol{x}_f$

As with the Gradient technique, the search direction, $\boldsymbol{d}$, is determined through the minor optimization problem, seeking to maximize the improvement on the cost function on the current iteration. Again, assume temporarily that $\alpha = 1$ so that the minor optimization variable is the search direction,

$$\mathscr{J} = F(\boldsymbol{x} + \Delta\boldsymbol{x}) - F(\boldsymbol{x}) = F(\boldsymbol{x} + \boldsymbol{d}) - F(\boldsymbol{x}) = F_{\boldsymbol{x}}(\boldsymbol{x})\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T F_{\boldsymbol{xx}}(\boldsymbol{x})\boldsymbol{d}.$$

The objective remains to minimize $\mathscr{J}$ with respect to $\boldsymbol{d}$. While the Gradient technique requires a constraint on the magnitude of $\boldsymbol{d}$, this minimization can remain unconstrained because of the quadratic form of the cost function. Under the assumption that the second derivative Hessian, $F_{\boldsymbol{xx}}(\boldsymbol{x}) > 0$ (positive definite), a minimum on $\mathscr{J}$ can be determined. Thus, taking the derivative with respect to $\boldsymbol{d}$ and setting it equal to zero,

$$\begin{aligned} \mathscr{J}_{\boldsymbol{d}} &= F_{\boldsymbol{x}}(\boldsymbol{x}) + \boldsymbol{d}^T F_{\boldsymbol{xx}}(\boldsymbol{x}) = 0, \\ \boldsymbol{d} &= -F_{\boldsymbol{xx}}^{-1}(\boldsymbol{x}) F_{\boldsymbol{x}}^T(\boldsymbol{x}). \end{aligned} \tag{2.11}$$

Equation 2.11 represents the optimal update (magnitude and direction) for a quadratic cost function (that is, $\Delta \boldsymbol{x} = \boldsymbol{d}$). With a higher-order or nonlinear cost function, then, this update direction is employed, but the best magnitude for α is determined with the one-dimensional line search. Thus, in general, the Newton-Raphson update is

$$\Delta \boldsymbol{x} = \alpha \boldsymbol{d} = -\alpha F_{\boldsymbol{xx}}^{-1}(\boldsymbol{x}) F_{\boldsymbol{x}}^{T}(\boldsymbol{x}). \tag{2.12}$$

Although it is generally much faster than the Gradient technique, two significant drawbacks to the Newton-Raphson technique exist. First, as seen in Equation 2.12, the Hessian inverse must be calculated. For higher-dimensional optimization problems, a formal calculation of the inverse may be costly. Consequently, it is recommended to use less time consuming methods for solving the linear equation,

$$F_{\boldsymbol{xx}}(\boldsymbol{x})\boldsymbol{d} = -F_{\boldsymbol{x}}^{T}(\boldsymbol{x}).$$

In addition, it is possible for a higher-order or nonlinear cost function to have a non-positive definite Hessian at the current point of $\boldsymbol{x}$, even when the cost function contains a minimum. In this case, a common practice is to use a Gradient technique until the Hessian becomes positive definite. (Near the minimum, the Hessian is necessarily positive definite.)

Significant theoretical work has gone into improving upon the Newton-Raphson iteration. More efficient ways of updating the Hessian matrix have been discovered so that it does not have to be directly calculated on each iteration. *Variable metric* or *quasi-Newton* techniques comprise those that follow the basic update of Equation 2.12 with creative Hessian updates, $\Delta F_{\boldsymbol{xx}}$. Notable among these are the Davidon-Fletcher-Powell and Broyden-Fletcher-Shanno-Goldfarb update algorithms.[31,32] Most current nonlinear programming algorithms employ one of these Hessian update procedures to increase the efficiency of the algorithm.

2.2.2 Constrained Optimization

Consider the constrained problem, where Equation 2.9 is subject to equality and inequality constraints, respectively,

$$\begin{aligned} c_i(\boldsymbol{x}) &= 0, \;\; i = 1, \ldots, n_c, \\ \tilde{c}_j(\boldsymbol{x}) &\leq 0, \;\; j = 1, \ldots, n_{\tilde{c}}. \end{aligned}$$

One way of determining the solution to this optimization problem is to convert it into an unconstrained problem and use one of the treatments described above. A generalized approach to accomplish this, one that accommodates all forms of constraints, involves the use of penalty functions. For example, define positive weights, $\boldsymbol{w}$ and $\tilde{\boldsymbol{w}}$, and modify the cost function to:

$$\tilde{\mathfrak{J}} = F(\boldsymbol{x}) + \sum_{i=1}^{n_c} w_i \, |c_i| + \sum_{j=1}^{n_{\tilde{c}}} \tilde{w}_j \max\{0, \tilde{c}_j\}.$$

In this formulation, penalties are applied to any of the unsatisfied constraint equations, according to the weights. Although the minimum of this function is also the minimum of the unpenalized function when the constraints are met, careful consideration must go into the initial guess and the values of the weights to ensure that the optimization algorithm does not settle on a local minimum that does not satisfy the constraints.

A more commonly used approach to constrained optimization is *sequential quadratic programming* (SQP). Define the augmented cost function, also known as the Lagrangian, as

$$\tilde{\mathfrak{J}} = L(\boldsymbol{x}, \boldsymbol{\eta}) = F(\boldsymbol{x}) + \boldsymbol{\eta}^T \begin{bmatrix} \boldsymbol{c} \\ \tilde{\boldsymbol{c}} \end{bmatrix}.$$

At each iteration, a quasi-Newton algorithm is used to approximate the Hessian matrix of the Lagrangian function, $L_{\boldsymbol{xx}}$. Then, a minor optimization problem is derived from the quadratic approximation of the Lagrangian,

$$\mathscr{J} = L(\boldsymbol{x} + \boldsymbol{d}) - L(\boldsymbol{x}) = L_{\boldsymbol{x}}(\boldsymbol{x})\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T L_{\boldsymbol{xx}}(\boldsymbol{x})\boldsymbol{d},$$

with linearized constraints

$$\begin{aligned}(c_i)_{\boldsymbol{x}}\,\boldsymbol{d} + c_i &= 0, \quad i = 1, \ldots, n_c,\\ (\tilde{c}_j)_{\boldsymbol{x}}\,\boldsymbol{d} + \tilde{c} &\leq 0, \quad j = 1, \ldots, n_{\tilde{c}}.\end{aligned}$$

The minor optimization problem is a quadratic programming problem in $\boldsymbol{d}$. Note that it is similar in form to the second order unconstrained minor problem. The only differences are that the quadratic, $\mathscr{J}$, is an approximation of the augmented cost function, instead of an unconstrained cost function, and that there are now linear equality and inequality constraints. With the new constraints, an analytic solution for $\boldsymbol{d}$ is no longer available, but a solution can generally be found easily using an iterative quadratic programming algorithm.

The method receives its title in that, on every major iteration (which determines α and $\boldsymbol{d}$), a minor optimization is solved with quadratic programming. As it turns out, this is an efficient way of solving the constrained problem. Often, SQP algorithms can converge on a constrained problem even quicker than on an unconstrained problem, as the constraints serve to limit the solution search space.

2.3 Numerical Methods of Solving the Optimal Control Problem

The techniques described above treat constrained and unconstrained function minimization problems to determine an optimal point, $\boldsymbol{x}$. The optimal control problem of Equations 1.4, 1.1, and 1.5, however, seeks a minimum function, $\boldsymbol{u}(t)$. Thus, to use the techniques described above, it is necessary to determine a method for transforming the optimal control problem into a function minimization problem. A problem in $\boldsymbol{x}$ must be devised, such that the n parameters represent, in some way, the information necessary to identify the constants and time-varying states, controls, and Lagrange multipliers of the optimal control problem. The method determines the form of the equality and inequality constraints $\boldsymbol{c}(\boldsymbol{x})$ and $\tilde{\boldsymbol{c}}(\boldsymbol{x})$, as well as the optimizing function, $F(\boldsymbol{x})$.

Many methods exist for reformulating the optimal control problem so that it may be implemented using a standard NLP technique. The number of parameters required can vary greatly across methods. Each method, of course, has its advantages and disadvantages. For a given problem, the most suitable method is sought. In this section, several methods are outlined and contrasted, through an example, to identify a method suitable for the applications explored in this investigation.

The methods for transforming an optimal control problem, in preparation for the subsequent numerical solution process, fall into one of two categories: indirect and direct. One distinction between the two categories is that *indirect* methods rely heavily on the theory presented in Section 2.1, while *direct* methods take a more 'brute force' approach. The direct methods are classified further based on how the differential constraints are imposed. Explicit integration, implicit integration, and spectral (differencing) methods are discussed here.

The performance of each method can be characterized by the accuracy of the numerical solution as compared to an analytic solution, the ease of convergence, the speed of convergence, robustness to initial guesses, and so on. In some cases, the problem may dictate the appropriate method to use, depending on which of these characteristics is more important.

2.3.1 Optimal Control Solutions via the Indirect Method

An indirect method of solving the optimal control problem employs the analytical results derived in Section 2.1.1. All optimal control problems can be reformulated into a two-point boundary value problem (BVP) in $\boldsymbol{y}$ and $\boldsymbol{\lambda}$, and an indirect method attempts to solve the BVP numerically.

Consider again the Euler-Lagrange conditions,

$$\begin{aligned} \dot{\boldsymbol{y}} &= \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \\ \dot{\boldsymbol{\lambda}} &= -H_{\boldsymbol{y}}^T(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \\ \boldsymbol{0} &= H_{\boldsymbol{u}}^T(t, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \end{aligned} \tag{2.7}$$

and the boundary conditions,

$$\begin{array}{ccc} \boldsymbol{\psi}_0 = \boldsymbol{0}, & \boldsymbol{\psi}_f = \boldsymbol{0}, & \boldsymbol{\beta} = \boldsymbol{0}, \\ H_0 = G_{t_0}, & H_f = -G_{t_f}, & \\ \boldsymbol{\lambda}_0 = -G_{\boldsymbol{y}_0}^T, & \boldsymbol{\lambda}_f = G_{\boldsymbol{y}_f}^T. & \end{array} \tag{2.8}$$

Recall that the path constraint Lagrange multipliers, $\boldsymbol{\mu}$, are chosen to make $H_{u_i} = 0$ for all controls, $i = 1, \ldots, n_u$. Then, as a minimum, $\boldsymbol{\mu}$ is known as a function of t, $\boldsymbol{y}$, and $\boldsymbol{\lambda}$. Thus, with the equations $\boldsymbol{\beta} = \boldsymbol{0}$ and $H_{\boldsymbol{u}} = \boldsymbol{0}^T$, it is possible to determine the control law,

$$\boldsymbol{u} = \boldsymbol{u}(t, \boldsymbol{y}, \boldsymbol{\lambda}),$$

and the control input can be completely eliminated from the equations. Subsequently, an augmented state vector may be defined as $\boldsymbol{z} = \left[\boldsymbol{y}^T \ \boldsymbol{\lambda}^T\right]^T$. Note that the differential equations describing the augmented states are of the form $\dot{\boldsymbol{z}} = \dot{\boldsymbol{z}}(t, \boldsymbol{z})$:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\boldsymbol{y}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}\left(t, \boldsymbol{y}, \boldsymbol{u}(t, \boldsymbol{y}, \boldsymbol{\lambda})\right) \\ -H_{\boldsymbol{y}}^T\left(t, \boldsymbol{y}, \boldsymbol{u}(t, \boldsymbol{y}, \boldsymbol{\lambda}), \boldsymbol{\lambda}, \boldsymbol{\mu}(t, \boldsymbol{y}, \boldsymbol{\lambda})\right) \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{y}}(t, \boldsymbol{z}) \\ \dot{\boldsymbol{\lambda}}(t, \boldsymbol{z}) \end{bmatrix}.$$

Along with the remaining boundary conditions, the result is a two-point BVP on $[t_0 \ t_f]$.

The BVP can be implemented into a numerical algorithm. As an illustrative example, consider the case where the initial and final values of the states and time are specified. That is,

$$\begin{aligned} \boldsymbol{\psi}_0 &= \begin{bmatrix} \boldsymbol{y}_0 - \boldsymbol{y}_0^* \\ t_0 - t_0^* \end{bmatrix} = \boldsymbol{0}, \\ \boldsymbol{\psi}_f &= \begin{bmatrix} \boldsymbol{y}_f - \boldsymbol{y}_f^* \\ t_f - t_f^* \end{bmatrix} = \boldsymbol{0}. \end{aligned}$$

The corresponding Lagrange multipliers for the endpoints are $\boldsymbol{\nu}_0 = \left[\boldsymbol{\nu}_{\boldsymbol{y}_0}^T \ \nu_{t_0}\right]^T$ and $\boldsymbol{\nu}_f = \left[\boldsymbol{\nu}_{\boldsymbol{y}_f}^T \ \nu_{t_f}\right]^T$. In this case, the remaining boundary conditions for the costates are $\boldsymbol{\lambda}_0 = -\boldsymbol{\nu}_{\boldsymbol{y}_0}$

and $\boldsymbol{\lambda}_f = \boldsymbol{\nu}_{\boldsymbol{y}_f}$, which are both unknown. Thus, half of the initial conditions, $\boldsymbol{z}_0$, and half of the final conditions, $\boldsymbol{z}_f$, are known. All that is left is to define an initial guess for $\boldsymbol{\lambda}_0$, integrate $\dot{\boldsymbol{z}}$ forward in time from t_0 to t_f, and continue to iterate on $\boldsymbol{\lambda}_0$ until the integrated values of $\boldsymbol{y}_f$ match the specified values, $\boldsymbol{y}_f^*$.

This can be expressed using a shooting function, $\boldsymbol{g}$, where

$$\boldsymbol{y}_f = \boldsymbol{g}(\boldsymbol{\lambda}_0),$$

and the values of $\boldsymbol{y}_0^*$, t_0^*, and t_f^* are embedded into $\boldsymbol{g}$ in order to perform the integration of the augmented states. Previously, the numerical techniques formulate the problem as a cost index, $\mathcal{J} = F(\boldsymbol{x})$, subject to constraints, $\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$. Let the constraint equations be defined as

$$\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{y}_f^*.$$

Then the parameters to be optimized in $\boldsymbol{x}$ are simply the initial values of the costates, $\boldsymbol{\lambda}_0$, and the size of the problem is simply $n = n_y$.

The problem set up is complete even though $F(\boldsymbol{x})$ is not defined. Note that the optimization has already been accomplished, since the first order conditions of optimality are intrinsically satisfied through the integration of $\dot{\boldsymbol{z}}$. The numerical technique is only necessary for determining the values of $\boldsymbol{x}$ that satisfy the constraints. When that is accomplished, both $\boldsymbol{y}$ and $\boldsymbol{\lambda}$ are known as a function of time, and the control, $\boldsymbol{u}(t, \boldsymbol{y}, \boldsymbol{\lambda})$, can be extracted.

The process just described is sometimes called *shooting.* Essentially, the guessed initial values are shot to the final time, and then modifications are made to the initial values through iteration until the constraints are satisfied. In this work, this method is specified as *indirect shooting* to distinguish it from a method to be described below. Obviously, this indirect shooting example is only for a fixed time problem with initial and final states specified. However, the concept is the same for different sets of initial and final conditions, and the parameters that go into $\boldsymbol{x}$ vary from problem to problem, along with the constraints that must be satisfied.

Characteristics of Indirect Optimization

Because this process employs the Euler-Lagrange equations derived from variational calculus, one expects the solution obtained from an indirect method to be the exact optimal solution, if one is available. Thus, solutions obtained through indirect methods are more accurate than those acquired through the use of direct methods.

One drawback to indirect shooting originates from the need to supply an initial guess for the Lagrange multipliers. Thus, the convergence of the numerical algorithm is dependent upon a priori insight into the behavior of the costates, an aspect of the problem that is generally non-intuitive. Although some costates for certain problems might have physical significance, it is very difficult to guess their initial values within a reasonable level of accuracy. In addition, the costates are generally quite unstable. Small changes to their initial values often produce large changes over their integration. This sensitivity makes it difficult for a numerical algorithm to converge on the correct values. Thus, the user is generally responsible for identifying a reasonably accurate initial guess for the parameters, although this may not be possible for every problem.

Additionally, the above demonstration implies that the formulation of the BVP is significantly altered whenever the boundary conditions are changed. Certainly, a different dynamic model or cost function requires the formulation to be modified. However, even the simplest change to the boundary conditions implies a change of potentially both input and output arguments of $\boldsymbol{g}$. This characteristic, by itself, is a potential deterrent when a generalized methodology is desired.

2.3.2 Optimal Control Solutions via Direct Methods

As an alternative, direct methods do not require insight into the Lagrange multipliers, as the Euler-Lagrange equations are not applied explicitly. There is an intuitive appeal here; actually, limited understanding of optimal control theory is required to solve a problem. At the most basic level, all direct methods attempt to discretize the problem of Equations 1.4, 1.1, and 1.5 in order to produce parameters that the numerical techniques can optimize.

The most simple implementation of this is *direct shooting.* Here, a continuous optimal control problem is transcribed into a parameter optimization problem where the parameters are simply the control values at specified (or derived) points in time. Definitions of F and $\boldsymbol{c}$ are derived in order to represent the cost function and constraints of the optimal control problem in terms of the control values (which will necessarily dictate the states) . Similar to indirect shooting, a function is generated that shoots the initial conditions to the final conditions,

$$\boldsymbol{y}_f = \boldsymbol{g}(\boldsymbol{u}),$$

and is used in F and $\boldsymbol{c}$. The concept is intuitive: if the objective is to determine an optimal control, let the optimization parameters be the control values.

Discretize the problem in time at n_n different places, called *nodes*, associated with times, $t_j,\ j = 1, \ldots, n_n$. Then the parameters to optimize represent the values of the controls at those times.

$$\boldsymbol{x} = \left[u_1(t_0)\ \cdots\ u_i(t_j)\ \cdots\ u_{n_u}(t_f)\right]^T .$$

If the discretized times are assumed to be uniformly spaced over the interval $[t_0\ t_f]$, then

$$t_j = t_0 + \frac{j-1}{n_n - 1}\left(t_f - t_0\right) . \tag{2.13}$$

If the initial and final times are free in the optimal control problem, then t_0 and t_f are included as optimization parameters in $\boldsymbol{x}$. In this case, the number of required parameters becomes

$$n = n_u n_n + 2$$

where n_u is the dimension of the control, n_n is the number of nodes, and two parameters are designated for the unspecified initial and final times. Upon each iteration, the current control values are explicitly integrated from the specified initial states and time to the current final time. Then, the cost index and constraint conditions are evaluated in order to choose a search direction to improve their values according to the numerical technique.

With direct shooting, insight into the costates is not required. Instead, only a guess for the terminal times and the control values at the nodes is required to initiate an optimization algorithm. The associated values of the states are derived through explicit integration.

One may take advantage of available insight regarding the 'shape' of the trajectory by modifying a direct shooting scheme to *multiple direct shooting*. The shape of a trajectory refers to the physical states. By including as parameters the values of the states at some of the node points, the states need only be integrated between parameterized state values. At each iteration, the integrated states are compared to current values of the parameterized states; the difference is called the residual. For each state that is included as a parameter, a constraint equation is generated to allow the optimizer to drive residual errors to zero, resulting in a smooth trajectory when all constraints are satisfied to tolerance. As a limiting case, states can be parameterized at each of the nodes where the controls are parameterized.

Consider a parameter vector containing each of the states and controls at the nodes, along with the initial and final time.

$$\boldsymbol{x} = \left[y_1(t_0)\ \cdots\ y_i(t_j)\ \cdots\ y_{n_y}(t_f)\ u_1(t_0)\ \cdots\ u_i(t_j)\ \cdots\ u_{n_u}(t_f)\ t_0\ t_f\right]^T \tag{2.14}$$

State variables, control variables, and time are all represented in this parameterization so that cost and constraint functions in terms of $\boldsymbol{x}$ can easily be derived to represent the optimal control problem. Depending on how the constraints are defined, this parameterization goes by several names. It is used in the limiting case of multiple direct shooting, but it is also used for *collocation* (or *implicit integration methods*) and for *spectral methods* (or *differencing methods*).

Under this parameterization, the number of parameters is potentially much greater than for both indirect and direct shooting. The parameter vector in Equation 2.14 is of dimension

$$n = (n_y + n_u)n_n + 2,$$

indicating the number of parameters that are optimized by the NLP algorithm.

The parameters thus defined as in Equation 2.14, the cost and constraint functions can be defined in terms of $\boldsymbol{x}$. The vector constraint equation, $\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$, accounts for the initial, final, path, and dynamical constraints. Specifically, let $\boldsymbol{c}(\boldsymbol{x})$ be decomposed as

$$\boldsymbol{c}(\boldsymbol{x}) = \left[\boldsymbol{c}_{\psi_0}^T(\boldsymbol{x})\ \boldsymbol{c}_{\psi_f}^T(\boldsymbol{x})\ \boldsymbol{c}_{\beta}^T(\boldsymbol{x})\ \boldsymbol{c}_{\dot{y}}^T(\boldsymbol{x})\right]^T. \tag{2.15}$$

Each of the elements of Equation 2.15 is defined next. The first three sets of constraints are straightforward, and the rules outlined here apply for all of the methods using this parameterization. Enforcing the dynamical (i.e. continuity) constraints, on the other hand, requires further consideration.

2.3.2.1 Point and Path Constraints

Recall the initial, final, and path conditions as

$$\begin{aligned} \boldsymbol{0} &= \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0), \\ \boldsymbol{0} &= \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f), \\ \boldsymbol{0} &= \boldsymbol{\beta}(t, \boldsymbol{y}, \boldsymbol{u}). \end{aligned} \tag{1.5}$$

The objective is to generate equations in $\boldsymbol{x}$ that represent these conditions. For the initial and final conditions, however, the arguments of $\boldsymbol{\psi}_0$ and $\boldsymbol{\psi}_f$ are contained within the parameter vector since the first node is the initial point and the last node is the final point. Therefore, the initial and final constraints translate directly, as

$$\begin{aligned} \boldsymbol{c}_{\psi_0}(\boldsymbol{x}) &= \boldsymbol{\psi}_0(t_0, \boldsymbol{y}(t_0)), \\ \boldsymbol{c}_{\psi_f}(\boldsymbol{x}) &= \boldsymbol{\psi}_f(t_f, \boldsymbol{y}(t_f)), \end{aligned}$$

where $\boldsymbol{c}_{\psi_0}$ contributes n_{ψ_0} constraints and $\boldsymbol{c}_{\psi_f}$ contributes n_{ψ_f} constraints.

The path constraint, $\boldsymbol{\beta}$, is enforced continuously for all $t \in [t_0\ t_f]$. With a parameterized set of states, controls, and times, a logical way of implementing a path constraint is at each node individually. The n_β path constraints are imposed at each of the nodes, such that

$$\begin{aligned} \boldsymbol{c}_{\beta_j} &= \boldsymbol{\beta}(t_j, \boldsymbol{y}(t_j), \boldsymbol{u}(t_j)), \\ \boldsymbol{c}_\beta &= \begin{bmatrix} \boldsymbol{c}_{\beta_1}^T & \cdots & \boldsymbol{c}_{\beta_{n_n}}^T \end{bmatrix}^T. \end{aligned}$$

This leads to $n_\beta n_n$ path constraints. Note that, in this formulation, the path constraints are not necessarily enforced between the nodes. This is easily resolved, for example, by selecting a representative number of nodes in the transcription. Making n_n large limits the spacing between nodes and increases the number of path constraints. As $n_n \to \infty$, the constraints of $\boldsymbol{c}_\beta$ approach the continuous path constraints $\boldsymbol{\beta}$.

2.3.2.2 Continuity Constraints

In this section, the dynamical constraints of Equation 1.1 are transcribed into equations of the form

$$\boldsymbol{c}_{\dot{y}}(\boldsymbol{x}) = \boldsymbol{0}.$$

The presence of dynamical constraints is the most important distinction between an optimal control problem and a parameter optimization problem. Thus, in the conversion process, identifying the dynamical constraints is the central characteristic of the transcription. The literature cites many different forms; some of them are compared here. In this study, the methods are divided into integration methods and differencing methods. The integration methods are further divided between explicit and implicit integration methods; the differencing methods are sometimes known as spectral methods.

Consider momentarily the one-dimensional equation, $\dot{y} = f(t, y)$. If $h = t_{j+1} - t_j$ defines the spacing between two nodes, then the first order explicit (Euler) integration formula is defined as

$$y_{j+1} = y_j + hf(t_j, y_j), \tag{2.16}$$

where $y_j = y(t_j)$. This can also be written as

$$0 = y_{j+1} - y_j - hf(t_j, y_j), \tag{2.17}$$

a more convenient form when the formula is employed as a dynamical constraint. Rearranging Equation 2.16, one arrives at the first order forward differencing equation:

$$\frac{y_{j+1} - y_j}{h} = f(t_j, y_j),$$

which can also be rewritten as

$$0 = \frac{y_{j+1} - y_j}{h} - f(t_j, y_j). \tag{2.18}$$

Equations 2.17 and 2.18 are essentially the same equation, scaled by h. Thus, to first order, integration and differencing are quite similar. The methods diverge as the order of accuracy increases, but on a basic level the concept is the same. In order to establish constraints to represent Equation 1.1, sets of equations like those in Equations 2.17 and 2.18 are imposed at each of the nodes.

Note also that a similar relation exists between the first order implicit integration formula and the first order backward differencing equation:

$$\begin{aligned} y_{j+1} &= y_j + hf(t_{j+1}, y_{j+1}), \\ \frac{y_{j+1} - y_j}{h} &= f(t_{j+1}, y_{j+1}). \end{aligned} \tag{2.19}$$

In Equation 2.19, y_{j+1} appears on both sides of the equation, and iteration is required to arrive at its proper value. This is the primary distinction between implicit and explicit integration. It can be shown that the implicit methods are more accurate for a given order (that is, a given number of function evaluations), but at the cost of increased iteration. However, the numerical techniques used in optimization are already iterative in nature, so no significant added expense is incurred if implicit integration is selected as the method of choice.

Continuity Constraints using Explicit Euler Integration

The explicit Euler formula of Equation 2.16 is derived from the Explicit Runge-Kutta Formulation[33]

$$\begin{aligned} y_{j+1} &= y_j + h\sum_{i=1}^{p} c_i f_i, \\ f_1 &= f(t_j, y_j), \\ f_i &= f\left(t_j + h\alpha_i, y_j + h\sum_{k=1}^{i-1} \beta_{jk} f_k\right), \end{aligned}$$

with $p = 1$ and the variables α_i and β_{jk} chosen optimally. Therefore, only one evaluation of the function f is required to shoot a state from one node to the next.

Now consider the optimal control problem with states $\boldsymbol{y}$, whose values are known at the nodes. More precisely, the states at the nodes are simply parameters that must be optimized while meeting constraints. Therefore, the values of $\boldsymbol{y}_j$ associated with the current iteration are known. The explicit Euler formula can be used to shoot the states, $\boldsymbol{y}_j$, to the next node at t_{j+1}. This provides an approximation for the states, denoted $\hat{\boldsymbol{y}}_{j+1}$.

$$\hat{\boldsymbol{y}}_{j+1} = \boldsymbol{y}_j + h\boldsymbol{f}(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j)$$

Since the states, $\hat{\boldsymbol{y}}_{j+1}$, do not necessarily match the known states, $\boldsymbol{y}_{j+1}$, an equality constraint can be imposed in the parameter optimization problem to ensure that they do match at the final iteration. Thus, define the constraint,

$$\begin{aligned} \boldsymbol{c}_{\dot{y}_j} &= \boldsymbol{y}_{j+1} - \hat{\boldsymbol{y}}_{j+1} \\ &= \boldsymbol{y}_{j+1} - \boldsymbol{y}_j - h\boldsymbol{f}(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j). \end{aligned} \tag{2.20}$$

This constraint is imposed between each node for $j = 1, ...n_n - 1$. Assembling all of these constraints into a single vector leads to $n_y(n_n - 1)$ continuity constraints,

$$\boldsymbol{c}_{\dot{y}} = \begin{bmatrix} \boldsymbol{c}_{\dot{y}_1}^T & \cdots & \boldsymbol{c}_{\dot{y}_{n_n-1}}^T \end{bmatrix}^T. \tag{2.21}$$

When $\boldsymbol{c}_{\dot{y}} = \boldsymbol{0}$, to within some specified tolerance, the dynamical constraint, $\dot{\boldsymbol{y}} = \boldsymbol{f}$ is satisfied.

Continuity Constraints using Explicit Trapezoidal Integration

The Runge-Kutta formulation for $p = 2$ yields a second order explicit integration scheme.

$$\hat{\boldsymbol{y}}_{j+1} = \boldsymbol{y}_j + \frac{h}{2}\left[\boldsymbol{f}(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j) + \boldsymbol{f}\left(t_{j+1}, \boldsymbol{y}_j + h\boldsymbol{f}(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j), \boldsymbol{u}_{j+1}\right)\right]$$

Note that the Euler formula is embedded in the second function evaluation. If this relation is combined with Equations 2.20 and 2.21, then $n_y(n_n - 1)$ continuity constraints result with errors now on the order $\mathcal{O}(h^3)$.

Continuity Constraints using Implicit Trapezoidal Integration

The Runge-Kutta Formulation for implicit integration is

$$\begin{aligned} y_{j+1} &= y_j + h \sum_{i=1}^{p} c_i f_i, \\ f_1 &= f(t_j, y_j), \\ f_i &= f\left(t_j + h\alpha_i, y_j + h \sum_{k=1}^{p} \beta_{jk} f_k \right). \end{aligned} \tag{2.22}$$

Note the subtle difference from the explicit formulation: the summation in Equation 2.22 is upper bounded by p. Thus, all of the f_i equations are in the definitions of each other.

The implicit trapezoidal integration scheme is not much different than the explicit, except that $\boldsymbol{y}_{j+1}$ now appears on both sides of the equation,

$$\hat{\boldsymbol{y}}_{j+1} = \boldsymbol{y}_j + \frac{h}{2} \left[\boldsymbol{f}(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j) + \boldsymbol{f}\left(t_{j+1}, \boldsymbol{y}_{j+1}, \boldsymbol{u}_{j+1}\right) \right].$$

Again, the iterative nature of the integration equation is well-suited for an iterative optimization method. Equations 2.20 and 2.21 are used to generate the $n_y(n_n - 1)$ continuity constraints with the improved integration scheme.

Continuity Constraints using Hermite-Simpson Integration

The Hermite-Simpson integration is the Runge-Kutta implicit integration scheme with function evaluations $p = 3$, delivering an accuracy on the order of $\mathcal{O}(h^5)$. Define

$$\begin{aligned} t_m &= \frac{1}{2}(t_j + t_{j+1}), \\ \boldsymbol{y}_m &= \frac{1}{2}\left(\boldsymbol{y}_j + \boldsymbol{y}_{j+1}\right) + \frac{h}{8}\left(\boldsymbol{f}_j - \boldsymbol{f}_{j+1}\right), \\ \boldsymbol{u}_m &= \frac{1}{2}(\boldsymbol{u}_j + \boldsymbol{u}_{j+1}), \\ \boldsymbol{f}_m &= \boldsymbol{f}(t_m, \boldsymbol{y}_m, \boldsymbol{u}_m). \end{aligned}$$

Then, the estimate of the state at t_{j+1} is given by[33]

$$\hat{\boldsymbol{y}}_{j+1} = \boldsymbol{y}_j + \frac{h}{6}\left(\boldsymbol{f}_j + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1}\right).$$

The residual defects between nodes j and $j+1$ are calculated as

$$\boldsymbol{c}_{\dot{y}_j} = \boldsymbol{y}_{j+1} - \boldsymbol{y}_j - \frac{h}{6}\left(\boldsymbol{f}_j + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1}\right). \tag{2.23}$$

As before, the residual equation is a vector of size n_y, calculated for all $j = 1, \ldots, n_n - 1$, and applied to the constraint vector in Equation 2.21. When the defects are driven to zero through the optimization process, and enough nodes have been applied to the discretization, Equation 1.1 is satisfied to a tolerance within the accuracy of the integration scheme.

Continuity Constraints using Fourth-Order Differencing

An alternative approach for defining continuity constraints uses finite differences instead of numerical integration equations. With integration schemes, the accuracy is improved by using more function evaluations in the comparison between the j^{th} and the $(j+1)^{\text{th}}$ node. With finite differences, improvements in accuracy are achieved by using more points in the evaluation of the derivative approximation. Since the values of the states are already known at the nodes, the objective is to use these values to approximate the derivatives at the nodes.

Consider a single state, whose values, y_j, are known at each node. Define the vector

$$\bar{y} = [y_1 \ \cdots \ y_j \ \cdots \ y_{n_n}]^T$$

to be values of the single state at each node. Spectral methods seek to determine an approximation of the derivatives, $\dot{\bar{y}}$, by using a linear combination of the values, $\bar{y}$. Specifically, let the derivative at the nodes be approximated as

$$\dot{\bar{y}} = \boldsymbol{D}\bar{y}, \tag{2.24}$$

where $\boldsymbol{D}$ is the $n_n \times n_n$ differentiation matrix, and $D_{i,j}$ denotes the element located in the i^{th} row of the j^{th} column. Now for the n_y state vector, define the matrix $\bar{\boldsymbol{y}} \in \mathbb{R}^{n_n \times n_y}$ to be all of the states at each of the nodes, and $\bar{\boldsymbol{f}} \in \mathbb{R}^{n_n \times n_y}$ to be the values of the function at each state and each node. Thus, $\bar{y}_{j,i} = y_i(t_j)$ and $\bar{f}_{j,i} = f_i(t_j, \boldsymbol{y}_j, \boldsymbol{u}_j)$. Define $\boldsymbol{C}$ as a matrix of constraints, where

$$\begin{aligned} \boldsymbol{C} &= \dot{\bar{\boldsymbol{y}}} - \bar{\boldsymbol{f}}, \\ &= \boldsymbol{D}\bar{\boldsymbol{y}} - \bar{\boldsymbol{f}}. \end{aligned}$$

If $\boldsymbol{C}$ is driven to zero to within some specified tolerance, then Equation 1.1 is satisfied. In this case, the continuity constraints, $\boldsymbol{c}_{\dot{y}}(\boldsymbol{x})$, are the elements of $\boldsymbol{C}$,

$$\boldsymbol{c}_{\dot{y}} = \begin{bmatrix} C_{1,1} & \cdots & C_{n_n,n_y} \end{bmatrix}^T .$$

Note that there are a total of $n_y n_n$ continuity constraints. The exact form of the differentiation matrix, $\boldsymbol{D}$, is yet to be determined at this point.

One candidate representation for $\boldsymbol{D}$ employs the Taylor series expansion of five points near t_j of the single state, y, truncated after the first five terms.

$$\begin{array}{rcccccccccc} y_{j-2} & = & y_j & - & 2h\dot{y}_j & + & \frac{4h^2}{2!}\ddot{y}_j & - & \frac{8h^3}{3!}\dddot{y}_j & + & \frac{16h^4}{4!}y_j^{(4)} \\ y_{j-1} & = & y_j & - & h\dot{y}_j & + & \frac{h^2}{2!}\ddot{y}_j & - & \frac{h^3}{3!}\dddot{y}_j & + & \frac{h^4}{4!}y_j^{(4)} \\ y_j & = & y_j & & & & & & & & \\ y_{j+1} & = & y_j & + & h\dot{y}_j & + & \frac{h^2}{2!}\ddot{y}_j & + & \frac{h^3}{3!}\dddot{y}_j & + & \frac{h^4}{4!}y_j^{(4)} \\ y_{j+2} & = & y_j & + & 2h\dot{y}_j & + & \frac{4h^2}{2!}\ddot{y}_j & + & \frac{8h^3}{3!}\dddot{y}_j & + & \frac{16h^4}{4!}y_j^{(4)} \end{array} \tag{2.25}$$

Using the relations in Equation 2.25, it is possible to approximate $\dot{y}_j$ with a linear combination of these five points. The approximation is determined as the derivative at t_j of a 4th order interpolating polynomial that passes through each of the five points. Express $\dot{y}_j$ as

$$h\dot{y}_j = a_{j-2}y_{j-2} + a_{j-1}y_{j-1} + a_j y_j + a_{j+1}y_{j+1} + a_{j+2}y_{j+2}, \tag{2.26}$$

where a_{j-2} through a_{j+2} represent a set of coefficients to be determined. Inserting the truncations of Equation 2.25 into Equation 2.26, and matching coefficients for each term of the form $h^k y_j^{(k)}$, leads to five linear equations in five unknowns.

$$\begin{array}{lcccrcrcccrcr}
h^0 y_j & : & 0 & = & a_{j-2} & + & a_{j-1} & + & a_j & + & a_{j+1} & + & a_{j+2} \\
h^1 \dot{y}_j & : & 1 & = & -2a_{j-2} & - & a_{j-1} & & & + & a_{j+1} & + & 2a_{j+2} \\
h^2 \ddot{y}_j & : & 0 & = & 2a_{j-2} & + & \frac{1}{2}a_{j-1} & & & + & \frac{1}{2}a_{j+1} & + & 2a_{j+2} \\
h^3 \dddot{y}_j & : & 0 & = & -\frac{4}{3}a_{j-2} & - & \frac{1}{6}a_{j-1} & & & + & \frac{1}{6}a_{j+1} & + & \frac{4}{3}a_{j+2} \\
h^4 y_j^{(4)} & : & 0 & = & \frac{2}{3}a_{j-2} & + & \frac{1}{24}a_{j-1} & & & + & \frac{1}{24}a_{j+1} & + & \frac{2}{3}a_{j+2}
\end{array}$$

When the system of equations is solved, the fourth order differencing equation is

$$\dot{y}_j = \frac{y_{j-2} - 8y_{j-1} + 8y_{j+1} - y_{j+2}}{12h}.$$

This derivative approximation uses two node points before and two node points after the node for which the derivative is being evaluated. For the nodes near the beginning and end of the trajectory, it may not be possible to use as many points on each side of the node. However, the concept shown above can be applied for any number of points before and after the node. For example, the differentiation matrix associated with the use of two forward and two backward points is

$$\boldsymbol{D} = \frac{1}{h}\begin{bmatrix}
-\frac{3}{2} & 2 & -\frac{1}{2} & & & & \\
-\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} & & & \\
\frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\
 & & & \frac{1}{6} & -1 & \frac{1}{2} & \frac{1}{3} \\
 & & & & \frac{1}{2} & -2 & \frac{3}{2}
\end{bmatrix}.$$

Along the interior nodes, the differencing produces fourth-order accuracy. At the nodes t_0 and t_f, second order accuracy is achieved.

Continuity Constraints for Two Pseudospectral Methods

The limiting cases of the spectral methods are those that use *every* node point available to evaluate the derivative at every node. This is equivalent to fitting an interpolating polynomial through all of the nodes, and evaluating the derivative of the polynomial at each of the nodes. In the Lagrange form, the fitted polynomial can be written as

$$\mathcal{Y}(t) = \sum_{j=1}^{n_n} \phi_j(t) y_j,$$

where

$$\phi_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^{n_n} \frac{t - t_k}{t_j - t_k}.$$

The node points, y_j, may be treated as constants, so the derivative of the fitted polynomial is simply

$$\dot{\mathcal{Y}}(t) = \sum_{j=1}^{n_n} \dot{\phi}_j(t) y_j. \tag{2.27}$$

The objective is to determine, based on this structure, the value of this derivative at each of the nodes, that is, $\dot{\mathcal{Y}}(t_i)$ which will serve as the approximation for $\dot{y}$. Once again, the derivative approximation is expressed in terms of the differentiation matrix, as in Equation 2.24. Based on the form of Equation 2.27, it is evident that the elements of the differentiation matrix are simply

$$D_{ij} = \dot{\phi}_j(t_i).$$

It is possible to deduce, through careful differentiation, that

$$\dot{\phi}_j(t_i) = \prod_{\substack{k=1 \\ k \neq j}}^{n_n} \frac{1}{t_j - t_k} \sum_{\substack{k=1 \\ k \neq j}}^{n_n} \left[\prod_{\substack{l=1 \\ l \neq k,j}}^{n_n} (t_i - t_l) \right]. \tag{2.28}$$

Further simplifications of Equation 2.28 are possible when assumptions are made as to the spacing of the nodes. Two common spacings are traditionally considered for collocation methods. Uniform spacing is assumed in all of the methods discussed thus far, but the pseudospectral method is generally implemented with nodes distributed at the Legendre-Gauss-Lobatto (LGL) points. The matrix definitions for both are presented next.

With uniform spacing, relations can be drawn on the differences $t_j - t_k$ and $t_i - t_l$ in Equation 2.28 in terms of the node space, $h = t_{j+1} - t_j$. This leads to the uniform pseudospectral differentiation matrix, whose elements are defined as

$$D_{ij} = \begin{cases} \dfrac{(-1)^{j-i}(n_n - i)!(i-1)!}{h(i-j)(n_n-j)!(j-1)!}, & i \neq j, \\ \dfrac{1}{h}\displaystyle\sum_{\substack{k=1 \\ k \neq i}}^{n_n} \dfrac{1}{i-k}, & i = j. \end{cases}$$

Alternatively, the spacing of the nodes may be at the LGL points, and the justification for this becomes apparent in the example presented in Section 2.4. A complete derivation of the Legendre pseudospectral differentiation matrix is not included in this work, although interested readers may look in several different sources.[34–37] Within the present scope, a familiar knowledge of Legendre polynomials, $P_N(t)$, is sufficient. These polynomials represent an orthogonal set on the interval $t^* \in [-1\ 1]$. The polynomial $P_N(t)$ is an Nth order polynomial with N roots within the interval. The LGL points that define the node spacing here are the roots of the derivative polynomial, $\dot{P}_{n_n-1}(t)$, along with the endpoints -1 and 1. As usual, this contributes a total of n_n nodes. Let the LGL points be identified as $t_j^* \in [-1\ 1]$. With this, the pseudospectral differentiation matrix can be defined.

$$D_{ij} = \begin{cases} \left(\dfrac{2}{t_f - t_0}\right) \dfrac{P_{n_n-1}(t_i^*)}{(t_i^* - t_j^*)P_{n_n-1}(t_j^*)}, & i \neq j \\ -\left(\dfrac{2}{t_f - t_0}\right) \dfrac{n_n(n_n-1)}{4}, & i = j = 1 \\ \left(\dfrac{2}{t_f - t_0}\right) \dfrac{n_n(n_n-1)}{4}, & i = j = n_n \\ 0, & \text{otherwise} \end{cases} \tag{2.29}$$

Observe that, in Equation 2.29, each term is scaled by $2/(t_f - t_0)$. While the spacing of the LGL points goes from -1 to 1, the optimal control problem remains on the interval $[t_0\ t_f]$. The differentiation must account for the actual interval of the trajectory. If this is inconvenient, then all of the times could be scaled to the Legendre interval. In the case here, where the differentiation matrix is scaled to the true interval, the constraint equation,

$$\boldsymbol{C} = \boldsymbol{D}\bar{\boldsymbol{y}} - \bar{\boldsymbol{f}},$$

considers the actual times, $t_j \in [t_0 \; t_f]$, when evaluating the function. The relation between the actual and the LGL times is simply

$$t_j = t_0 + (t_f - t_0)\frac{t_j^* + 1}{2}.$$

2.3.2.3 Characteristics of Collocation Methods

The major disadvantage of a collocation method is the relatively large number of parameters necessary in defining the transcription. This naturally has an effect on the speed of convergence, as more functions and function derivatives must be evaluated or approximated on each iteration. However, this effect can be minimized by using a non-linear programming algorithm that identifies the sparsity patterns and efficiently evaluates functions and function derivatives.

As always, convergence is dependent upon the initial guess. In this case, a larger number of parameters allows the user to generate a guess from insight into either the states, the controls, or both. An important characteristic of this parameterization is that the states and controls do not need to be consistent for an initial guess. That is, the values of the guessed states and controls need not immediately satisfy the governing differential equations. For example, if a user has some idea of the shape of the trajectory, it can be reflected in the state-parameters, and the guess for the control-parameters can be completely independent. The resulting initial guess, in this case, does not represent a feasible solution, but the optimizer is free to roam the solution space to find a feasible and locally optimal solution. The flexibility gained in using a large number of parameters improves the likelihood for convergence.

Additionally, a large number of parameters can potentially guide an optimizer to a solution. However, if a set of parameters represents a feasible or near-feasible solution, it may be difficult for the optimizer to move away from the current point towards a more optimal solution.

2.3.2.4 Verifying the Optimality of Collocation Solutions

Thus far, the methods presented are based on the assumption that the numerical solution to the optimal control problem adequately represents the optimal analytic solution. However, it is still important to provide adequate proof that the resulting arcs are truly optimal, at least locally. To that end, the methodology presented here offers a means of verifying the optimality of the numerically determined solutions.

In the transcription of the optimal control problem, a constrained parameter optimization problem results with n parameters, $\boldsymbol{x}$, and n_c constraint equations, $\boldsymbol{c}(\boldsymbol{x})$. Recall that the transcribed problem, with n_n nodes, can be expressed in terms of a cost function and constraints defined by

$$\begin{aligned} \mathfrak{J} &= F(\boldsymbol{x}), \qquad (2.9) \\ \mathbf{0} &= \boldsymbol{c}(\boldsymbol{x}), \end{aligned}$$

where the numbers of parameters, n, and constraints, n_c, are determined as,

$$\begin{aligned} n &= (n_y + n_u)n_n + 2, \\ n_c &= n_{\psi_0} + n_{\psi_f} + n_s(n_n - 1). \end{aligned}$$

The traditional solution to the transcribed problem involves an augmented cost index that is subject to a different set of Lagrange multipliers, $\boldsymbol{\eta}$, of dimension n_c,

$$\tilde{\mathfrak{J}} = F(\boldsymbol{x}) + \boldsymbol{\eta}^T \boldsymbol{c}(\boldsymbol{x})$$

The vector of constraints, $\boldsymbol{c}$, consists of n_{ψ_0} equations that correspond to $\boldsymbol{\psi}_0$, n_{ψ_f} equations that correspond to $\boldsymbol{\psi}_f$, $n_\beta n_n$ equations that correspond to β, and $n_{\dot{y}} = n_y(n_n - 1)$ or $n_{\dot{y}} = n_y n_n$ equations that parameterize the differential conditions, $\dot{\boldsymbol{y}} = \boldsymbol{f}$. Likewise, the vector of Lagrange multipliers, $\boldsymbol{\eta}$, has n_{ψ_0} values that correspond to $\boldsymbol{\nu}_0$, n_{ψ_f} values that correspond to $\boldsymbol{\nu}_f$, $n_\beta n_n$ values that correspond to $\boldsymbol{\mu}$, and $n_{\dot{y}}$ values that correspond to $\boldsymbol{\lambda}$ at or in-between the nodes.

Thus, the Lagrange multipliers of the parameter optimization problem correspond directly to the Lagrange multipliers of the optimal control problem.[35] For $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_f$, corresponding values of $\boldsymbol{\eta}$ will match identically. Since the path and dynamical constraints are satisfied at the nodes (for path constraints and spectral continuity constraints) or between the nodes (for integration continuity constraints), then values of $\boldsymbol{\eta}$ will match with corresponding values of $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ either at or in-between the nodes.

Many numerical optimization packages allow the user to extract the Lagrange multipliers along with final values of the parameters. Thus, the Lagrange multipliers, $\boldsymbol{\eta}$, are available to the user along with the solution. After a numerical algorithm has converged, all the states, controls, times, and Lagrange multipliers are known at each node. With direct methods, the conditions of Equations 2.7 and 2.8 are not directly applied. However, they can be used *after* a solution is determined to validate it. A user may verify that the conditions are satisfied with the known parameters and multipliers. Simply, the states and costates can be integrated from their initial values and time (which are known, at a minimum, through the values of the Lagrange multipliers), to their final time to ensure that the optimality conditions are satisfied.

2.3.2.5 Using Direct Solutions to Supplement an Indirect Method

While the accuracy of a solution from a direct optimization scheme may be inferior to an indirect solution, the above observation lays a path for finding a solution with an indirect shooting method. For example, if a particular optimal control problem is using collocation, then Lagrange multipliers are available that offer a *very good* guess to the actual (continuous problem) Lagrange multipliers. This is a major benefit of collocation methods, as the literature commonly refers to the difficulty in guessing the initial values of the differential Lagrange multipliers for an indirect method.

Also, it is observed that a direct shooting scheme does not offer the same advantage of providing insight into the values of the costates. The scheme does not require the states to be carried as parameters to be optimized, and consequently, the constraint equations to be satisfied in the parameterized problem do not correlate the same way to the initial, final, and differential conditions of the optimal control problem. Although the implementation of direct shooting is less cumbersome, the ability to verify optimality and/or provide insight for indirect optimization generally offsets the added complexity of the collocation formulation.

2.4 Example: The Zermelo Navigation Problem

A straightforward approach to comparing some of the methods discussed above is through example. In this section, a classical navigation problem is investigated using several direct and indirect methods.

2.4.1 Problem Statement and Description

Consider a boat moving across a stream of water from one dock to another, as in Figure 2.3. The stream is 10 meters wide, and the landing dock is 10 meters upstream from the departing dock. The stream has a current described by,

$$W(\boldsymbol{y}) = 1 + \sin(y_1),$$

and the boat moves at a constant velocity of $V = 2$ m/s. The objective is to move the boat from one dock to the other in the minimum possible amount of time by controlling

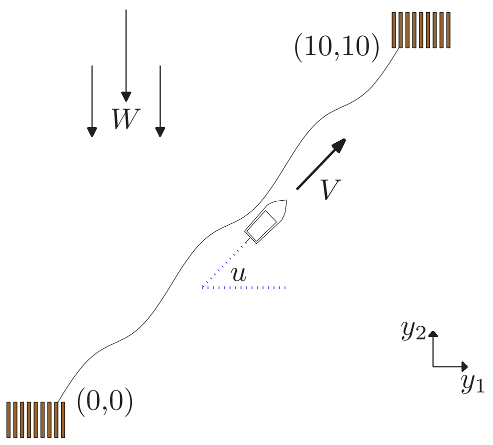


Figure 2.3: The Zermelo Navigation Problem

the boat's pointing angle, $u(t)$. Mathematically, the objective is to minimize

$$\mathcal{J} = t_f,$$

subject to

$$\begin{aligned}
\dot{\boldsymbol{y}} &= \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} V\cos u \\ V\sin u - W(\boldsymbol{y}) \end{bmatrix}, \\
\mathbf{0} &= \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) = \begin{bmatrix} t_0 - t_0^* \\ y_1(t_0) - y_1^*(t_0) \\ y_2(t_0) - y_2^*(t_0) \end{bmatrix}, \\
\mathbf{0} &= \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f) = \begin{bmatrix} y_1(t_f) - y_1^*(t_f) \\ y_2(t_f) - y_2^*(t_f) \end{bmatrix}.
\end{aligned}$$

2.4.2 Direct Approach

To obtain a direct solution to the optimal control problem, consider the collocation parameterization of Equation 2.14. Let the number of nodes be selected as $n_n = 20$. With

$n_y = 2$ states and $n_u = 1$ control, the size of the problem is

$$n = (n_y + n_u)n_n + 2 = 62,$$

with parameters

$$\begin{array}{rclccccccccccl} \boldsymbol{x} & = & [& x_1 & x_2 & \cdots & x_{39} & x_{40} & x_{41} & \cdots & x_{60} & x_{61} & x_{62} &]^T \\ & = & [& y_1(t_1) & y_2(t_1) & \cdots & y_1(t_{n_n}) & y_2(t_{n_n}) & u(t_1) & \cdots & u(t_{n_n}) & t_0 & t_f &]^T. \end{array}$$

The times, t_j, vary based on the node spacing of the method, but with both uniform and Legendre node spacing, it is certain that $t_1 = t_0$ and $t_{n_n} = t_f$.

The initial and final constraints, according to this parameterization, become

$$\begin{aligned} \boldsymbol{c}_{\psi_0}(\boldsymbol{x}) &= \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) = \begin{bmatrix} t_0 - t_0^* \\ y_1(t_0) - y_1^*(t_0) \\ y_2(t_0) - y_2^*(t_0) \end{bmatrix} = \begin{bmatrix} x_{61} - 0 \\ x_1 - 0 \\ x_2 - 0 \end{bmatrix} \\ \boldsymbol{c}_{\psi_f}(\boldsymbol{x}) &= \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f) = \begin{bmatrix} y_1(t_f) - y_1^*(t_f) \\ y_2(t_f) - y_1^*(t_f) \end{bmatrix} = \begin{bmatrix} x_{39} - 10 \\ x_{40} - 10 \end{bmatrix}. \end{aligned}$$

Without path constraints, $\boldsymbol{c}_\beta = [\,]$.

The simplest scheme for generating the continuity constraints is explicit Euler integration. In this case,

$$\begin{aligned} \boldsymbol{c}_{\dot{y}}(\boldsymbol{x}) &= \begin{bmatrix} y_1(t_2) - y_1(t_1) - h\{V \cos u(t_1)\} \\ y_2(t_2) - y_2(t_1) - h\{V \sin u(t_1) - 1 - \sin y_1(t_1)\} \\ \vdots \\ y_1(t_{n_n}) - y_1(t_{n_n-1}) - h\{V \cos u(t_{n_n-1})\} \\ y_2(t_{n_n}) - y_2(t_{n_n-1}) - h\{V \sin u(t_{n_n-1}) - 1 - \sin y_1(t_{n_n-1})\}) \end{bmatrix} \\ &= \begin{bmatrix} x_3 - x_1 - h\{V \cos x_{41}\} \\ x_4 - x_2 - h\{V \sin x_{41} - 1 - \sin x_1\} \\ \vdots \\ x_{39} - x_{37} - h\{V \cos x_{59})\} \\ x_{40} - x_{38} - h\{V \sin x_{59} - 1 - \sin x_{37}\} \end{bmatrix}. \end{aligned}$$

Assuming uniform spacing, the distance between two nodes, h, is defined as

$$h = \frac{t_f - t_0}{n_n - 1} = \frac{x_{62} - x_{61}}{n_n - 1}.$$

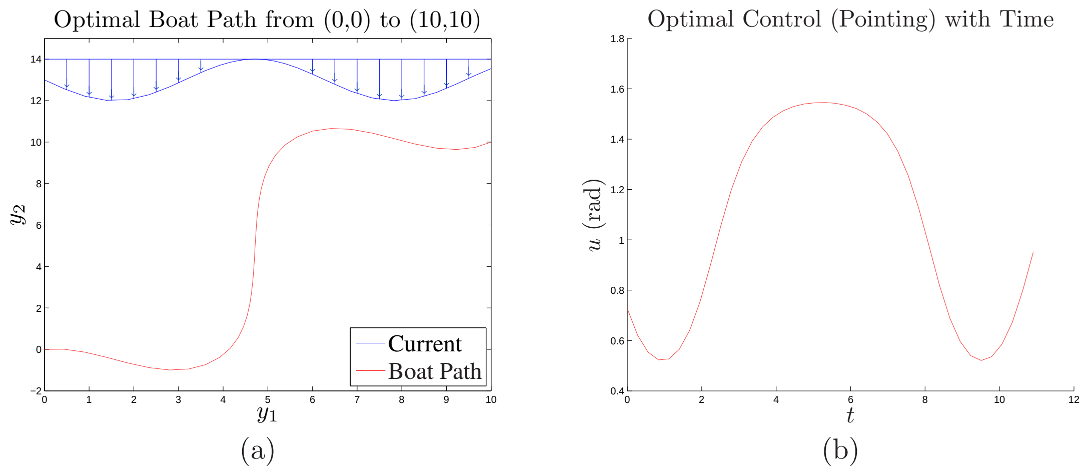


Figure 2.4: Optimal Path (a) and Control (b) for the Zermelo Problem

Finally, the cost function is simply,

$$F(\boldsymbol{x}) = t_f = x_{62}.$$

The parameter optimization problem is completely defined. Of course, modifications are required to the constraints $\boldsymbol{c}_{\dot{y}}(\boldsymbol{x})$ to accommodate the other continuity schemes discussed. These are not expanded here for the sake of brevity, but the associated framework was presented in earlier sections.

Comparison of Direct Solutions

The optimal trajectory for the Zermelo boat is illustrated in Figure 2.4. Displayed is the path of the boat from the initial point, (0,0), to the final point, (10,10), and the control history over the course of the trajectory. One may investigate how well the continuity schemes for each of the direct methods presented here perform in duplicating this solution.

Each scheme is applied with $n_n = 20$, as developed previously, and also with $n_n = 40$. Intuitively, it is expected that with more nodes, the solution improves, perhaps at the cost of speed in convergence time. It is also expected that higher order methods of the same class perform better (Explicit Trapezoidal integration should outperform Explicit Euler; Implicit Hermite-Simpson integration should outperform Implicit Trapezoidal). A comparison in performance of each scheme is displayed in Table 2.1. For both sets of nodes, the table describes whether the method was successful (i.e. the solution resembles that of Figure 2.4), the cost (i.e. the final time t_f), and the number of iterations incurred.

For each trial, the initial guess places the Zermelo boat along a direct path (straight line) from the initial point to the final point over a duration of 10 seconds. The guessed control is simply $u(t_j) = 1$ for all nodes $j = 1, ..., n_n$. Identical guesses are used for each trial for fair comparison.

With the exception of the uniform pseudospectral method, all of the methods perform reasonably well, with different levels of accuracy. That is, all but the uniform pseudospectral method converge on a reasonable solution. With 20 nodes, the uniform pseudospectral algorithm is able to converge, however, it does not lead to a solution that is near a feasible trajectory (even though it may have satisfied the given constraint equations). With 40 nodes, it is unable to perform any improvements at all, quitting before the first iteration.

The number of iterations per method seem to be relatively consistent with all of the

Table 2.1: Direct Solutions with 20 Nodes and 40 Nodes

	20 nodes			40 nodes		
Method	Success?	t_f	Iterations	Success?	t_f	Iterations
Explicit Euler	Yes	10.9245	29	Yes	10.9301	31
Explicit Trapezoidal	Yes	10.8065	27	Yes	10.8796	28
Implicit Trapezoidal	Yes	10.8994	27	Yes	10.9043	27
Implicit Simpson	Yes	10.9046	26	Yes	10.9045	26
4th Order Spectral	Yes	11.4427	868	Yes	11.1849	25
Uniform Pseudospectral	No	106.2257	43	No	0	—
Legendre Pseudospectral	Yes	10.9045	35	Yes	10.9126	35
Actual Solution	—	10.9045				

methods, regardless of the number of nodes. The fourth order spectral method is the only outlier, requiring many more iterations to converge in the case of 20 nodes. For the majority of those iterations, the set of parameters hovered near the optimal point, making extremely small steps towards the minimum. It is assumed that along the path to the minimum, the parameter set landed on a flat gradient, requiring more iterations to achieve convergence. Note that this was not the case with more nodes.

It is generally expected that a superior method, with an increased number of nodes (i.e. 40 vs. 20), results in a more accurate solution. This is verified in the cases involving implicit integration schemes, the explicit trapezoidal scheme, and the fourth order spectral scheme. However, the explicit Euler integration and the Legendre pseudospectral methods actually exhibit performance degradation with the finer grid.

The implicit Hermite-Simpson method delivers the best results of this trial set. The 20-node solution is extremely close to the actual solution, and the 40-node solution is the closest to the analytical solution. For a relatively similar number of iterations, the Hermite-Simpson method produces superior results.

2.4.3 Indirect Approach

Once again, consider the Zermelo problem and the associated parameters outlined in the previous section. Using Equation 2.2 and 2.3, the Bolza Function and the Hamiltonian are

$$\begin{aligned} G &= t_f + \nu_{0_1} t_0 + \nu_{0_2} y_1(t_0) + \nu_{0_3} y_2(t_0) + \nu_{f_1}\{y_1(t_f) - 10\} + \nu_{f_2}\{y_2(t_f) - 10\}, \\ H &= \lambda_1\{V\cos u\} + \lambda_2\{V\sin u - (1+\sin y_1)\}. \end{aligned} \tag{2.30}$$

With the Euler-Lagrange conditions of Equations 2.7, the conditions of optimality are given by

$$\begin{array}{rcl} \left\{ \begin{array}{l} \dot{y}_1 \\ \dot{y}_2 \end{array} \right. & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} V\cos u \\ V\sin u - 1 - \sin y_1 \end{array} \\ \left\{ \begin{array}{l} \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{array} \right. & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} \lambda_2 \cos y_1 \\ 0 \end{array} \\ 0 & = & -\lambda_1 \left\{ V\sin u \right\} + \lambda_2 \left\{ V\cos u \right\} \end{array}$$

with boundary conditions according to Equations 2.8:

$$\begin{array}{rcl} H(t_0) & = & \nu_{0_1}, \\ H(t_f) & = & -1, \\ \left\{ \begin{array}{l} \lambda_1(t_0) \\ \lambda_2(t_0) \end{array} \right. & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} -\nu_{0_2}, \\ -\nu_{0_3}, \end{array} \\ \left\{ \begin{array}{l} \lambda_1(t_f) \\ \lambda_2(t_f) \end{array} \right. & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} \nu_{f_1}, \\ \nu_{f_2}. \end{array} \end{array}$$

Observe that, since the Hamiltonian does not explicitly depend on time, $\dot{H} = 0$. Also, since $\dot{\lambda}_2 = 0$, it is evident that $\nu_{0_1} = -1$ and $\lambda_2 = -\nu_{0_3} = \nu_{f_2}$. In addition, the optimal control law is determined as

$$\tan u = \frac{\lambda_2}{\lambda_1}. \tag{2.31}$$

Equation 2.31, by itself, does not uniquely describe the control, since $\tan u = \tan(\pi + u)$. However, consider the two triangles in Figure 2.5. In both configurations illustrated, Equation 2.31 is satisfied, and the hypotenuse of the triangle is

$$\lambda_3 = \sqrt{\lambda_1^2 + \lambda_2^2}.$$

In order to determine which is correct, observe that

$$[\cos u, \sin u] = \left\{ \begin{array}{l} \left[\ \frac{\lambda_1}{\lambda_3}, \ \ \frac{\lambda_2}{\lambda_3} \right] \\ \\ \left[-\frac{\lambda_1}{\lambda_3}, -\frac{\lambda_2}{\lambda_3} \right] \end{array} \right. .$$

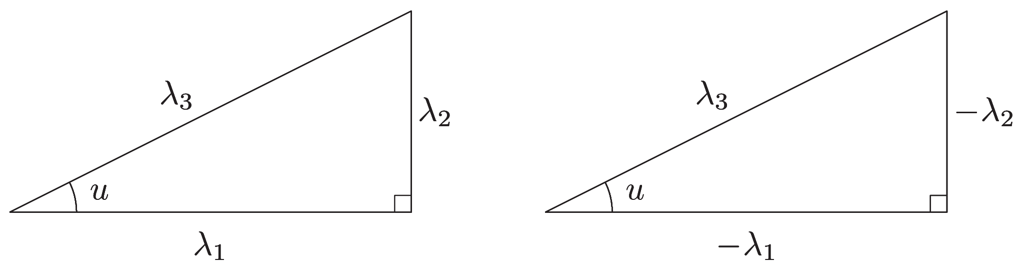


Figure 2.5: Geometric Interpretations of Equation 2.31

In Equation 2.30, the Hamiltonian is minimized when $[\cos u, \sin u] = [-\lambda_1/\lambda_3, -\lambda_2/\lambda_3]$. Using this relation, the control input can be completely removed from the dynamical equations for the states and costates. The result is a two-point boundary value problem that can be solved numerically.

Define the augmented states, $\boldsymbol{z} = \left[\boldsymbol{y}^T\ \boldsymbol{\lambda}^T\right]^T$. The dynamics of the augmented states becomes

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{bmatrix} = \begin{bmatrix} -V\frac{\lambda_1}{\lambda_3} \\ -V\frac{\lambda_2}{\lambda_3} - 1 - \sin y_1 \\ \lambda_2 \cos y_1 \\ 0 \end{bmatrix},$$

where u has been replaced by its relations with the costates. The known initial conditions for $\boldsymbol{z}$ are $y_1(0) = 0$ and $y_2(0) = 0$, and the initial conditions for $\lambda_1(0)$ and $\lambda_2(0)$ must be guessed. In addition, the final desired states are $y_1^*(t_f) = 10$ and $y_2^*(t_f) = 10$, but the actual final time, t_f, is unknown. Define the shooting function such that,

$$\begin{bmatrix} \boldsymbol{y}(t_f) \\ \boldsymbol{\lambda}(t_f) \end{bmatrix} = \boldsymbol{g}(\lambda_1(0), \lambda_2(0), t_f),$$

which integrates the augmented states (with known initial states and guessed initial costates) to the guessed final time. Finally, define the parameter vector as

$$\boldsymbol{x} = [\lambda_1(0)\ \lambda_2(0)\ t_f]^T .$$

Then, the indirect problem is completely solved when the following system of constraints is satisfied.

$$\boldsymbol{c}(\boldsymbol{x}) = \begin{bmatrix} y_1(t_f) - y_1^*(t_f) \\ y_2(t_f) - y_2^*(t_f) \\ 1 + H(t_f) \end{bmatrix} = \begin{bmatrix} g_1(\boldsymbol{x}) - y_1^*(t_f) \\ g_2(\boldsymbol{x}) - y_2^*(t_f) \\ 1 - V\sqrt{g_3(\boldsymbol{x})^2 + g_4(\boldsymbol{x})^2} - g_4(\boldsymbol{x})(1 + \sin g_2(\boldsymbol{x})) \end{bmatrix} = \mathbf{0}$$

A converged solution has $x_3 = t_f = 10.9045$. While it may be intuitive to guess a final time in this vicinity, guessing the values of the initial costates is difficult. In this example, arbitrary guesses for the initial states do not generally result in convergence along the correct path (although one solution was found with a negative final time!). However, having already solved the problem using the direct method, approximate values of the Lagrange multipliers are already known. Submitting as an initial guess the approximations yields a successful result.

2.5 Summary

In this chapter, numerical methods for solving optimal control problems, and some of their associated characteristics, are discussed. Among them are the accuracy of the solution, the speed of convergence, and the robustness of the solution to the initial guess.

The most accurate solutions are obtained through indirect methods. These are accurate to the order of the numerical integrator used to integrate initial values of the states and costates. Among the direct methods featured in this development, the most accurate and consistent results are obtained through use of the Hermite-Simpson integration equations. With fewer nodes, the solution is still accurate, but the solution naturally

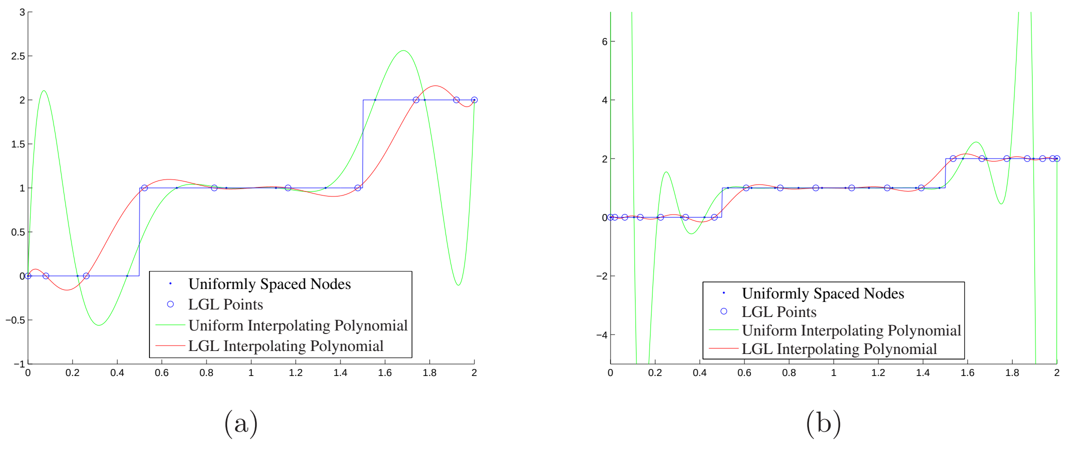


Figure 2.6: Uniformly Spaced and LGL Spaced Interpolating Functions for $r(t)$ with 10 nodes (a) and 20 nodes (b)

improves when the fidelity of the discretized model is enhanced (by increasing the number of nodes).

It is interesting to observe why the uniform pseudospectral method failed. In both pseudospectral methods, an interpolating polynomial is fit to the node points, wherever they are distributed. A simple way of demonstrating the reason for the failure is with an example. Consider the function, $r(t)$, which simply rounds the argument to the nearest integer value. Although a simple function, it is clearly one difficult to fit with a polynomial. In Figure 2.6, the interpolating polynomials are shown with both 10 and 20 nodes using both uniform spacing and LGL-point spacing. With uniform spacing, it is evident that the interpolating polynomial experiences difficulties near the end points, while the polynomial derived from LGL-point spacing is better behaved. Recalling that the pseudospectral methods attempt to match the derivatives of the interpolating polynomial to the dynamic function, it is evident that the uniform pseudospectral method performs poorly because it attempts to match derivatives to a poor approximation of the actual states.

The speed of convergence could be measured in terms of either the number of iterations or in actual computer time. If measuring in terms of iterations, one must consider the computer time necessary *per iteration*. This depends on the number of functions to be evaluated, (a potential factor associated with the number of nodes) and the complexity of those functions. In the example presented in Section 2.4, the number of parameters and the complexity of the functions evaluated are relatively low for each of the direct and indirect methods, so it is difficult to get a sense of comparison for the speed of each of the algorithms based on this example alone. In general, it is safe to assume that direct methods take longer per iteration, although they may not take as many iterations as the indirect method.

The most important factor in this investigation is the robustness to the initial guess. With all locally minimizing algorithms, one can generally only expect to find a minimum in the vicinity of the initial guess. However, how close to that local minimum does the initial guess have to be? The indirect methods are far less robust in this respect. Small variations in a guess of the unknown initial values of the Lagrange multipliers may have dramatic effects on the final values of the states and costates. Without a good initial guess, it may be impossible to find a solution with an indirect method.

Additionally, it is desirable to develop a methodology that may be applied easily to optimal control problems characterized by many different forms of boundary conditions and dynamical constraints. Since the form of the constraints for indirect optimization may vary quite significantly, an indirect method is an unlikely candidate for this task.

The desire in this investigation is to determine a stable way of examining many optimal solutions for many different problems. To do this, it is concluded that a direct collocation method offers the highest probability of success. Among the direct methods, the most accurate and stable method in this chapter evaluates continuity constraints via the Hermite-Simpson equations. Consequently, it is this method that is used throughout the remainder of this work.

Chapter 3

Transcription Formulations for the Finite Set Control Problem

In Chapter 2, the basic transcription formulations for direct methods are developed. The optimal control problem is parameterized into values of the states and controls at nodes, which are contained in the parameter vector, $\boldsymbol{x}$, along with the initial time and the final time. Methods are demonstrated for generating the optimization function and the constraint functions in terms of $\boldsymbol{x}$. The Hermite-Simpson integration scheme is selected as a stable and accurate form for continuity constraints. In this chapter, modifications to the traditional collocation formulation of the optimal control problem are developed to treat the hybrid control problem as a nonlinear programming problem. Parameterization variations, derivative evaluations, and unique implementation issues are discussed.

In the development, a primary consideration for discrete variables is that discontinuities necessarily exist when switching from one value to another. Consequently, the general method is first altered to divide a problem into multiple segments, allowing for ideal treatment of variable discontinuities. Next, the formulation is expanded further to account for independent switching between multiple control variables. In the process of accounting for finite set control conditions, control variables are ultimately eliminated from the parameterization. Instead, state parameters and switching time parameters are optimized over a continuous spectrum to yield the desired control history.

The implementation of the Finite Set Control Transcription is subsequently explored in detail. A chief consideration is the evaluation of up to $(n_c+1)n$ partial derivatives describing the gradients of the objective function and each constraint with respect to each variable in $\boldsymbol{x}$. The numerical techniques clearly rely on these derivatives, stored in the Jacobian matrix, for choosing a search direction. The FSCT method exhibits unique characteristics in the behavior of some derivative elements, and their effect on the solution process is addressed.

3.1 Unique Formulation Characteristics

In one sense, the hybrid control problem can be approached as a constrained continuous control problem. For example, the discrete control, $u_i \in \mathbb{U}_i = \{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i}\}$ as in Equation 1.3, can equivalently be treated as the continuous control, $u \in \mathbb{R}$, subject to the path constraint,

$$\beta_i = (u_i - \tilde{u}_{i,1}))(\cdots)(u_i - \tilde{u}_{i,m_i}) = 0.$$

With this in mind, the FSCT method is developed by determining effective ways to treat continuous control problems with these unique constraints.

The foundation of this approach is the general collocation method presented in Chapter 2. The method is altered to divide a problem into multiple segments, where the control discontinuities associated with finite set control are allowed at the segment boundaries. The formulation is subsequently expanded to account for multiple independent control variables. This ultimately requires dynamic changes in the time dependencies of each segment, but it is this modification that leads to feasible control solutions.

3.1.1 Multiple Segments and Knots

Consider the parameterization defined in Chapter 2.

$$\boldsymbol{x} = \left[y_1(t_0)\ \cdots\ y_i(t_j)\ \cdots\ y_{n_y}(t_f)\ u_1(t_0)\ \cdots\ u_i(t_j)\ \cdots\ u_{n_u}(t_f)\ t_0\ t_f\right]^T \tag{2.14}$$

Recall that i is designated to count members of a state or control vector, while j counts the nodes. A by-product of the Hermite-Simpson integration scheme is that the nodes along the solution are spaced evenly between $[t_0\ t_f]$, and the time associated with a given node is assigned based on

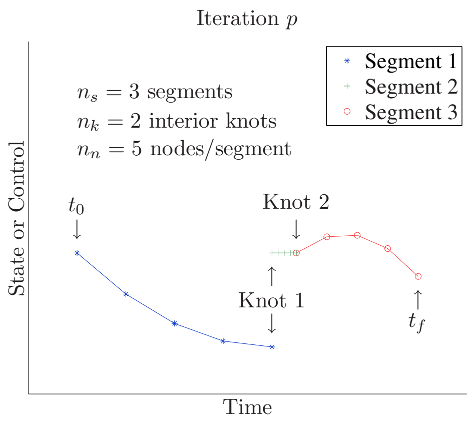
$$t_j = t_0 + \frac{j-1}{n_n - 1}\left(t_f - t_0\right). \tag{2.13}$$

Thus, an optimal solution contains state and control values at the nodes, and the continuous state and control histories must be interpolated in between the nodes. Of course, a constraint can be imposed anywhere along the path, not necessarily at a point that coincides with a node. Consider, for example, a constraint associated with $t = 4.2$. If $n_n = 10$, $t_0 = 1$ and $t_f = 10$, then $t_j = j$, and the time at each node is an integer value. Clearly, the parameterization considered thus far is not equipped to handle this constraint. The ability to incorporate constraints at any point along a trajectory is necessary in many applications.

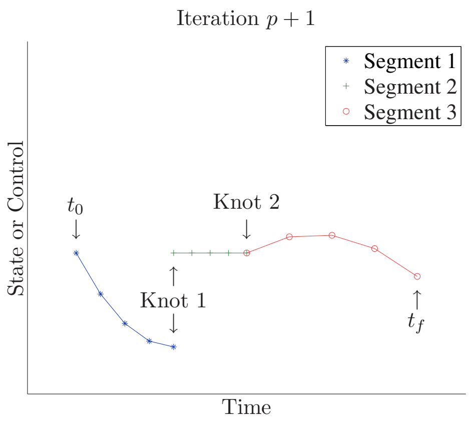
Specifically, state or control discontinuities cannot be effectively represented in the above parameterization, since they require intermediate constraints. For example, an orbital transfer that employs impulsive maneuvers requires some condition $\boldsymbol{v}^+ - \boldsymbol{v}^- = \Delta\boldsymbol{v}$. A staged rocket experiences mass discontinuities between stages. In each of these cases, the dynamics are impacted by these discontinuities.

Moreover, when the control space is limited to a finite set of values, there are necessarily control discontinuities when switching between values. It is not sufficient to represent a discontinuity by exhibiting $u(t_j)$ at one feasible control value and $u(t_{j+1})$ at another. The parameterized solution does not effectively communicate the time between t_j and t_{j+1} at which the discontinuity occurs, nor do the state equations account for the control discontinuity in the evaluation of state continuity constraints.

Thus motivated, the parameterization above is modified to account for discontinuities (in either states or controls). To facilitate the enhancement, knots are introduced into the parameterization. A *knot* represents the point at which a discontinuity might occur,



(a)



(b)

Figure 3.1: An Example of Segments and Knots at Two Consecutive Iterations

and it divides the trajectory into separate subarcs, or *segments*. The knots may be either fixed or free in time. When knots are free, the times at which they occur are optimized. Demonstrated presently, knots represent an effective means of incorporating control discontinuities. In addition, knots may divide a trajectory into different segments to accommodate changes in the dynamic model along the path.

Figure 3.1 illustrates the conceptual relation between knots, nodes, and segments at two consecutive iterations. Consider a formulation with n_s segments. The start of each segment constitutes a knot, with the exception of the first segment along the trajectory, where t_0 defines the start of the segment. Similarly, every segment ends at a knot with the exception of the last segment along the solution where t_f defines the end of the segment. Thus, the formulation consists of $n_k = n_s - 1$ knots.

Let every segment consist of n_n nodes. Although, conceptually, each segment may consist of a different number of nodes, the transcription is simplified by assuming an equal number of nodes per segment. Furthermore, while the nodes are uniformly spaced within a segment, they are not necessarily uniformly spaced along the course of the trajectory. That is, each segment may span a different length of time. In most cases, it is desirable for the segment duration (or the location of the knots) to be optimized as well. In Figure 3.1, the knot times change between iteration p and iteration $p+1$ as they are optimized by the NLP algorithm. Consequently, assigned times for nodes change respectively to maintain uniform spacing per segment.

The easiest implementation of a multiple-segment formulation includes the terminal times of each segment (that is, t_0, the interior knots, and t_f) in the parameter vector. The interior knots contribute n_k parameters, while t_0 and t_f are included in $\boldsymbol{x}$ as before. The parameterization described has n_s segments, each with n_n nodes. Each node is associated with n_y states and n_u controls. The parameter vector is defined as,

$$\boldsymbol{x} = \left[\cdots\ \boldsymbol{y}_{j,k}^T\ \cdots\ \cdots\ \boldsymbol{u}_{j,k}^T\ \cdots\ \cdots\ t_k\ \cdots\ t_0\ t_f\right]^T. \tag{3.1}$$

In Equation 3.1, $\boldsymbol{y}_{j,k} = \boldsymbol{y}(t_{j,k})$ and $\boldsymbol{u}_{j,k} = \boldsymbol{u}(t_{j,k})$ represent the state and control vectors (respectively) at the j^{th} node of the k^{th} segment. As a collection, these elements make up the arrays $\boldsymbol{Y}_{n_y,n_n,n_s}$ and $\boldsymbol{U}_{n_u,n_n,n_s}$, as summarized in Table 3.1. The node times of $\boldsymbol{T}_{n_n,n_s}$ are

$$t_{j,k} = t_{k-1} + \frac{j-1}{n_n-1}(t_k - t_{k-1}) \quad j = 1, \ldots, n_n,\ k = 1, \ldots, n_s, \tag{3.2}$$

where $t_{n_s} = t_f$ and the elements of $\boldsymbol{T}_{n_k}$ $(t_k,\ k = 1, \ldots, n_k)$ are the times at which each interior knot occurs. Thus, i identifies members of a vector, j denotes the respective nodes,

Table 3.1: Nomenclature Summary: Multiple Segment Formulation

Array	Description	Dimension	Element
$\boldsymbol{Y}_{n_y,n_n,n_s}$	States by node	$n_y \times n_n \times n_s$	$\boldsymbol{y}_{j,k}$ or $y_{i,j,k}$
$\boldsymbol{U}_{n_u,n_n,n_s}$	Controls by node	$n_u \times n_n \times n_s$	$\boldsymbol{u}_{j,k}$ or $u_{i,j,k}$
$\boldsymbol{T}_{n_n,n_s}$	Node times	$n_n \times n_s$	$t_{j,k}$
$\boldsymbol{T}_{n_k}$	Knot times	n_k	t_k
$\Delta\boldsymbol{T}_{n_s}$	Segment durations	n_s	Δt_k
n_s	Number of segments	$n_k + 1$	

and k represents the relevant knot or segment. The parameters can also be listed as

$$\boldsymbol{x} \quad = \quad [\underbrace{\cdots\ y_{i,j,k}\ \cdots}_{n_y n_n n_s}\ \underbrace{\cdots\ u_{i,j,k}\ \cdots}_{n_u n_n n_s}\ \underbrace{\cdots\ t_k\ \cdots}_{n_k}\ \underbrace{t_0\ t_f}_{2}]^T$$

and the dimension of the parameter vector is $n = (n_y + n_u)n_n n_s + n_k + 2$. The number of variables for multiple segments is approximately n_s times the number of variables for a single segment. However, multiple segments also introduce additional constraints.

$$\boldsymbol{c}(\boldsymbol{x}) = \left[\boldsymbol{c}_{\psi_0}^T(\boldsymbol{x})\ \boldsymbol{c}_{\psi_f}^T(\boldsymbol{x})\ \boldsymbol{c}_{\beta}^T(\boldsymbol{x})\ \boldsymbol{c}_{\dot{y}}^T(\boldsymbol{x})\ \boldsymbol{c}_{s}^T(\boldsymbol{x})\ \tilde{\boldsymbol{c}}_{t}^T(\boldsymbol{x})\right]^T$$

Among these constraints, the initial conditions and final conditions remain unchanged from those presented in Chapter 2. The path constraints, $\boldsymbol{c}_\beta(\boldsymbol{x})$, still impose the n_β conditions of $\boldsymbol{\beta}$ at each of the nodes. That is, within a segment, there are $n_\beta n_n$ elements in $\boldsymbol{c}_\beta$. Since there may be path constraints along each segment, $\boldsymbol{c}_\beta$ becomes a vector of dimension $n_\beta n_n n_s$. Continuity constraints within a segment are similarly addressed using the Hermite-Simpson implicit integration scheme to impose the dynamical conditions. However, since there are n_s segments, $n_y(n_n - 1)n_s$ continuity constraints are included in $\boldsymbol{c}_{\dot{y}}$.

The next set of constraints, $\boldsymbol{c}_s$, establishes conditions between segments (at the knots). These may include intermediate point constraints or continuity conditions. Generally, the vector $\boldsymbol{c}_s$ is used to address state continuity between segments. At the knots, time continuity is assumed so that the last node of the k^{th} segment occurs at the same time as the first node of the $(k+1)^{\text{th}}$ segment. Thus, in the absence of state discontinuities, states are subject to equality constraints. The behavior illustrated at Knot 2 in Figure 3.1 is representative of a state continuity constraint at a knot. Thus,

$$\begin{aligned}\boldsymbol{c}_{s_k} &= \boldsymbol{y}_{1,k+1} - \boldsymbol{y}_{n_n,k}, \quad k = 1, \ldots, n_s - 1, \\ \boldsymbol{c}_s &= \left[\boldsymbol{c}_{s_1}^T\ \cdots\ \boldsymbol{c}_{s_{n_s-1}}^T\right]^T,\end{aligned}$$

and $\boldsymbol{c}_s$ is a vector of dimension $n_y(n_s - 1)$.

The last set of constraints are time inequality constraints of the form $\tilde{\boldsymbol{c}}_t(\boldsymbol{x}) \leq \mathbf{0}$. Although knots are allowed to float freely in time, their order must be chronological. If the k^{th} knot lags behind the $(k-1)^{\text{th}}$ knot, for example, the duration of the associated segment is negative, a nonsensical result. To prevent this behavior, let

$$\begin{aligned} \tilde{c}_{t_k} &= t_{k-1} - t_k, \quad k = 1, \dots, n_s, \\ \tilde{\boldsymbol{c}}_t &= \left[\tilde{c}_{t_1} \ \cdots \ \tilde{c}_{t_{n_s}}\right]^T, \end{aligned} \tag{3.3}$$

where, again $t_{n_s} = t_f$. Note that the variable segment durations produce n_s time conditions.

The total number of constraints, including initial conditions, final conditions, path constraints, dynamical constraints, knot continuity constraints, and time inequality constraints becomes $n_c = n_{\psi_0} + n_{\psi_f} + n_\beta n_n n_s + n_y(n_n - 1)n_s + n_y(n_s - 1) + n_s$. The vector $\boldsymbol{c}(\boldsymbol{x})$ now includes both equality and inequality constraints. In setting up the NLP problem, the user must exercise care to properly distinguish the two types of constraints.

A more convenient parameterization includes the duration of the segments, $\Delta \boldsymbol{T}_{n_s}$, in the parameter vector $\boldsymbol{x}$, instead of the knot times. Consider the parameter vector,

$$\boldsymbol{x} = [\underbrace{\cdots \ y_{i,j,k} \ \cdots}_{n_y n_n n_s} \ \underbrace{\cdots \ u_{i,j,k} \ \cdots}_{n_u n_n n_s} \ \underbrace{\cdots \ \Delta t_k \ \cdots}_{n_s} \ \underbrace{t_0 \ t_f}_{2}]^T.$$

The dimension of $\boldsymbol{x}$ is now $n = (n_y + n_u)\, n_n n_s + n_s + 2$, adding one dimension to the previously identified parameterization (n_s segment durations are needed instead of n_k knot times). This offers the advantage of converting the inequality constraints of $\tilde{\boldsymbol{c}}_t(\boldsymbol{x})$ into a single equality constraint. Let the knot times be defined as

$$t_k = t_0 + \sum_{\kappa=1}^{k} |\Delta t_\kappa| \, . \tag{3.4}$$

Then the inequality, $t_{k-1} - t_k \leq 0$, is guaranteed, and need not be included in the constraint vector, $\boldsymbol{c}(\boldsymbol{x})$. Instead $\tilde{\boldsymbol{c}}_t(\boldsymbol{x})$ is replaced by the single time equality constraint,

$$c_t(\boldsymbol{x}) = (t_f - t_0) - \sum_{k=1}^{n_s} |\Delta t_k| \,. \tag{3.5}$$

Now, the total constraint vector is simply

$$\boldsymbol{c}(\boldsymbol{x}) = \left[\boldsymbol{c}_{\psi_0}^T(\boldsymbol{x}) \; \boldsymbol{c}_{\psi_f}^T(\boldsymbol{x}) \; \boldsymbol{c}_{\beta}^T(\boldsymbol{x}) \; \boldsymbol{c}_{y}^T(\boldsymbol{x}) \; \boldsymbol{c}_{s}^T(\boldsymbol{x}) \; c_t(\boldsymbol{x}) \right]^T , \tag{3.6}$$

which includes $n_c = n_{\psi_0} + n_{\psi_f} + n_\beta n_n n_s + n_y(n_n - 1)n_s + n_y(n_s - 1) + 1$ equality constraints. An appealing aspect of this alternate formulation is that it is not necessary to track, during the solution process, which constraints are active. In this formulation, all constraints are active at all times. This formulation also enhances the convergence properties of the algorithm.

For instance, constraints may not necessarily be satisfied at a given iteration leading to convergence. The inequality constraints presented in Equation 3.3 are applied to avoid the occurrence of segments of negative duration. However, in the intermediate step leading to a solution, some segments may violate this constraint. Depending upon the specific nature of the problem, this may prevent the optimizer from converging on a solution. The formulation in Equation 3.5 ensures this situation does not arise.

Furthermore, $|\Delta t_k|$ can be replaced by Δt_k if a simple bound is placed on the variable to limit it to nonnegative values. In other words, the optimizer can be instructed not to search for a solution where Δt_k is negative. This has the same effect as the absolute values formulation.

An Application Requiring Multiple Segments

A statement of the Zermelo Navigation problem can be used to demonstrate the effectiveness of using a multiple-segment formulation. Specifically, it is demonstrated presently how this formulation may facilitate solutions when control discontinuities exist. The result is a more precise solution than the single-arc case.

Consider a formulation where the velocity of the boat is no longer constant. Now, the vector, $\boldsymbol{u}$, includes a variable control acceleration along with the control pointing direction. Consider two equivalent representations of the new dynamics:

$$\begin{aligned}
(1) \qquad \dot{\boldsymbol{y}}' &= \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}'_4 \end{bmatrix} = \begin{bmatrix} y_3 \\ y'_4 - (1 + \sin y_1) \\ u_1 \cos u_2 \\ u_1 \sin u_2 \end{bmatrix}, \\
\boldsymbol{y}'(t_0) &= \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
(2) \qquad \dot{\boldsymbol{y}} &= \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} y_3 \\ y_4 \\ u_1 \cos u_2 \\ u_1 \sin u_2 - y_3 \cos y_1 \end{bmatrix}, \\
\boldsymbol{y}(t_0) &= \begin{bmatrix} 0 & 0 & 0 & -1 \end{bmatrix}^T.
\end{aligned} \tag{3.7}$$

The form for $\boldsymbol{y}'$ most resembles the previous dynamics, while the form for $\boldsymbol{y}$ lets the third and fourth states represent velocities. The simple change of variables $y_4 = y'_4 - (1 + \sin y_1)$ gives the canonical form desired. In each case, the initial conditions place the starting dock at the origin, while the water current places the boat's initial velocity in the $-y_2$ direction.

The objective now is to move from one dock to the other in a specified amount of time while minimizing the control acceleration. That is, minimize

$$\mathfrak{J} = \int_{t_0}^{t_f} |u_1| \, dt,$$

subject to Equations 3.7 and boundary conditions

$$\mathbf{0} = \boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) = \begin{bmatrix} t_0 - 0 \\ y_1(t_0) - 0 \\ y_2(t_0) - 0 \\ y_3(t_0) - 0 \\ y_4(t_0) + 1 \end{bmatrix} \quad \text{and} \quad \mathbf{0} = \boldsymbol{\psi}_f(t_f, \boldsymbol{y}_f) = \begin{bmatrix} t_f - 10 \\ y_1(t_f) - 10 \\ y_2(t_f) - 10 \\ y_3(t_f) - 0 \\ y_4(t_f) - 0 \end{bmatrix}.$$

The final conditions specify the final time, $t_f^* = 10$, the final position at the second dock located at $(10, 10)$, and zero final velocity to ensure a smooth landing. In addition, assume

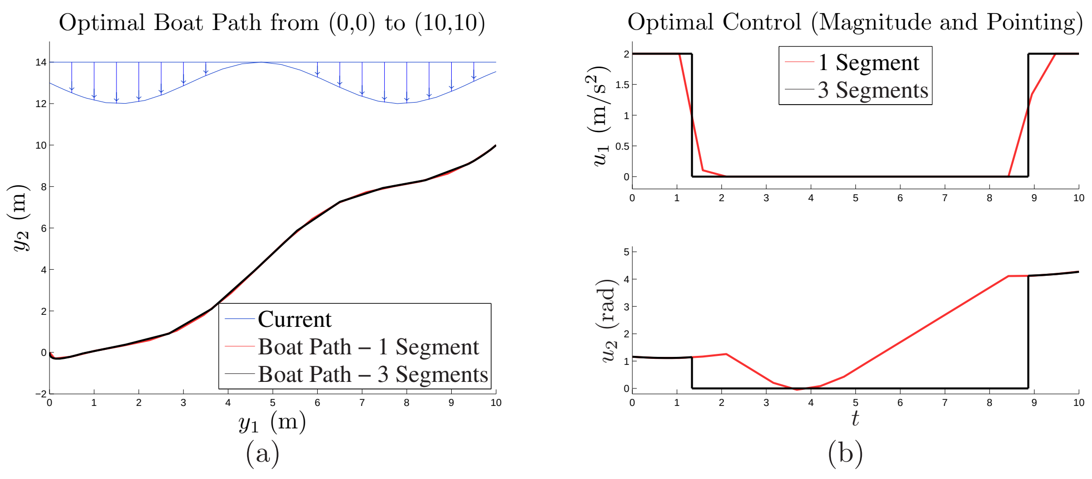


Figure 3.2: Optimal Path (a) and Control (b) for the Minimum Acceleration Zermelo Problem

that conditions are imposed on the control acceleration so that it is bounded by $0 \leq u_1 \leq 2$ throughout the trajectory.

The solution to this optimal control problem exhibits a 'bang-off-bang' structure. Intuitively, to minimize the accumulated thrust acceleration, u_1 must operate at either its maximum or minimum value. The solution to the transcribed optimization problem should exhibit this behavior, as well. For comparison, the problem is solved numerically using two different formulations. Figure 3.2 illustrates both solutions. The boat path, along with the control acceleration and pointing are shown for each solution. For the single segment solution, the formulation is identical to that described in Chapter 2, with a modified cost function. A solution with three segments is contrasted in the same figure, using the multiple-segment formulation described here with two time-free knots.

Note that for the single segment, although u_1 does see its extreme values, it is not a tight transition between extremes. Alternatively, the segmented formulation allows the optimizer to find solutions with perfect jump discontinuities. In this case, the formulation is configured to hold the control magnitudes constant at pre-specified values for each segment.

Anticipating a 'bang-off-bang' solution, the control acceleration is set at its maximum value for the first and third segments and at its minimum for the second segment. This forces the solution to contain the perfect jumps in the control history. Consequently, the segmented solution is a better approximation of the true analytical solution to the optimal control problem. Rather than seeking values of the control magnitude that make the constraint equations satisfied, the optimizer focuses primarily on determining the right time to switch control values. Remember with one segment, this could only happen between nodes, the placement of which is completely defined by t_0 and t_f. However, in a multiple segment formulation, the switching time is free to float to the optimal value. Since, the switching times are most likely useful information to a user, there is motivation to determine these precisely.

Clearly, the paths of each solution are similar. The cost of each solution is also similar (2.4852 with one segment, 2.4691 with three). Potentially, with more nodes, the single segment control history may appear more like the segmented solution's control history. However, for efficient use of parameters, a better solution generally comes from the segmented approach, especially if there is some insight into what the solution should look like.

Since the optimal solution has two jumps in the control, letting $n_s = 3$ in the segmented formulation is sufficient. However, what if it is not known in advance how many jumps in the control should exist in the solution? Conveniently, a solution can be found by formulating a problem with *more* segments than necessary. For example, a user could have chosen five segments, designating them as 'bang-off-bang-off-bang.' Since the actual solution has only two 'bang' segments, one at the beginning and one at the end, the optimizer would determine that Δt_3, the duration of the third segment (the middle 'bang'), is zero. The two 'off' segments would effectively occur sequentially, divided only by a zero duration segment, thus creating a continuous coasting arc. For general problem solving, a practical approach is to always include more segments than are needed. This gives the optimizer the freedom to use all segments if necessary, and bring superfluous segments to zero duration. The term used in this investigation to describe this practice is *overparameterization.*

Note also that by pre-designating the control acceleration magnitude in each segment, the values of $u_1(t_{j,k})$, the control acceleration at each node, are already known. They do not need to be included in the parameter vector! In addition, the values of $u_2(t_{j,k})$ in a coasting segment do not have any effect on the solution, as they describe the control pointing when there is no control acceleration. These values could be removed from $\boldsymbol{x}$ as well.

3.1.2 Switching Segments and Time for Multiple Independent Controls

Another application for the above formulation is a finite burn orbital transfer problem with control variables indicating thrust magnitude and direction. The multiple-segment formulation presented facilitates the realization of control discontinuities in the optimal solution. In this case, the optimal solution exhibits discontinuities in thrust magnitude. To enforce this, the problem is formulated to hold the thrust magnitude constant along each segment, and its value for each segment is pre-specified. For example, the k^{th} segment is given a specified thrust magnitude of $T = T^*$, while the segments before and after are designated to have zero thrust ($T = 0$). Thus, the knots on both sides of the k^{th} segment will exhibit control discontinuities, as the thrust magnitude changes instantaneously between 0 and T^*, as an optimal 'bang-bang' solution requires. The control variables of $\boldsymbol{U}_{n_u,n_n,n_s}$ represent thrust magnitudes in each direction, and are constrained such that $\boldsymbol{u}_{j,k}^T \boldsymbol{u}_{j,k} = T^2$ for each segment. The parameters, $\Delta \boldsymbol{T}_{n_s}$, indicate durations for thrust or coast segments. Notice, however, that all elements of $\boldsymbol{u}_{j,k}$ experience discontinuities simultaneously. In addition, along a segment, control variables may still vary over a continuous spectrum. It is observed, though, that the thrust magnitude is automatically limited to a finite set of values, at most equal to the number of segments (less, if some segments are designated the same value). The multiple-segment formulation presented above is effective when only one variable (in this case, thrust magnitude) exhibits jumps between values of a finite set.

When there are multiple independent controls that exhibit switching characteristics, however, additional modifications are necessary. For example, if the spacecraft orientation is restricted in the finite burn problem, multiple thrusters acting independently are required for 3-axis maneuvering. Applying the concepts developed above, the formulation is modified further to consider the situation when all control variables are limited to finite sets.

Assume that the i^{th} control variable is constrained to the set of values contained in $\mathbb{U}_i = \{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m}\}$. Thus, each control variable is limited to m different values. Let each control variable be similarly constrained. Although it is possible for the dimension of each control space, $\mathbb{U}_i$, to be different, assume for simplicity that each control variable can take on exactly m values. This constraint can be expressed as

$$\prod_{\mu=1}^{m} (u_i - \tilde{u}_{i,\mu}) = 0, \quad i = 1, \ldots, n_u. \tag{3.8}$$

Then, at each instant in time, the optimal solution must indicate which of the m possible values is best for each variable.

Ultimately, an optimizer must determine a control solution over the $\bar{m} = m^{n_u}$ possible control combinations at each instant in time. The optimization process is rooted in a gradient method, but gradients cannot be defined between discrete values. Instead, gradient methods treat parameters as continuous variables subject to constraints. Thus, the optimizer must be allowed to move the control parameters over a continuous space, as long as the constraint in Equation 3.8 is imposed in the final solution. Unfortunately, gradient methods move a point towards a root of the constraint according to the gradient at the current point of $\boldsymbol{x}$, independent of whether it is the *right* root to choose. Once the constraint is met, it is difficult, if not impossible, for the optimizer to seek another root. Although the constraint is valid, it cannot be implemented with traditional transcription.

Approach 1

To circumvent this problem, consider the multiple-segment formulation of Section 3.1.1. Instead of optimizing the control values, *assume* the control values and optimize the *switching times.* In one implementation, a formulation is devised that assigns each segment one of the $\bar{m}$ options, and then requires the optimizer to determine the duration of each segment. With this formulation, the predetermined control values are removed from the parameter vector. This is analogous to having one discrete control variable whose $\bar{m}$ values represent each of the possible control modes available to the system.

For example, a segment k is designated with control $\boldsymbol{u}(t_{j,k}) = [\tilde{u}_{1,1}, \ldots, \tilde{u}_{n_u,1}]^T$ for all nodes $j = 1, \ldots, n_n$ along that segment. If this control combination is not desirable in the optimal solution, the optimizer determines that the duration of that segment, Δt_k, is zero. With $n_s = \bar{m}$ segments, it is conceivable that the durations of several, if not a significant number of segments are reduced to zero. Recall that any segment includes parameters for the n_n nodes with n_y states assigned to each node. Since no time elapses over a zero duration segment, no changes are applied to any of the states over that segment. Thus a zero duration segment implies that some to many of the variables in the NLP problem are essentially wasted, taking up space and computational time, but contributing nothing to the solution.

In addition, without having insight into the optimal solution, no information is available regarding the order of the segments. Consider the optimal solution that requires one particular control combination before another. If the segments are not prearranged in that order, there is no way to achieve this solution under the formulation with only $\bar{m}$ segments. Instead, one might include multiple sets of $\bar{m}$ segments. This also allows for multiple firings in any direction. For example, with 3 sets, there are now $n_s = 3\bar{m}$ segments. The resulting NLP problem is naturally large, with many wasted variables to slow down the algorithm.

Approach 2

The formulation ultimately selected for this investigation minimizes the number of variables while offering the greatest flexibility in the solution process. Knots and segments are employed, but their definitions are slightly altered. This formulation is considerably more complex but consistently determines feasible and optimal solutions.

The salient feature of this formulation is that *each of the control axes is treated independently.* Let a knot be defined as any interior point where one member of the control vector switches from one value to another. The fundamental distinction here is that each knot *is associated with a control axis.* Before, the knot times could be described by $\boldsymbol{T}_{n_k}$, an array with n_k values. Likewise, the segment durations were $\Delta\boldsymbol{T}_{n_s}$, and there were $n_s = n_k + 1$ segments. Now, the knot times are described as $\boldsymbol{T}_{n_u,n_k}$, a two-dimensional array with $n_u \times n_k$ values. In other words, there are knots for each control axis. The *axis durations* are also two-dimensional, as $\Delta\boldsymbol{T}_{n_u,n_k+1}$, with $n_u \times (n_k + 1)$ values. These are the time durations between two switches in a given axis. It is important to note that this is not necessarily the duration of the segments. A segment is now defined as the time elapsed between switches in *any* of the controls. For example, the first segment is bounded both by t_0 and the first switch in the control, regardless of the axis in which the switch occurs. With $n_u n_k$ interior knots to separate the segments, the total number of segments becomes $n_s = n_u n_k + 1$. The nomenclature for the multiple independent control formulation is summarized in Table 3.2.

Consider the concept demonstrated, as an example, with $n_k = 3$ interior knots per each of $n_u = 3$ control axes. Let $\mathbb{U}_i = \{-1, 0, 1\}$ for each axis. Figure 3.3 demonstrates

Table 3.2: Nomenclature Summary: Multiple Independent Control Formulation

Array	Description	Dimension	Element
$\boldsymbol{Y}_{n_y,n_n,n_s}$	States by node	$n_y \times n_n \times n_s$	$\boldsymbol{y}_{j,k}$ or $y_{i,j,k}$
$\boldsymbol{U}^*_{n_u,n_k+1}$	Pre-specified controls	$n_u \times (n_k+1)$	$u^*_{i,k}$
$\boldsymbol{U}_{n_u,n_s}$	Controls by segment	$n_u \times n_s$	$u_{i,k}$
$\boldsymbol{U}_{n_u,n_n,n_s}$	Controls by node	$n_u \times n_n \times n_s$	$\boldsymbol{u}_{j,k}$ or $u_{i,j,k}$
$\boldsymbol{T}_{n_n,n_s}$	Node times	$n_n \times n_s$	$t_{j,k}$
$\boldsymbol{T}_{n_u,n_k}$	Knot times	$n_u \times n_k$	$t_{i,k}$
$\Delta\boldsymbol{T}_{n_u,n_k+1}$	Axis durations	$n_u \times (n_k+1)$	$\Delta t_{i,k}$
$\boldsymbol{T}'_{n_s+1}$	Unordered knot times	$0 \ldots n_s$	t'_k
$\boldsymbol{T}_{n_s+1}$	Ordered knot times	$0 \ldots n_s$	t_k
n_s	Number of segments	$n_u n_k + 1$	

the concept at two consecutive iterations of the optimization process. The control values are pre-designated in a pattern logical for an aerospace application, where nonzero control values indicate thrusting arcs. In each control axis, the control value begins at 1 (a positive thrusting arc), and a coasting arc follows the first knot. At the second knot, the control value switches to -1 (a negative thrusting arc) until the third knot, where another coasting arc begins. With additional knots, this pattern continues. The control values in each of the n_u control axes is designated similarly. In this example, observe that the total number of segments is $n_s = (3)(3) + 1 = 10$.

Let the time elements of the parameter vector, $\boldsymbol{x}$, be identified as $\boldsymbol{x}_t$. The time elements include all of the axis durations, $\Delta\boldsymbol{T}_{n_u,n_k+1}$, along with the initial and final time. Here, $\boldsymbol{x}_t$ is defined as

$$\boldsymbol{x}_t = \; [\; \Delta t_{1,1} \quad \Delta t_{1,2} \quad \Delta t_{1,3} \quad \Delta t_{2,1} \quad \Delta t_{2,2} \quad \Delta t_{2,3} \quad \Delta t_{3,1} \quad \Delta t_{3,2} \quad \Delta t_{3,3} \quad t_0 \quad t_f \;]^T.$$

The knot times, $\boldsymbol{T}_{n_u,n_k}$, are defined according to elements in $\boldsymbol{x}_t$.

$$t_{i,k} = t_0 + \sum_{\kappa=1}^{k} |\Delta t_{i,\kappa}| \,. \tag{3.9}$$

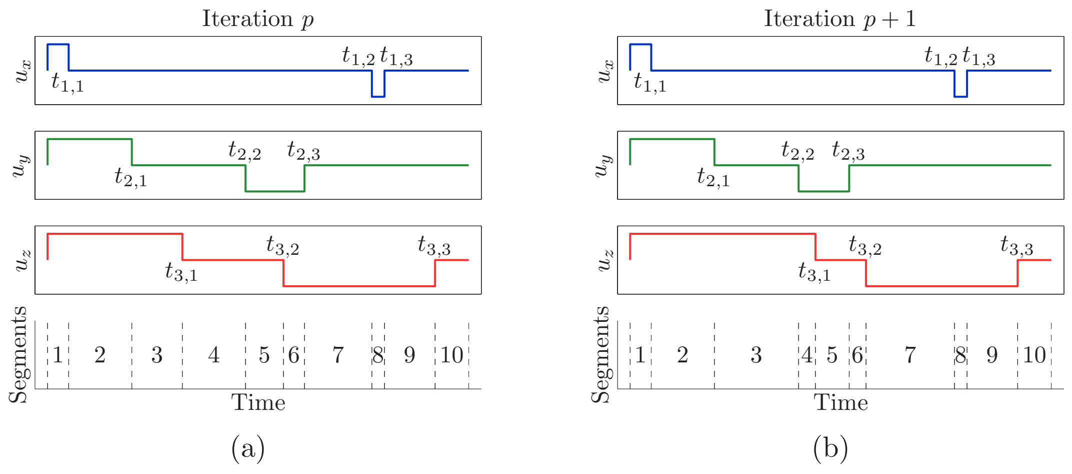


Figure 3.3: Conceptual Control Profile with Segment Divisions at Two Consecutive Iterations

Thus, $\Delta t_{i,k}$, represents the time duration between the knots located at $t_{i,k-1}$ and $t_{i,k}$. The definition in Equation 3.9 guarantees that the knot times remain in chronological order for a given control axis. That is,

$$t_{i,k-1} - t_{i,k} \leq 0, \quad k = 1, \ldots, n_k + 1$$

where $t_{i,0} = t_0$ and $t_{i,n_k+1} = t_f$. For example, in Figure 3.3, $t_{2,1}$ precedes $t_{2,2}$, which is followed by $t_{2,3}$. However, there is no relation between knot times in different control axes. Therefore, $t_{2,1}$ need not be after $t_{1,1}$. In general,

$$\iota < i \not\Rightarrow t_{\iota,\kappa} \leq t_{i,k} \; \forall \, i, \iota = 1, \ldots, n_u, \; \forall \, k, \kappa = 1, \ldots, n_k.$$

Compare the arrangement of knot times for iteration p and iteration $p+1$ in the figure. Both illustrations represent valid arrangements for the knots. However, it is clear that the chronological ordering of all the knots may change during the optimization process.

This algorithm defines segment boundaries by the chronological ordering of knot times, including t_0 and t_f at the end points. Let the unordered segment boundaries be defined as

$$\left[t'_0\; t'_1\; \cdots\; t'_\kappa\; \cdots\; t'_{n_s-1}\; t'_{n_s}\right] \equiv \left[t_0\; t_{1,1}\; \cdots\; t_{i,k}\; \cdots\; t_{n_u,n_k}\; t_f\right]. \tag{3.10}$$

Because knots are completely free to move on $[t_0\; t_f]$ and are independent between control axes, the segment boundaries, $\boldsymbol{T}'_{n_s+1}$, are not necessarily in ascending order. Thus, a sorting algorithm converts $\boldsymbol{T}'_{n_s+1}$ into the sequential listing, $\boldsymbol{T}_{n_s+1}$, of the segment boundaries. Now, t_{k-1} and t_k bound the k^{th} segment.

Observe, for example, that Segment 5 is bounded at the p^{th} iteration by knots located at $t_{2,2}$ and $t_{3,2}$. Between iterations, though, the axis durations $\Delta\boldsymbol{T}_{n_u,n_k+1}$ change values, thereby changing the positions of the knots at the next iteration. At iteration $p+1$, Segment 5 is bounded by $t_{3,1}$ and $t_{2,3}$. Completely different sets of variables now define the segment boundaries. Expressing the segment boundaries for Segment 5 in terms of $\boldsymbol{x}_t$, on the p^{th} iteration,

$$\begin{aligned} t_4(\boldsymbol{x}_t) &\equiv t_{2,2}(\boldsymbol{x}_t) = t_0 + |\Delta t_{2,1}| + |\Delta t_{2,2}|\,, \\ t_5(\boldsymbol{x}_t) &\equiv t_{3,2}(\boldsymbol{x}_t) = t_0 + |\Delta t_{3,1}| + |\Delta t_{3,2}|\,, \end{aligned}$$

while on the $(p+1)^{\text{th}}$ iteration,

$$\begin{aligned} t_4(\boldsymbol{x}_t) &\equiv t_{3,1}(\boldsymbol{x}_t) = t_0 + |\Delta t_{3,1}|\,, \\ t_5(\boldsymbol{x}_t) &\equiv t_{2,3}(\boldsymbol{x}_t) = t_0 + |\Delta t_{2,1}| + |\Delta t_{2,2}| + |\Delta t_{2,3}|\,. \end{aligned}$$

Thus, the dependencies of segment boundaries continuously change throughout the optimization process. The term *segment-time switching* is used to refer to the switching dependencies of a segment's time elements demonstrated here. Segment-time switching is the fundamental characteristic of the parameterization presented in this investigation. It is addressed in more detail subsequently.

3.2 Implementation of the Finite Set Control Transcription

In Section 3.1, a formulation is developed for solving finite set control problems. Starting from a traditional collocation method, the transcription is first augmented to manage multiple segments, and it is further modified for multiple switching controls. The resulting method is termed in this investigation the Finite Set Control Transcription (FSCT) method. It is characterized by containing only state and time elements in the parameter vector, pre-designated controls along each segment, and a segment-time switching phenomenon throughout the optimization process. Although the nature of switching dependencies is introduced above, a more complete understanding can be gained through a description of the FSCT implementation.

Implementation considerations are presented in the context of how an optimizer iterates to determine a solution to an NLP problem. At each major iteration, a step direction is determined using the partial derivatives of each function (F and $\boldsymbol{c}$) with respect to each variable in $\boldsymbol{x}$. Their role in a Quasi-Newton technique is discussed in Chapter 2. The Jacobian matrix comprised of these partial derivatives has dimension $(n_c + 1) \times n$, and for a collocation method, this can be quite large. For example, if the number of parameters, n, is of $\mathcal{O}(10^3)$, it is reasonable to have $\mathcal{O}(10^6)$ Jacobian elements. The Jacobian will generally be relatively sparse, as each function is only dependent upon a small number of variables. However, there still may be $\mathcal{O}(10^4)$ nonzero elements (or more) that are evaluated at each iteration. The performance of the optimization process is dependent upon these partials being evaluated quickly and accurately.

The partial derivatives can be calculated either numerically or analytically. If a derivative is determined numerically, a finite difference approximation is evaluated. To do this, the functions F and $\boldsymbol{c}$ are evaluated at $\boldsymbol{x}$ and at nearby points where one element of $\boldsymbol{x}$ is perturbed for each evaluation. When n is large, this process may become computationally expensive. Consequently, analytic derivatives are considered superior, not only because they may be faster to compute, but also since they are generally more accurate. However, the burden is on the user to supply routines to evaluate not only F and $\boldsymbol{c}$, but also $\frac{\partial F}{\partial \boldsymbol{x}}$ and $\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}$.

In this investigation, analytic derivatives are supplied to the optimizer to reduce computational time. In developing analytic expressions for problem derivatives, some interesting insights into the segment-time switching phenomenon may be observed. Specifically, switching dependencies occur within the continuity constraints, and consequently, the process of evaluating derivatives for these constraints is discussed below. This leads to further analysis of the characteristics of the FSCT method involving the accuracy of derivative evaluations and the convergence performance of the method.

First, however, it is necessary to demonstrate how the optimization parameters are manipulated at the beginning of each iteration in order to properly evaluate any constraints or derivatives.

3.2.1 The Optimization Parameters

In the FSCT method, the parameter vector is defined as

$$\boldsymbol{x} = \left[\cdots \; y_{i,j,k} \; \cdots \; \cdots \; \Delta t_{i,k} \; \cdots \; t_0 \; t_f\right]^T, \tag{1.8}$$

where $y_{i,j,k}$ is the i^{th} state at the j^{th} node of the k^{th} segment, and $\Delta t_{i,k}$ is the time duration for the i^{th} control variable at its k^{th} pre-specified value. The parameterization consists of state values, time durations between control switches, and bounding times t_0 and t_f. In the evaluation of cost and constraint functions, the elements of the current point, $\boldsymbol{x}$, are deparameterized and assigned to the respective state or time values that they represent. The 3-dimensional array, $\boldsymbol{Y}_{n_y,n_n,n_s}$, is assigned for the states, while the 2-dimensional array,

$\Delta\boldsymbol{T}_{n_u,n_k+1}$, carries the time durations. The deparameterization is simply

$$\begin{aligned} y_{i,j,k} &= x_{n_y n_n(k-1)+n_y(j-1)+i}, \\ \Delta t_{i,k} &= x_{q_1+n_u(k-1)+i}, \\ t_0 &= x_{q_2+1}, \\ t_f &= x_{q_2+2}, \end{aligned} \tag{3.11}$$

where $q_1 = n_y n_n n_s$ and $q_2 = q_1 + n_u(n_k+1)$. Adopting a shorthand notation, the expressions in Equation 3.11 are implied by

$$\boldsymbol{x} \rightarrow \boldsymbol{Y}_{n_y,n_n,n_s},\ \Delta\boldsymbol{T}_{n_u,n_k+1},\ t_0,\ t_f,$$

where $a \rightarrow b$ indicates a mapping from input a to output b. Likewise, the application of Equation 3.9 reveals the times of each knot.

$$t_0,\ \Delta\boldsymbol{T}_{n_u,n_k+1} \rightarrow \boldsymbol{T}_{n_u,n_k}.$$

Recall that the elements of $\boldsymbol{T}_{n_u,n_k}$ are $t_{i,k}$, where the control axis and knot number are distinguished by i and k respectively. Thus, in this form, knot times are arranged in a two dimensional array, and the chronological ordering of knot times is unknown.

Segment Boundaries and Control Values

In order to evaluate continuity constraint equations, the states, controls, and time must be known at each node of each segment. The states are contained in the array $\boldsymbol{Y}_{n_y,n_n,n_s}$, extracted directly from $\boldsymbol{x}$. The node times, $\boldsymbol{T}_{n_n,n_s}$, are easily determined through Equation 3.2 if the segment boundaries, $\boldsymbol{T}_{n_s+1}$, are known. Presently, then, $\boldsymbol{U}_{n_u,n_n,n_s}$ and $\boldsymbol{T}_{n_s+1}$ are extracted. Notice that the controls and segment boundaries are related, as the latter are simply the knot times that define switches in the control. The arrays are determined simultaneously, then, as the ordering of knots ultimately defines the control sequence.

In the first step, the segment boundaries are placed in an unordered listing according to Equation 3.10. The segment boundaries include the initial time, the final time, and each knot contained in $\boldsymbol{T}_{n_u,n_k}$. In shorthand,

$$t_0,\ \boldsymbol{T}_{n_u,n_k},\ t_f \rightarrow \boldsymbol{T}'_{n_s+1},$$

where the elements of the output are $t'_\kappa,\ \kappa = 0, \ldots, n_s$. It is clear that $t'_0 = t_0$ and $t'_{n_s} = t_f$, however, the interior segment boundaries are not arranged chronologically. Each interior segment boundary, $t'_\kappa,\ \kappa = 1, \ldots, n_s - 1$, is directly linked to an element, $t_{i,k}$, and it is necessary to maintain in storage the i, k pairing associated with each segment boundary. Next a sorting algorithm places the total collection of knots in chronological order to define the interior segment boundaries.

$$\boldsymbol{T}'_{n_s+1} \rightarrow \boldsymbol{T}_{n_s+1}$$

The segment boundaries, $\boldsymbol{T}_{n_s+1}$, represent the chronological ordering of knots. When the segment boundaries are sorted, the respective i, k pairings must be passed along. Thus, it is crucial to identify not only the time for each knot in $\boldsymbol{T}_{n_s+1}$, but also the element in $\boldsymbol{T}_{n_u,n_k}$ from which it came. This provides the basis for determining the control sequence along each segment.

Let the element $u^*_{i,k}$ be the pre-specified control value for the i^{th} control axis between $t_{i,k-1}$ and $t_{i,k}$. The array of pre-specified controls is $\boldsymbol{U}^*_{n_u,n_k+1}$, and has dimension $n_u \times n_k + 1$. In conjunction with the sorted knots, the segment control values are determined.

$$\boldsymbol{T}_{n_u,n_k},\ \boldsymbol{T}_{n_s+1},\ \boldsymbol{U}^*_{n_u,n_k+1} \rightarrow \boldsymbol{U}_{n_u,n_s}$$

The control values are completely contained in $\boldsymbol{U}_{n_u,n_s}$, where $u_{i,k}$ indicates the control value for the i^{th} axis on the k^{th} segment. In this process, the control values along the first segment are simply the first pre-specified control values, that is

$$u_{i,1} = u^*_{i,1}.$$

Then, at each segment boundary, all control values are held constant *except* in the control axis associated with the given knot. Since this knot identifies a control switch in its corresponding axis, the value in that control axis is updated to its next pre-specified value. Thus, if κ_i indicates the current control value of the i^{th} axis, then

$$\begin{aligned} u_{i,k} &= u^*_{i,\kappa_i} \\ u_{i,k+1} &= \begin{cases} u^*_{i,\kappa_i+1}, & t_k \equiv t_{i,\kappa_i}, \\ u^*_{i,\kappa_i}, & \text{otherwise.} \end{cases} \end{aligned} \tag{3.12}$$

Upon each control switch, the update $\kappa_i = \kappa_i + 1$ is performed in the switching axis to complete the recursive step. Equation 3.12 guarantees that along the final segment,

$$u_{i,n_s} = u^*_{i,n_k+1}.$$

In a final step, the controls are determined at each node.

$$\boldsymbol{U}_{n_u,n_s} \rightarrow \boldsymbol{U}_{n_u,n_n,n_s}$$

Since the controls are assumed constant over all segments, it is clear that $u_{i,j,k} = u_{i,k}$ for all nodes $j = 1, \ldots, n_n$. With this, all necessary elements are extracted, and the cost function and constraints can be evaluated in terms of the extracted elements.

3.2.2 Dynamical Constraints Using Simpson Integration Equations

The dynamical constraints are the central feature of a collocation method. When they are satisfied to tolerance, state continuity exists in the context of the given dynamical model. The dynamical constraints are also the key functions affected by the segment-time switching phenomenon.

The $n_y(n_n - 1)n_s$ dynamical constraints are of the form

$$\boldsymbol{c}_{\dot{y}}(\boldsymbol{x}) = \left[\boldsymbol{c}^T_{\dot{y}_{1,1}}(\boldsymbol{x}) \ \cdots \ \boldsymbol{c}^T_{\dot{y}_{j,k}}(\boldsymbol{x}) \ \cdots \ \boldsymbol{c}^T_{\dot{y}_{n_n-1,n_s}}(\boldsymbol{x})\right]^T,$$

where

$$\begin{aligned} \boldsymbol{c}_{\dot{y}_{j,k}}(\boldsymbol{x}) &= \boldsymbol{y}_{j+1,k} - \boldsymbol{y}_{j,k} \\ &\quad - \frac{h}{6}\left[\boldsymbol{f}\left(t_{j,k}, \boldsymbol{y}_{j,k}, \boldsymbol{u}_{j,k}\right) + 4\boldsymbol{f}\left(t_m, \boldsymbol{y}_m, \boldsymbol{u}_m\right) + \boldsymbol{f}\left(t_{j+1,k}, \boldsymbol{y}_{j+1,k}, \boldsymbol{u}_{j+1,k}\right)\right]. \end{aligned} \tag{3.13}$$

In Equation 3.13, the time variables are determined from previously extracted quantities by

$$\begin{aligned} h &= \frac{t_k - t_{k-1}}{n_n - 1}, \\ t_{j,k} &= t_{k-1} + h(j-1), \\ t_m &= \frac{1}{2}\left(t_{j,k} + t_{j+1,k}\right). \end{aligned}$$

Note that t_k are elements of the 1-dimensional array $\boldsymbol{T}_{n_s+1}$ representing segment boundaries, while $t_{j,k}$ are members of the 2-dimensional array $\boldsymbol{T}_{n_n,n_s}$ denoting the node times within a segment. The already extracted state and control values at the nodes, $\boldsymbol{y}_{j,k}$ and $\boldsymbol{u}_{j,k}$, are used to determine the midpoint states and controls:

$$\begin{aligned} \boldsymbol{y}_m &= \frac{1}{2}\left(\boldsymbol{y}_{j,k} + \boldsymbol{y}_{j+1,k}\right) + \frac{h}{8}\left(\boldsymbol{f}_{j,k} - \boldsymbol{f}_{j+1,k}\right), \\ \boldsymbol{u}_m &= \frac{1}{2}\left(\boldsymbol{u}_{j,k} + \boldsymbol{u}_{j+1,k}\right) \\ &= \boldsymbol{u}_{j,k} = \boldsymbol{u}_{j+1,k}. \end{aligned} \tag{3.14}$$

The result in Equation 3.14 is trivial, since the controls remain constant over any segment.

3.2.2.1 Partial Derivatives for the Simpson Integration Equations

For any set of dynamics, the general partial derivatives of $\boldsymbol{c}_{\dot{y}_{j,k}}$ may be determined as follows:

$$
\begin{aligned}
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{y}_{j,k}} &= -\boldsymbol{I} - \frac{h}{6}\left(\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{y}_{j,k}} + 4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j,k}}\right) && (3.15)\\
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{y}_{j+1,k}} &= \boldsymbol{I} - \frac{h}{6}\left(4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j+1,k}} + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{y}_{j+1,k}}\right) && (3.16)\\
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{u}_{j,k}} &= -\frac{h}{6}\left(\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{u}_{j,k}} + 4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{u}_m}\frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j,k}}\right) && (3.17)\\
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{u}_{j+1,k}} &= -\frac{h}{6}\left(4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{u}_m}\frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j+1,k}} + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{u}_{j+1,k}}\right) && (3.18)\\
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_{k-1}} &= -\frac{1}{6}\left(\boldsymbol{f}_{j,k} + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1,k}\right)\frac{\partial h}{\partial t_{k-1}} - \frac{h}{6}\left[\frac{\partial \boldsymbol{f}_{j,k}}{\partial t_{j,k}}\frac{\partial t_{j,k}}{\partial t_{k-1}}\right.\\
&\quad \left. + 4\left(\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial h}\frac{\partial h}{\partial t_{k-1}} + \frac{\partial \boldsymbol{f}_m}{\partial t_m}\frac{\partial t_m}{\partial t_{k-1}}\right) + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial t_{j+1,k}}\frac{\partial t_{j+1,k}}{\partial t_{k-1}}\right] && (3.19)\\
\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_k} &= -\frac{1}{6}\left(\boldsymbol{f}_{j,k} + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1,k}\right)\frac{\partial h}{\partial t_k} - \frac{h}{6}\left[\frac{\partial \boldsymbol{f}_{j,k}}{\partial t_{j,k}}\frac{\partial t_{j,k}}{\partial t_k}\right.\\
&\quad \left. + 4\left(\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial h}\frac{\partial h}{\partial t_k} + \frac{\partial \boldsymbol{f}_m}{\partial t_m}\frac{\partial t_m}{\partial t_k}\right) + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial t_{j+1,k}}\frac{\partial t_{j+1,k}}{\partial t_k}\right]. && (3.20)
\end{aligned}
$$

In Equations 3.15-3.20, $\frac{\partial \boldsymbol{f}}{\partial t}$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$, and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ are dependent on the chosen dynamics, while

$$
\begin{aligned}
\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j,k}} &= \frac{1}{2}\boldsymbol{I} + \frac{h}{8}\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{y}_{j,k}}, & \frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j+1,k}} &= \frac{1}{2}\boldsymbol{I} - \frac{h}{8}\frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{y}_{j+1,k}},\\
\frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j,k}} &= \frac{1}{2}\boldsymbol{I}, & \frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j+1,k}} &= \frac{1}{2}\boldsymbol{I},\\
\frac{\partial h}{\partial t_{k-1}} &= -\frac{1}{n_n - 1}, & \frac{\partial h}{\partial t_k} &= \frac{1}{n_n - 1},\\
\frac{\partial t_{j,k}}{\partial t_{k-1}} &= 1 - \frac{j-1}{n_n - 1}, & \frac{\partial t_{j,k}}{\partial t_k} &= \frac{j-1}{n_n - 1},\\
\frac{\partial t_m}{\partial t_{k-1}} &= 1 - \frac{j-\frac{1}{2}}{n_n - 1}, & \frac{\partial t_m}{\partial t_k} &= \frac{j-\frac{1}{2}}{n_n - 1}.
\end{aligned}
$$

Considering the formulation presented in this investigation, one may observe immediately that the derivatives $\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{x}}$ are completely defined without needing to apply Equations 3.17

and 3.18. Since the control values are pre-specified in this formulation, they are not optimization variables and their constraint partials can be ignored. They are included above only for completeness.

In contrast, Equations 3.15 and 3.16 are directly implementable, as $\boldsymbol{y}_{j,k}$ are variables contained in the parameter vector, $\boldsymbol{x}$. If the variables are divided into state elements and time elements, $\boldsymbol{x} = \left[\boldsymbol{x}_y^T \; \boldsymbol{x}_t^T\right]^T$, then the Jacobian elements are

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{x}_{y_\gamma}} = \begin{cases} \dfrac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{y}_{j,k}}, & \boldsymbol{x}_{y_\gamma} \equiv \boldsymbol{y}_{j,k}, \\ \dfrac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{y}_{j+1,k}}, & \boldsymbol{x}_{y_\gamma} \equiv \boldsymbol{y}_{j+1,k}, \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

Alternatively, Equations 3.19 and 3.20 are not immediately implementable because the time variables t_{k-1} and t_k do not appear directly as parameters in $\boldsymbol{x}$. However, t_{k-1} and t_k are dependent upon the time parameters, $\boldsymbol{x}_t$, such that

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \boldsymbol{x}_t} = \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_{k-1}} \frac{\partial t_{k-1}}{\partial \boldsymbol{x}_t} + \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_k} \frac{\partial t_k}{\partial \boldsymbol{x}_t}. \tag{3.21}$$

To apply Equation 3.21, the partials $\frac{\partial t_k}{\partial \boldsymbol{x}_t}$ must be determined. Recall from Section 3.2.1 that segment boundaries of $\boldsymbol{T}_{n_s+1}$ are directly linked to knot times $\boldsymbol{T}_{n_u,n_k}$ that are functions of the parameters in $\boldsymbol{x}_t$. Therefore, it is possible to determine

$$\frac{\partial t_k}{\partial t_0}, \quad \frac{\partial t_k}{\partial t_f}, \quad \text{and} \quad \frac{\partial t_k}{\partial \Delta t_{i,\kappa}},$$

for all segments boundaries $k = 0, \ldots, n_s$, control elements $i = 1, \ldots, n_u$, and knots $\kappa = 1, \ldots, n_k$. Beginning with exterior segment boundaries, it is clear that, since $t_{n_s} = t_f$,

$$\begin{aligned} \frac{\partial t_0}{\partial t_0} &= 1, & \qquad \frac{\partial t_{n_s}}{\partial t_0} &= 0, \\ \frac{\partial t_0}{\partial t_f} &= 0, & \qquad \frac{\partial t_{n_s}}{\partial t_f} &= 1, \\ \frac{\partial t_0}{\partial \Delta t_{i,\kappa}} &= 0, & \qquad \frac{\partial t_{n_s}}{\partial \Delta t_{i,\kappa}} &= 0. \end{aligned}$$

This is true regardless of the ordering for interior knots. However, interior segment boundaries in $\boldsymbol{T}_{n_s+1}$ correspond to elements in the knot array $\boldsymbol{T}_{n_u,n_k}$. That is,

$$t_k \equiv t_{i,\kappa}, \quad 1 \leq k \leq n_s - 1$$

for some pair i, κ. Likewise, the derivatives of the element $t_{i,\kappa}$ are assigned to the element t_k,

$$\frac{\partial t_k}{\partial \boldsymbol{x}_t} \equiv \frac{\partial t_{i,\kappa}}{\partial \boldsymbol{x}_t}, \quad 1 \leq k \leq n_s - 1$$

Thus, all that remains is to identify the proper derivatives for the knot times. Under the assumption that all $\Delta t_{i,\kappa} \geq 0$,

$$\begin{aligned} \frac{\partial t_{i,k}}{\partial t_0} &= 1, \\ \frac{\partial t_{i,k}}{\partial \Delta t_{\iota,\kappa}} &= \begin{cases} 1, & \iota = i,\ \kappa \leq k, \\ 0, & \text{otherwise,} \end{cases} \\ \frac{\partial t_{i,k}}{\partial t_f} &= 0. \end{aligned}$$

With these derivatives carefully matched, the components of Equation 3.21 are completely defined.

To further understand the impact of segment-time switching on derivative evaluation, consider again the illustration of Figure 3.3. The partial derivatives of the segment boundary times with respect to $\boldsymbol{x}_t$ at the p^{th} and $(p+1)^{\text{th}}$ iterations are given below.

$$\begin{array}{lccccccccccccccc}
\boldsymbol{x}_t & = & [& \Delta t_{1,1} & \Delta t_{1,2} & \Delta t_{1,3} & \Delta t_{2,1} & \Delta t_{2,2} & \Delta t_{2,3} & \Delta t_{3,1} & \Delta t_{3,2} & \Delta t_{3,3} & t_0 & t_f &]^T \\
{\dfrac{\partial t_4}{\partial \boldsymbol{x}_t}}_{p} & = & [& 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 &] \\
{\dfrac{\partial t_5}{\partial \boldsymbol{x}_t}}_{p} & = & [& 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 &] \\
{\dfrac{\partial t_4}{\partial \boldsymbol{x}_t}}_{p+1} & = & [& 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 &] \\
{\dfrac{\partial t_5}{\partial \boldsymbol{x}_t}}_{p+1} & = & [& 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 &]
\end{array}$$

Thus, the segment boundary derivatives with respect to elements in $\boldsymbol{x}_t$ are always either 0 or 1, and it is the ordering of the knots that determine the proper structure at each iteration. The constraint derivatives for the 5th segment, then, is defined by

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial \boldsymbol{x}_t} = \frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial t_4}\frac{\partial t_4}{\partial \boldsymbol{x}_t} + \frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial t_5}\frac{\partial t_5}{\partial \boldsymbol{x}_t},$$

and clearly demonstrates dramatically different behavior on each iteration.

3.2.2.2 Time Invariance

The parameter t_0 appears in the definition of every knot time, with the exception of $t_{n_s} = t_f$. For an interior Segment k, then, t_0 defines both t_{k-1} and t_k. If the dynamic model is time invariant, the partial derivatives $\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_{k-1}}$ and $\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_k}$ have a canceling effect in Equation 3.21 such that

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_0} = \boldsymbol{0}$$

on an interior segment. Thus, because the parameters are axis durations instead of absolute times, t_0 and t_f only have nonzero effects on the last segment.

Likewise, in a time invariant formulation, if a Segment k is bounded by two knots associated with the same axis (such as, $t_{i,\kappa-1}$ and $t_{i,\kappa}$), all of the partials will cancel out with exception to that associated with $\Delta t_{i,\kappa}$. That is,

$$\begin{aligned} \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \Delta t_{i,\kappa}} &= \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_k}, \\ \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial (x_t)_\gamma} &= \boldsymbol{0}, \quad \text{otherwise.} \end{aligned}$$

3.2.2.3 Derivative Discontinuities for the Dynamical Constraints

The dynamical constraints exhibit discontinuities in function derivatives when the chronological ordering of the knots switch. This is demonstrated in how the partials $\frac{\partial t_k}{\partial \boldsymbol{x}_t}$

may change between the values 0 and 1 between iterations. A simple analogy is useful in conceptualizing this phenomenon. Consider a simple function of two variables defined by

$$f(x_1, x_2) = \min\{x_1, x_2\}.$$

If this function appears in a parameter optimization problem as either the objective or a constraint, its evaluation is straightforward:

$$f = \begin{cases} x_1, & x_1 < x_2, \\ x_1 = x_2, & x_1 = x_2, \\ x_2, & x_1 > x_2. \end{cases}$$

Interestingly, when $x_1 < x_2$, variable x_2 has seemingly no impact on the function. However, there is a switch in dependency when x_2 is smaller than x_1. The characteristics of this function are quite similar to those of the dynamical constraints in the FSCT method. Consider the derivative with respect to the first variable.

$$\frac{\partial f}{\partial x_1} = \begin{cases} 1, & x_1 < x_2, \\ \text{undefined}, & x_1 = x_2, \\ 0, & x_1 > x_2. \end{cases}$$

A derivative discontinuity exists when x_1 and x_2 are equal. However, an optimizer requires derivatives to be tractable at any point within the range of optimization. Therefore, if evaluating $\frac{\partial f}{\partial x_1}$ or $\frac{\partial f}{\partial x_2}$ analytically, it is up to the user to choose an appropriate definition in the case that $x_1 = x_2$. Three reasonable candidates for $\left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2}$ may be seen in how numerical derivatives could be evaluated using forward, backward or central differences:

$$\begin{aligned} \text{Forward:} \quad \left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} &= \frac{f(x_1+\delta, x_2) - f(x_1, x_2)}{\delta} &= 0, \\ \text{Backward:} \quad \left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} &= \frac{f(x_1, x_2) - f(x_1-\delta, x_2)}{\delta} &= 1, \\ \text{Central:} \quad \left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} &= \frac{f(x_1+\delta, x_2) - f(x_1-\delta, x_2)}{2\delta} &= \frac{1}{2}. \end{aligned}$$

Thus, choosing an analytic expression for $\left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2}$ is equivalent to selecting a finite differencing scheme for numerically evaluated derivatives. In most cases, the selection is arbitrary,

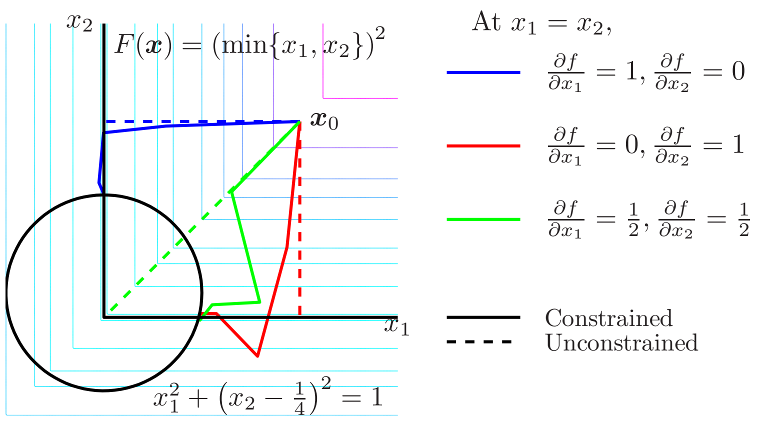


Figure 3.4: Effects of Alternative Derivative Definitions on Optimization Path

as there is probably no basis for valuing one method over another. However, it is clear that when $x_1 = x_2$, the path of the optimizer can be altered dramatically simply by selecting a different derivative definition.

For example, consider the optimization of the function $F(\boldsymbol{x}) = (f(x_1, x_2))^2$ illustrated in Figure 3.4. Note that the absolute minimum exists at any point where either $x_1 = 0$ or $x_2 = 0$ and the other variable is positive. Six optimal paths are shown, demonstrating unconstrained and constrained solutions. In the constrained cases, the equation $x_1^2 + \left(x_2 - \frac{1}{4}\right)^2 = 1$ is imposed. The paths differ in how the derivatives are defined along the edge $x_1 = x_2$, using forward, backward, or central differencing schemes. Thus, from a starting point along the edge, each derivative definition converges on a different locally optimal point. However, in each of the six cases, the resulting solution equates to the global minimum value of the cost function.

This conceptualization is easily extended to the continuity constraints of the FSCT method. Consider as an example the 4-knot sequence described in Figure 3.5 where $t_0 \leq t_1 \leq t_2 \leq t_3$. The first four knots are defined by t_0 and the knot times $t_{2,1}$, $t_{3,1}$, and $t_{1,1}$,

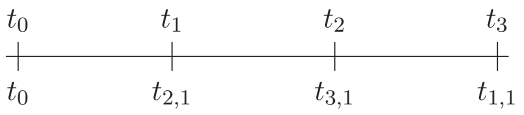


Figure 3.5: Sample Four-Knot Sequence

respectively. This implies that, in this example,

$$\Delta t_{2,1} \leq \Delta t_{3,1} \leq \Delta t_{1,1}.$$

According to the FSCT implementation of analytic derivatives, the associated dynamics equations for the three segments shown have time dependencies as given in Table 3.3. Thus, when time derivatives are calculated for the dynamical constraints on one of these segments, nonzero Jacobian elements appear for their respective dependent members of $\boldsymbol{x}_t$. The derivatives with respect to all other members of $\boldsymbol{x}_t$, including t_f and other members of $\Delta \boldsymbol{T}_{n_u,n_k+1}$, are identically zero.

As long as the knots are sufficiently far away, no difficulties confront the solution process. However, consider further the scenario that

$$\Delta t_{2,1} = \Delta t_{3,1} \leq \Delta t_{1,1}$$

such that $t_1 = t_2$. In the context of the dynamical problem, this situation indicates that the controls in two axes must switch simultaneously. Thus, knots come together to exist at

Table 3.3: Segment Dependencies for Analytic and Numerical (Forward Differencing) Derivatives

		Dependent Members of $\boldsymbol{x}_t$	
Segment	Boundaries	Analytic Derivatives	Forward Differences
1	$t_0,\ t_1$	$t_0,\ \Delta t_{2,1}$	$t_0,\ \boldsymbol{\Delta t_{3,1}}$
2	$t_1,\ t_2$	$t_0,\ \Delta t_{2,1},\ \Delta t_{3,1}$	$t_0,\ \boldsymbol{\Delta t_{3,1}},\ \boldsymbol{\Delta t_{2,1}}$
3	$t_2,\ t_3$	$t_0,\ \Delta t_{3,1},\ \Delta t_{1,1}$	$t_0,\ \boldsymbol{\Delta t_{2,1}},\ \Delta t_{1,1}$

the same point in time, but their ordering is still specified according to the implementation (in this case, $t_1 \equiv t_{2,1}$ is ordered before $t_2 \equiv t_{3,1}$). In this situation, it is possible for the $\boldsymbol{x}_t$ dependencies determined by numerical derivatives to be completely different then the analytic set. To illustrate, assume that finite differences are calculated using forward differences such that

$$\frac{\partial c}{\partial x} \approx \frac{c(x+\delta)-c(x)}{\delta}.$$

Then when $\Delta t_{2,1}$ is perturbed forward by $\delta > 0$, the knot ordering will necessary change since

$$\Delta t_{3,1} \leq \Delta t_{2,1} + \delta \leq \Delta t_{1,1}.$$

The $\boldsymbol{x}_t$ dependencies via forward differences are also presented in Table 3.3. To contrast the two derivative methods, variations for forward differences are highlighted. Thus, according to the forward difference, changes to $\Delta t_{2,1}$ have no effect on Segment 1, as its bounding times remain the same even when the parameter is perturbed. Instead, when $\Delta t_{2,1}$ is perturbed forward, it affects Segment 3, despite the fact that knot ordering indicated otherwise. Likewise, on Segment 2, analytic calculations consider $\Delta t_{2,1}$ as the left bounding time, while a forward differencing scheme necessarily implies it is the right bounding time. On Segment 2, one method will result in $\frac{\partial c}{\partial \Delta t_{2,1}} > 0$, while the other will compute a negative value.

Therefore, as with the sample function, $f = \min\{x_1, x_2\}$, several derivative definitions can be employed at the switching points where derivative discontinuities exist. It is clear that the path of $\boldsymbol{x}$ chosen by the optimizer will vary based on the derivatives calculated, but this does not inhibit the FSCT method. Considering the total number of parameters in $\boldsymbol{x}$ and the relatively few gradients that can be affected by the switching phenomenon, the optimizer can still make improvements in $\boldsymbol{x}$ on an iteration where a switch occurs, regardless of how the derivative is defined. Although the specific search direction of the switching iteration may vary, it is observed that an identical local minimum can be found via different paths. Thus, these derivative discontinuities do not present obstacles in determining an optimal solution.

It should be noted before proceeding that this situation only arises when two knots originating from *different* control axes occur simultaneously. Based on the definitions of knot times, knots in one control axis cannot switch in order. However, two knots in one axis can exist simultaneously when a given element $\Delta t_{i,k} = 0$. In fact, this is a common occurrence, as the user may pre-specify a value of $u^*_{i,k}$ that is simply not desired in the optimal solution. The optimizer makes zero a time duration in order to remove the non-optimal control value. Having simultaneous knots in a *single* axis is an anticipated condition, and it is important that this situation does not cause any adverse effects in derivative calculations. This is guaranteed in the definition of Equation 3.9, which is specifically formulated to produce this result.

3.2.3 Implementation of Numerical Derivatives

When a user selects a finite differencing method for evaluating function derivatives, some additional consideration may be required. Depending on the specific optimization algorithm, the optimizer may calculate finite differences automatically in the absence of analytic derivative expressions. If this is the case, the FSCT method's derivatives may not be effectively evaluated without user intervention.

Some sophisticated optimizers perform initialization routines to more efficiently calculate numerical derivatives. To avoid excess computation, functions are evaluated from an initial or random point of $\boldsymbol{x}$ to determine the structure of the Jacobian matrix. Linear constraints (constant Jacobian elements) and nonlinear constraints (varying Jacobian elements) are identified. This allows the routine to characterize the sparsity of the Jacobian. By performing this task in the initialization, the algorithm seeks to avoid recalculating non-changing elements. However, in the context of the FSCT implementation, this procedure will not be able to adequately identify the potential dependencies that would appear from a different evaluation point. Some Jacobian elements will *appear* to be nonvarying at a particular point based on the current arrangement of knots. If the initialization routine falsely identifies elements as constant (zero or nonzero), then proper derivatives are not evaluated for future iterations when the knot arrangement is different. Consequently, the optimizer cannot determine the best search direction for the next iteration point, and it is unlikely that the optimizer would converge on an optimal point.

This can be overcome with user defined procedures to flag *all* potential dependencies (nonzero Jacobian elements) as varying gradients. With this information, the optimizer will perform finite differences for each element, actively determining the dependencies on each iteration. Efficiency degrades, and there is necessarily excess computation, but derivatives can be calculated accurately.

3.2.4 Other Constraints

So far, the implementation of the FSCT method is presented in the context of the continuity constraints and the segment-time switching phenomenon that exists in those constraints. However, the continuity constraints along segments are only a subset of the constraints necessary for successful implementation of the method. Of course, the specific set of constraints may be dependent upon the actual problem transcribed. However, common to most applications are three additional subsets of constraints described presently. These implement various initial, knot, and time conditions.

3.2.4.1 Initial States and Time

The initial conditions most often employed with optimal control problems have all of the initial states as well as the initial time fixed. The optimization question posed is most often of the form: *Given where I am now, how do I do what I want to do?* Thus, the constraints, $\boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) = \mathbf{0}$, will capture fixed initial states and time.

Notice that the initial states are simply the states assigned to the first node of the first segment, $y_{i,1,1}$. Let the specified initial states be identified as $(\boldsymbol{y}_0^*)$, and the specified initial time as t_0^*. Then $n_{\psi_0} = n_y + 1$ constraints are imposed to represent $\boldsymbol{\psi}_0$ as

$$\boldsymbol{c}_{\psi_0}(\boldsymbol{x}) = \begin{bmatrix} y_{1,1,1} - (y_0^*)_1 \\ \vdots \\ y_{i,1,1} - (y_0^*)_i \\ \vdots \\ y_{n_y,1,1} - (y_0^*)_{n_y} \\ t_0 - t_0^* \end{bmatrix}.$$

The optimizer drives this vector to zero during the optimization. However, simple constraints like this could and should be met on the first iteration by appropriately assigning the initial conditions in the first guess for $\boldsymbol{x}$.

The Jacobian elements for these constraints are simple to compute.

$$\begin{aligned} \frac{\partial c_{\psi_{0_i}}}{\partial x_\gamma} &= \begin{cases} 1, & x_\gamma \equiv y_{i,1,1} \Leftrightarrow \gamma = i, \\ 0, & \text{otherwise}, \end{cases} \\ \frac{\partial c_{\psi_{0_{n_y+1}}}}{\partial x_\gamma} &= \begin{cases} 1, & x_\gamma \equiv t_0 \Leftrightarrow \gamma = n_y n_n n_s + n_u(n_k+1) + 1, \\ 0, & \text{otherwise}, \end{cases} \end{aligned}$$

for $i = 1, \ldots, n_y$.

Note that specified final states and time would be treated identically.

3.2.4.2 Segment Continuity Between Knots

Consider a set of dynamics where all states must be continuous across segments. Since no time elapses between the end of the k^{th} segment and the beginning of the $(k+1)^{\text{th}}$ segment, the states must be equal at those points. The $n_y(n_s - 1)$ segment constraints are formulated as

$$\boldsymbol{c}_s(\boldsymbol{x}) = \begin{bmatrix} y_{1,1,2} & - & y_{1,n_n,1} \\ & \vdots & \\ y_{i,1,k+1} & - & y_{i,n_n,k} \\ & \vdots & \\ y_{n_y,1,n_s} & - & y_{n_y,n_n,n_s-1} \end{bmatrix},$$

with Jacobian elements,

$$\frac{\partial c_{s_{n_y(k-1)+i}}}{\partial x_\gamma} = \begin{cases} 1, & x_\gamma \equiv y_{i,1,k+1}, \\ -1, & x_\gamma \equiv y_{i,n_n,k}, \\ 0, & \text{otherwise}, \end{cases}$$

for $i = 1, \ldots, n_y$ and $k = 1, \ldots, n_s - 1$.

3.2.4.3 Time

Time equality constraints ensure that the sums of the axes durations for each control axis are equal to the trajectory time of flight, $t_f - t_0$. That is, the n_u time constraints are

$$\boldsymbol{c}_t(\boldsymbol{x}) = \begin{bmatrix} t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{1,\kappa}| \\ \vdots \\ t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{i,\kappa}| \\ \vdots \\ t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{n_u,\kappa}| \end{bmatrix}.$$

The Jacobian elements for these constraints are

$$\frac{\partial c_{t_i}}{\partial x_\gamma} = \begin{cases} 1, & x_\gamma \equiv t_f, \\ -1, & x_\gamma \equiv t_0, \\ & \\ -1, & x_\gamma \equiv \Delta t_{i,k}, \\ 0, & \text{otherwise}. \end{cases}$$

for $i = 1, \ldots, n_u$ and $k = 1, \ldots, n_k + 1$. For these derivatives, it is assumed that $\Delta t_{i,k} \geq 0$. This is a reasonable assumption if simple bounds keep the time durations nonnegative. Alternatively, the derivatives associated with the durations are $-\text{sign}(\Delta t_{i,k})$ instead of -1.

3.3 Summary

It is often the case that unique problems require unique methods for determining solutions. This work explores one such problem, in which a continuous dynamical system is subject to a finite control space. This development results in an enhanced collocation method specifically designed to treat optimal control problems under this type of control constraint.

The Finite Set Control Transcription method is presented in this chapter along with implementation considerations. The class of hybrid problem targeted for this development consists of continuous states and discrete controls. It should be noted that further extensions of this method may be applied to different classes of hybrid control problems. One in particular considers both discrete and continuous control variables. The FSCT method may be augmented to treat extra continuous control variables as in a traditional collocation approach (see Chapter 6). Extensions can also be considered for autonomous switching with appropriate definitions of knot conditions.

The FSCT method is also applicable for completely continuous systems. For example, the method proves useful in forcing 'bang-bang' control histories, which result even when control variables are only constrained by saturation limits. The nature of the parameterization allows for switching times to be optimized over a continuous spectrum, unlike most collocation schemes. This facilitates the transition from the FSCT (parameterized) solution to the implementation in the actual continuous system.

The transcription and implementation development presented here is sufficient for a user to formulate any number of appropriate applications. These follow in subsequent chapters. General applications explore single and multiple discrete variables, 'bang-bang' control, and solution precision. Further applications demonstrate the FSCT method's capability in handling large-scale, complex problems.

Chapter 4

General Applications

Having detailed the implementation for the Finite Set Control Transcription (FSCT) method, it is beneficial to explore the capability and utility of the method by demonstrating a range of applications. In so doing, the scope of the method is characterized, ideally inspiring additional applications outside those presented here. The applications presented are focused on aerospace systems, as these served to motivate the method's development. The basic applications consider the stability of a switched linear system, minimum-time and minimum-acceleration solutions to a 2-dimensional lunar lander problem, and spacecraft attitude control based on inexpensive cold-gas thruster technology.

As the method is explored, several other hybrid control methods are introduced. A stability method involving *multiple Lypunov functions* is contrasted with the FSCT method, while a hybrid *model predictive control* scheme is used in conjunction with this investigation's focus. In the final application, the attitude dynamics of a spacecraft are expressed by two different models to demonstrate how control inputs traditionally considered in a continuous form may be expressed as a combination of an augmented set of continuous states and discrete controls.

To begin, a process for supplying initial guesses for the optimization is outlined. The initial guess is as crucial as the method itself for arriving at useful solutions. Although there may be several effective philosophies on how to generate an initial guess, the following process produces successful results for the applications in this and subsequent chapters.

4.1 Producing Effective Initial Guesses

Of course, one natural characteristic of the numerical optimization process is that it is iterative: a user must supply the optimizer with an initial guess for $\boldsymbol{x}$, and allow the algorithm to improve upon that value until the constraint functions are satisfied and the cost function is minimized. Therefore, the identification of adequate solutions requires a good initial guess, $\boldsymbol{x}_0$. In some sense, $\boldsymbol{x}_0$ should be 'close' to the desired solution. The final point (that is, the converged, optimal parameters), $\boldsymbol{x}_f$, will most likely be in the same region as $\boldsymbol{x}_0$. However, it is possible to expand the region containing the initial guess and final solution by ensuring that the initial guess is not 'too close' to an existing local minimum. It is certainly a balancing act, and determining the initial guess is often the most difficult aspect of numerical optimization. Thus, it is crucial to provide a philosophy by which effective initial guesses for $\boldsymbol{x}$ can be realized.

In this investigation, $\boldsymbol{x}_0$ is generated by selecting states along or close to a desired (or maybe anticipated) solution, while controls (that is, time durations) are arbitrarily selected. This approach is convenient because the analyst often possesses some intuitive knowledge regarding the behavior of the states. Initial conditions, final conditions, and path data, for example, are often available and can be easily converted into the state values of $\boldsymbol{x}_0$. Once the times for the nodes are known, a guess for $y_{i,j,k}$ can be interpolated. Likewise, starting values of t_0 and t_f can generally be deduced intuitively. Of course, identifying candidate initial values for the control history is not intuitive. In that case, it is convenient to assign initial values for control switches arbitrarily.

This approach can be quite effective. An arbitrarily designated control sequence, combined with an intuitive set of state parameters, generally results in a non-feasible initial guess. Of course, since the starting point is not feasible, it is also not likely in the vicinity of an existing local minimum. The optimizer will necessarily adjust the initial values of all the parameters to make $\boldsymbol{x}$ feasible, moving $\boldsymbol{x}$ away from what may be a poor initial guess.

However, by observing that most often $n_y n_n n_s + 2 > n_u(n_k + 1)$ (the number of states is greater than the number of control time parameters), the state elements within $\boldsymbol{x}_0$ provide inertia to keep $\boldsymbol{x}$ in the same vicinity. Thus, this approach helps in producing initial guesses that are not too close or too far away from a good solution.

In this study, the control values are generally pre-specified in a manner similar to the profile of Figure 1.1. The control values are selected as an ordered set, where each of the feasible values is present for multiple time durations. One way of expressing the pre-specified control values in this example is

$$u^*_{i,k} = \tilde{u}_{i,\mathrm{mod}(k-1,m_i)+1},$$

for $i = 1, \ldots, n_u$ and $k = 1, \ldots, n_k + 1$. This employs the modulus function, where $\mathrm{mod}(a, b) = a - \eta b$, and η is the largest integer multiplier such that $\eta b \leq a$. This choice can be altered should problem intuition dictate a less arbitrary arrangement.

An effective strategy for guessing the time durations, $\Delta t_{i,k}$, allows for uniform segment durations for all segments in the initial guess. One way of accomplishing this is through the definitions,

$$\begin{aligned} \bar{\Delta} &= \frac{t_f - t_0}{n_s} \\ \Delta t_{i,k} &= \left\{ \begin{array}{rl} i\bar{\Delta}, & k = 1 \\ n_u\bar{\Delta}, & k = 2, \ldots, n_k \\ (n_u + 1 - i)\bar{\Delta}, & k = n_k + 1 \end{array} \right. , \end{aligned}$$

for $i = 1, \ldots, n_u$. This guarantees that, upon the first iteration of the optimization, each segment has duration $\bar{\Delta}$, and the knots are placed as far apart as possible. Notice also that there is a structured rotation between the control variables with regard to switching times: u_1 switches, followed by u_2, and so on. The order of control switches is free to change throughout the optimization process (u_i does not always switch before u_{i+1}), but the initial ordering and initial separation between knot times allows for free movement of the knots in order to arrive at a feasible and locally optimal switching structure. This philosophy for generating $\boldsymbol{x}_0$ is practiced in each of the applications that follow.

4.2 Two Stable Linear Systems

Although the FSCT method is especially effective for multiple control variables, consider first a system controlled by only one decision. The system is

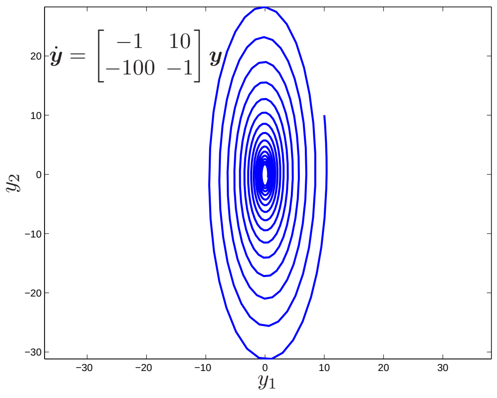
$$\begin{aligned} \dot{\boldsymbol{y}} &= \boldsymbol{f}(\boldsymbol{y}, u) = \boldsymbol{A}_u \boldsymbol{y}, && (4.1) \\ u &\in \{1, 2\}, && (4.2) \end{aligned}$$

where

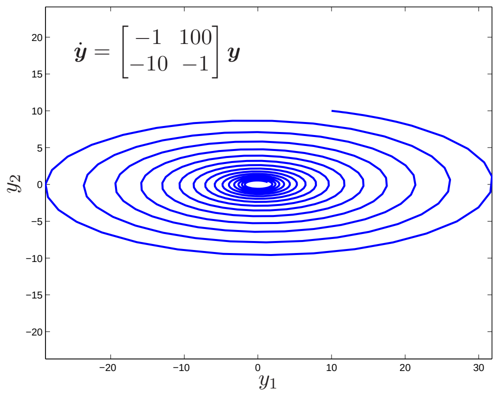
$$\boldsymbol{A}_1 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}, \qquad \boldsymbol{A}_2 = \begin{bmatrix} -1 & 100 \\ -10 & -1 \end{bmatrix}.$$

Thus, the system is characterized by two separate dynamical modes, and the decision variable determines which of the two is in play at any given time. Notice that individually, each mode is a linear, time-invariant system guaranteeing exponential stability at the origin, $\boldsymbol{y} = \boldsymbol{0}$. Observe in Figure 4.1 how both systems send trajectories to the origin from the initial point $\boldsymbol{y}_0 = [10\ 10]^T$.

This example is presented by Branicky[6,8] as a classical demonstration of how multiple Lyapunov functions can be used to develop switching laws for the system. It is intriguing in that, although individually stable, one cannot arbitrarily switch between dynamical modes and guarantee system stability. Branicky shows, for example, a switching law devised such that $u = 1$ when $\boldsymbol{y}$ is in Quadrants 2 and 4, and $u = 2$ when $\boldsymbol{y}$ is in Quadrants 1 and 3. From any point, the trajectory goes to infinity as illustrated in Figure 4.2(a). However, this is not the case for all switching functions. For example the law that switches modes when $\boldsymbol{y}$ crosses the line $y_2 = y_1$ results in a stable system converging on the origin (Figure 4.2(b)).



(a)



(b)

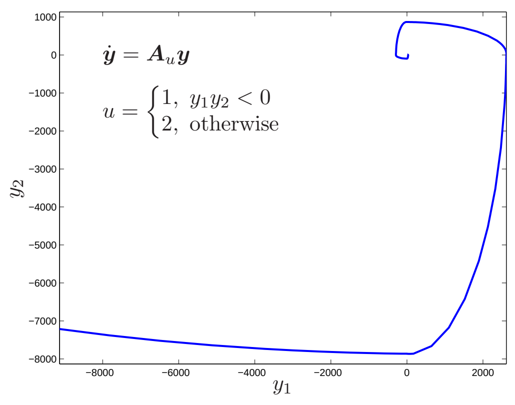
Figure 4.1: Individually Stable Systems

The characteristics of the system can be explained via Lyapunov analysis, which follows. The technique of multiple Lyapunov functions is intuitively applied since the switched system consists of multiple dynamical modes. Subsequently, the FSCT method is applied to demonstrate an alternative analysis technique for determining stable (and optimal) switching laws. This system presented in Equations 4.1-4.2 serves as an excellent example, since each method can be exercised in a graceful manner due to the inherent simplicity of the linear system. In addition, this example capitalizes on the familiarity of linear systems and Lyapunov stability theory to the general reader.

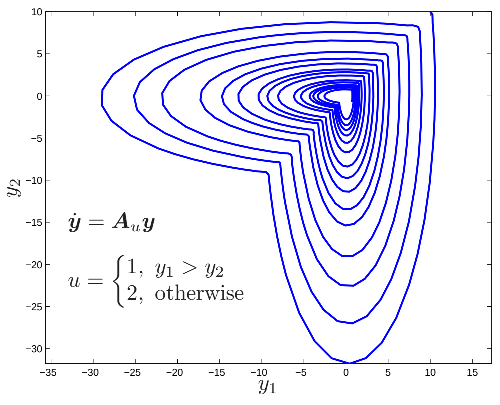
4.2.1 Stability via Multiple Lyapunov Functions

The key feature of the switching law of Figure 4.2(b) that guarantees stability is that the system remains in both modes for exactly one half of a revolution between each switch. Recall that the two state linear system with complex eigenvalues $\lambda_{1,2} = \alpha \pm j\omega$ and corresponding eigenvectors $\boldsymbol{v}_{1,2} = \boldsymbol{a} \pm j\boldsymbol{w}$ has solution of the form

$$\boldsymbol{y}(t) = e^{\boldsymbol{A}t}\boldsymbol{y}_0 = e^{\alpha t}\begin{bmatrix} \boldsymbol{a} & \boldsymbol{w} \end{bmatrix}\begin{bmatrix} \cos\omega t & \sin\omega t \\ -\sin\omega t & \cos\omega t \end{bmatrix}\begin{bmatrix} \boldsymbol{a} & \boldsymbol{w} \end{bmatrix}^{-1}\boldsymbol{y}_0.$$



(a)



(b)

Figure 4.2: Two Switching Laws

Then, one half revolution later from any point,

$$\begin{aligned} \boldsymbol{y}\left(t+\frac{\pi}{\omega}\right) &= e^{\alpha\left(t+\frac{\pi}{\omega}\right)}\begin{bmatrix}\boldsymbol{a}\ \boldsymbol{w}\end{bmatrix}\begin{bmatrix} \cos\omega\left(t+\frac{\pi}{\omega}\right) & \sin\omega\left(t+\frac{\pi}{\omega}\right) \\ -\sin\omega\left(t+\frac{\pi}{\omega}\right) & \cos\omega\left(t+\frac{\pi}{\omega}\right)\end{bmatrix}\begin{bmatrix}\boldsymbol{a}\ \boldsymbol{w}\end{bmatrix}^{-1}\boldsymbol{y}_0 \\ &= -e^{\alpha\frac{\pi}{\omega}}\boldsymbol{y}(t), \end{aligned}$$

provided that the system remains in the same mode over that time. Thus, for $\alpha < 0$ (a stable system), the Lyapunov function,

$$V = \boldsymbol{y}^T\boldsymbol{y},$$

which represents in a sense the energy of the system, is guaranteed to be smaller after one half of a revolution. Consistent switching at intervals of $\frac{\pi}{\omega}$ ensures an incremental decrease in system energy, resulting in convergence to the origin.

Other stable switching structures may also be obtained with a more classical Lyapunov argument. Considering each stable dynamical mode, $\boldsymbol{A}_u$, separately, there exist symmetric positive definite matrix pairs, $\boldsymbol{P}_u$ and $\boldsymbol{Q}_u$, such that

$$\boldsymbol{P}_u\boldsymbol{A}_u + \boldsymbol{A}_u^T\boldsymbol{P}_u = -\boldsymbol{Q}_u. \tag{4.3}$$

Stability for the mode is demonstrated through the Lyapunov function,

$$V_u = \boldsymbol{y}^T \boldsymbol{P}_u \boldsymbol{y} > 0,$$

with negative time derivative,

$$\begin{aligned} \dot{V}_u &= \boldsymbol{y}^T \boldsymbol{P}_u \dot{\boldsymbol{y}} + \dot{\boldsymbol{y}}^T \boldsymbol{P}_u \boldsymbol{y} \\ &= \boldsymbol{y}^T \left(\boldsymbol{P}_u \boldsymbol{A}_u + \boldsymbol{A}_u^T \boldsymbol{P}_u \right) \boldsymbol{y} \\ &= -\boldsymbol{y}^T \boldsymbol{Q}_u \boldsymbol{y} < 0. \end{aligned}$$

This standard analysis method offers a way of defining a stable switching law according to the behavior of the Lyapunov functions for each mode. For example, define $\boldsymbol{Q}_1 = \boldsymbol{Q}_2 = \boldsymbol{I}$ for simplicity. Then the Lyapunov Equation 4.3 can be solved uniquely to yield $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, corresponding to their respective modes. In this case, it is observed that regardless of the current mode, the energy of the system decreases according to $-\boldsymbol{y}^T \boldsymbol{Q}_u \boldsymbol{y} = -\boldsymbol{y}^T \boldsymbol{y}$. However, $V_1 \neq V_2$, and a reasonable switching law can be selected such that the Lyapunov function is minimized.[7] Thus,

$$u = \begin{cases} 1, & V_1 \leq V_2 \\ 2, & V_1 > V_2 \end{cases}.$$

A trajectory implementing this switching law is illustrated in Figure 4.3.

4.2.2 Optimal Switching via FSCT Method

The method of multiple Lyapunov functions demonstrated above can be effective in determining switching strategies between a finite number of system modes identified through a single decision variable. Variations on the theme arise by choosing the minimum $\dot{V}_u$ instead of V_u for some candidate Lyapunov functions, or by minimizing some combination of the two.[7] With an infinite set of stable switching laws, a question remains regarding efficiency and robustness. Although many criteria may be chosen to rank the effectiveness of a switching structure, a simple criterion is presently selected to demonstrate how the FSCT

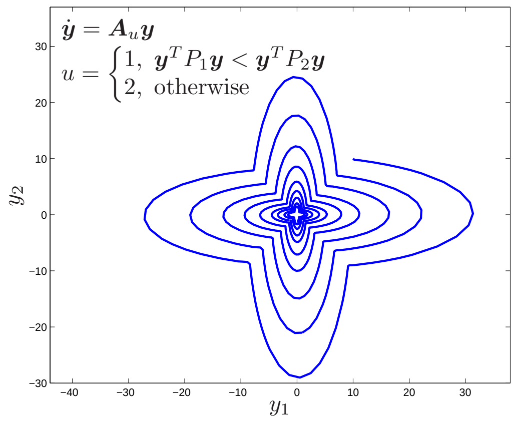


Figure 4.3: Minimum Lyapunov Function Switching Law

method can aid in the realization of an appropriate switching law. For this example, consider the objective of minimizing the time needed to move a point from its initial position to the vicinity of the origin. Naturally, the trajectory will never go through the origin, as $e^{\alpha t} > 0$ always. However, by choosing a region near the origin, a terminal condition for the optimal control problem is established. Let the final point be subject to

$$\boldsymbol{y}_f^T \boldsymbol{y}_f = 1,$$

such that the objective is to cross the boundary of the unit circle in minimum time, starting from the initial point, $\boldsymbol{y}_0 = [10\ 10]^T$. The optimal control law indicates when to switch between the dynamical modes of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ to most efficiently guide the trajectory to the terminal manifold.

The FSCT method is well equipped to solve this optimal control problem. Actually, many of the unique characteristics of the solution method are not exercised by this example due to the fact that the problem consists of only one decision variable. Notions of switching dependencies within the underlying nonlinear programming problem are necessarily excluded when only one control is involved. The total number of segments is exactly the number of pre-specified control values, and consequently, the control characteristics of each segment are known a priori. Thus, the optimization process simply determines appropriate switching times between segments.

To begin the process, a user selects the number of knots, indicating the total allowable control switches over the course of the trajectory. Let $n_k = 20$ knots for an initial optimization, and pre-specify control values such that

$$u_k^* = \frac{3}{2} + \frac{1}{2}(-1)^k,$$

indicating that u begins at the value 1 and alternates between 1 and 2 over each of $n_k+1 = 21$ segments. Additionally, a user selects a node count that sufficiently captures the state dynamics between control switches when state continuity conditions are satisfied. For this example, $n_n = 100$. Appropriate knot conditions are identified to ensure state continuity across segments, and the final condition is formulated as

$$c_{\psi_f} = (y_{1,n_n,n_s})^2 + (y_{2,n_n,n_s})^2 - 1 = 0,$$

which involves the states corresponding to the last node of the last segment. Finally, by identifying the optimization function, $\mathcal{J} = F(\boldsymbol{x}) = t_f - t_0$, the FSCT formulation is complete.

A preliminary guess is necessary to conduct the nonlinear optimization. The initial point, $\boldsymbol{x}_0$, is generated using an interpolation of the trajectory determined by the minimum Lyapunov function switching law of Figure 4.3 over the time interval $t \in [0\ 3]$. Thus, the preliminary cost of the optimization is 3, a reasonable estimate considering this trajectory first crosses the unit circle at time $t = 3.17$. The knots (switching times) between dynamical modes are uniformly spaced in time from 0 to 3. Thus, the preliminary guess does not satisfy the continuity constraints: the guessed control switches do not correspond to the control switches of the interpolated states. This is acceptable, as the optimization process ensures that the final solution is feasible as well as locally optimal.

An FSCT optimization applied for the selected initial guess leads to the trajectory illustrated in Figure 4.4(a). The final time is $t_f = 0.3782$, significantly smaller than the initial guess. The solution is feasible, and three control switches are clearly observable by the corners in the trajectory. With 20 knots, then, it is apparent that 17 possible control switches are not utilized. Indeed, the solution consists of many knots occurring simultaneously, resulting in zero-duration segments. Thus, the transcription is overparameterized for this solution. Observe that the control switches occur at the following states:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{k=1} = \begin{bmatrix} -9.3125 \\ 3.3180 \end{bmatrix}, \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{k=2} = \begin{bmatrix} 1.5297 \\ 4.2932 \end{bmatrix}, \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{k=3} = \begin{bmatrix} -1.7921 \\ 0.6385 \end{bmatrix}.$$

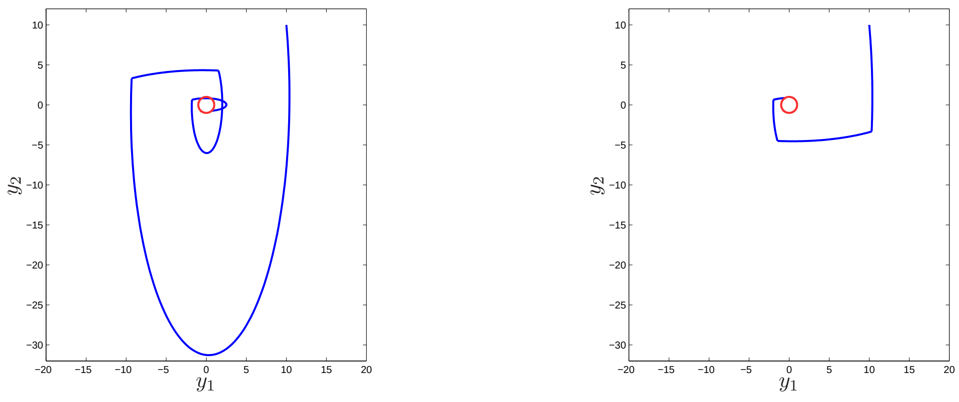
Notice that the switching points are related, as

$$-\left(\frac{y_1}{y_2}\right)_{k=1} = \left(\frac{y_2}{y_1}\right)_{k=2} = -\left(\frac{y_1}{y_2}\right)_{k=3} \equiv m.$$

This ratio *implies* the switching law

$$u = \begin{cases} 1, & -\frac{1}{m} \leq \frac{y_2}{y_1} \leq m \\ 2, & \text{otherwise} \end{cases} \tag{4.4}$$

where $m = 2.8067$. It is important to observe that the trajectory that resulted from the FSCT optimization shows control switches in only Quadrants 1 and 2, while the control law in Equation 4.4 observes switches in each of the four quadrants. The difference is explained through the realization that the solution method is capable of determining *locally* optimal solutions, most likely in the vicinity of the initial guess. Obviously, the solution is only a local minimum, as the trajectory actually crossed the terminal radius at one point before the final time. In this case, the initial guess and the parameterization leads to a solution with only three observable control switches. However, the symmetry of the trajectories generated in either dynamical mode imply that there should also exist symmetry in the switching law. This intuition leads to Equation 4.4. To validate this control law, a second optimization problem is solved, this time with a new initial guess. For the second optimization, the states and control switch times of the initial guess are generated using Equation 4.4. Optimizing



(a) (b)

Figure 4.4: FSCT Locally Optimal Switching Trajectories

this initial guess, the trajectory of Figure 4.4(b) is determined. Validating the control law, this second solution corresponds to the initial guess, *except* that in the final solution, $m = 3.0979$, a slightly larger slope for the switching line. However, the cost is even further improved, with $t_f = 0.0870$.

The fact that the slope value, m, changed between the two optimizations is not overly surprising. One reason for this is simply the fact that, in each case, the solution is a local, not global, minimum. Through further analysis, it is apparent that m is a factor of the initial point and the size of the terminal radius, as well. Indeed, a change to the terminal radius such that $\boldsymbol{y}_f^T \boldsymbol{y}_f = 0.5$ yields $m = 3.7786$ in the optimal solution.

The intent of this example is to demonstrate how a classical problem, which can be solved using traditional control techniques, can also be analyzed using the FSCT method. One advantage of the latter is the ability to optimize a control solution or control law according to a specified objective. In this case, the final time is minimized, however it might be equally useful to minimize the integral of the system energy over a fixed time, for example. Both costs capture, in a sense, the sentiment to drive a trajectory to the origin in an efficient manner, although both undoubtedly yield different solutions. It is observed, after all, that the trajectories of Figure 4.4 reach the unit circle quickly, but their control law does not guarantee that the trajectory will remain within that circle for all future time (it may escape the region and re-enter). Thus, the FSCT method can only guarantee optimality over the range of time considered, not beyond.

4.3 Lunar Lander

In a second example, it is useful to visit the classical Lunar Launch/Lunar Lander problem, commonly treated in the literature on optimal control theory.[23–25] The objective for the launch problem is to transfer a rocket from the lunar surface into a lunar orbit in minimum time. The problem is constructed in two dimensions, range and altitude, yielding 4 states (position and velocity in each dimension) and 1 control variable (thrust-direction angle). The thrust acceleration magnitude and the gravitational field are assumed constant. The rocket is initially at rest and must achieve a specified final altitude and range velocity. It is observed that the lunar lander problem (assuming a soft landing) is the identical problem, integrating backwards. The simplicity and familiarity of this problem make it an interesting test case for the FSCT method.

Consider the lunar lander problem with one added complexity: the vehicle cannot alter its thrust vector. Instead, the vehicle can thrust with constant acceleration magnitude in each principal direction. The system now has 4 states and 2 control variables, in a sense doubling the complexity of the Section 4.2 example. Specifically, by implementing multiple control values, the unique segment-switching characteristics of the FSCT method can be observed. The objective of this problem is to transfer a rocket from a lunar orbit to the lunar surface in minimum time or by using minimum acceleration. The dynamics are described by

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ u_1 \\ -g + u_2 \end{bmatrix},$$

where r, v, and u represent position, velocity, and control acceleration, respectively, and the subscripts indicate the horizontal and vertical dimensions. The gravitational constant of $g = 1.6231$ m/s^2 is obtained using the mass and equatorial radius of the Moon.[38] With initial conditions, $\boldsymbol{r}_0 = [200\ 15]^T$ km and $\boldsymbol{v}_0 = [-1.7\ 0]^T$ km/s and final conditions $\boldsymbol{r}_f = \boldsymbol{v}_f = \boldsymbol{0}$, the lander must achieve a soft landing on a specified target from a completely specified initial state. Both minimum-time and minimum-acceleration optimizations are realized with the finite set control constraints

$$\begin{aligned} u_1 &\in \{-\tilde{u}_1,\ 0,\ \tilde{u}_1\}, \\ u_2 &\in \{-\tilde{u}_2,\ 0,\ \tilde{u}_2\}, \end{aligned}$$

where $\tilde{u}_1 = 50$ m/s^2 and $\tilde{u}_2 = 20$ m/s^2. The control constraints ensure constant thrust acceleration during thrusting arcs.

4.3.1 Optimal Minimum-Time and Minimum-Acceleration Solutions

Optimal solutions are now demonstrated via the FSCT method. For this example, let $n_n = 5$ nodes per segment and $n_k = 14$ knots per control axis. In addition, let the pre-specified controls be identified as

$$u^*_{i,k} = \tilde{u}_i \cos\left(\frac{\pi}{2}(k-1)\right).$$

Thus, it is assumed in the control sequence that the vehicle thrusts initially in the positive directions (uprange and up), then coasts, then thrusts in the negative directions (downrange and down). The resulting optimizations determine the appropriate times for all control switches, indicating the durations for each thrusting and coasting arc.

An initial guess is devised with $t_0 = 0$, $t_f = 300$ seconds, and all knot times are evenly distributed over the interval such that each segment duration is identical. The state parameters in $\boldsymbol{x}$ are constructed to create a linear progression in each state from its initial value to its final value. Initial, final, and knot condition constraints are satisfied by the $\boldsymbol{x}$ supplied to the optimizer before the first iteration, but continuity constraints are not immediately satisfied. During the optimization process, $\boldsymbol{x}$ is improved such that *all* constraints are satisfied. In addition, the final $\boldsymbol{x}$ minimizes the objective function, representing $\mathcal{J} = t_f - t_0$ for minimum time or $\mathcal{J} = \int_{t_0}^{t_f} \boldsymbol{u}^T \boldsymbol{u}\, dt$ for minimum acceleration.

Figure 4.5 displays the solutions of both the minimum-time and minimum-acceleration problem. Vehicle positions and controls are plotted for both minimizations. Notice the control history u_1 for the minimum-time solution. In essence, this solution represents bang-bang control in the first axis, with $u_1(t) = -\tilde{u}_1$ on $t \in [0\ 33.66]$ seconds, and $u_1(t) = \tilde{u}_1$ for the remaining time until t_f, at 101.32 seconds. Of course, this control behavior is expected for a minimum-time optimization. Recall, however, that the pre-specified initial value for u_1 is $\tilde{u}_1$. As the illustration demonstrates, there is an instantaneous switch in the control at $t_0 = 0$ from $\tilde{u}_1$ to 0 and then from 0 to $-\tilde{u}_1$. The solution exhibits that $\Delta t_{1,1} = \Delta t_{1,2} = 0$ in order to accomplish this. In addition, there are instantaneous switches at $t = 34.06$, 68.65, and 101.32 seconds. At each of these times there exist time durations $\Delta t_{1,k}$ for coasting and negative-direction thrusting, and each has been optimized to be identically zero. This behavior is a common artifact of the FSCT formulation. It does not indicate that control switches should occur at these times; rather it indicates that the problem has been *overparameterized* with more knots than necessary. However, since control values are pre-specified in the optimization, it is useful to overparameterize the problem, allowing for more control switches than needed. Overparameterizing allows the optimizer to demonstrate the optimal number of switches (less than the parameterized number) by driving to zero superfluous control axis durations. The overparameterization also allows the user additional flexibility to arbitrarily pre-specify control values, knowing that non-optimal control values are eliminated in the final solution. In this case, specifying $n_k = 14$ knots represented an overparameterization in u_1, but not necessarily in u_2. In the vertical control axis, only three time durations are driven to zero by the optimization. These are the positive thrusting arc occurring at $t = 33.93$ seconds and the coasting and negative thrusting arcs occurring simultaneously at the final time.

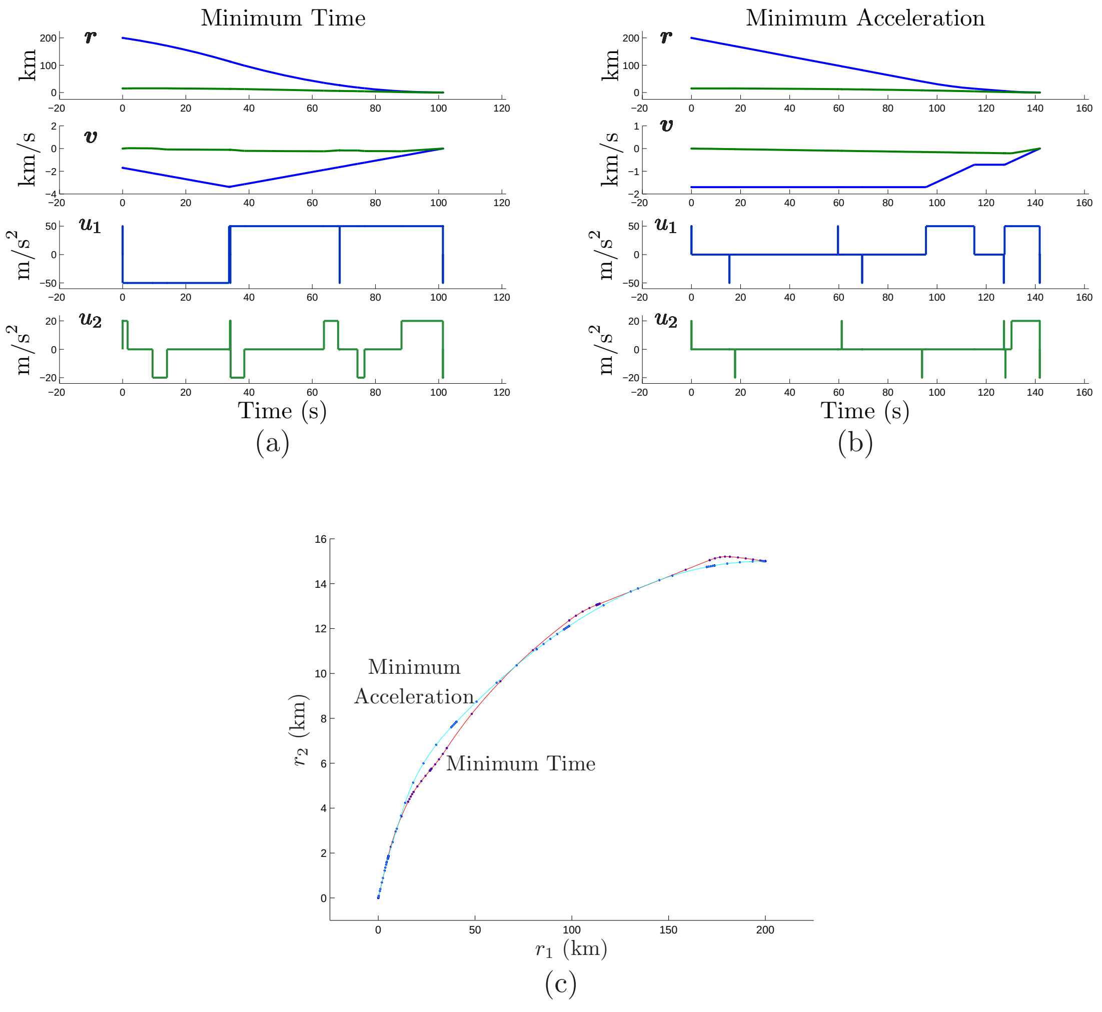


Figure 4.5: Optimal Solutions for the Minimum-Time and Minimum-Acceleration Lunar Lander Problem

This same behavior is observed for the minimum-acceleration optimization displayed in Figure 4.5(b). One may easily observe that most thrusting arcs are reduced exactly to zero by the optimizer for both control axes. This indicates that far fewer switches were necessary to identify this local minimum, and it provides confidence that the formulation has not underparameterized the problem by providing too few control switching opportunities.

Figure 4.5(c) plots the minimum-time and minimum-acceleration trajectories concurrently. For each trajectory, the dots indicate the actual values of the state-parameters optimized in $\boldsymbol{x}$. Thus, these are the states at the nodes along each segment. It is clear that the nodes are not evenly distributed spatially, nor are they distributed evenly in time. Again, this is due to the varying durations of each segment. Regardless, there are $n_n = 5$ nodes on each segment, and by segment they are evenly distributed in time. The lines between the nodes are not simply a connection of the dots. Rather, the lines indicate a propagation of the initial conditions along with their respective control solutions using a variable-step integrator. While the notions of overparameterization imply that enough knots are included in the parameterization, this illustration indicates the sufficiency of the node count, as the integrated trajectory matches nearly identically to the distributed nodes. The consistency between node locations and propagated states demonstrates the accuracy of the Hermite-Simpson integration equations. At the final time, state errors for both optimization solutions are $\mathcal{O}(10^{-6})$ m in positions and $\mathcal{O}(10^{-10})$ m/s in velocities, only slightly larger than the integration tolerances for the propagation.

An important discovery from the Lunar Lander example is the extent by which the FSCT method results in implementable control solutions. First, it is clear that the solution requires some ‘interpretation.’ Superfluous control switches must be discounted before implementing the control history. Actuators with minimum on-times do not support thrust durations approaching zero; however, within the tolerance of the optimization, zero or near-zero burn durations actually indicate that the respective actuation is not desirable.

Clearly, an optimization must be scaled properly in time to differentiate short actuation times from non-optimal control sequences.

Secondly, once a control solution is adequately interpreted, the performance of the solution in a continuous time setting can be nearly identical. Although this collocation technique does rely on a time discretization along each segment, the switching times between control values are optimized over a continuous spectrum. Therefore, the control solution represents exact switching times within the tolerance of the optimization. In this example, the continuous-time system is simulated with state and control propagation, showing negligible deviations from the FSCT solution.

Multiple Optimal Solutions

One final observation regarding overparameterization is now explored. Consider the case where the optimal solution requires that $\Delta t_{i,k} = 0$ for some i and k. Thus, $t_{i,k-1} = t_{i,k}$ and the pre-specified control value $u^*_{i,k}$ is not part of the optimal solution. In addition, let

$$u^*_{i,k-1} = u^*_{i,k+1}.$$

This is a common situation, as many applications may pre-specify control values to alternate between two values (or 'on' and 'off'). In this case, the control effectively remains at the value of $u^*_{i,k-1}$ from $t_{i,k-2}$ to $t_{i,k+1}$ (see Figure 4.6). Define $\bar{\Delta}$ as the total duration spent at this control value, such that

$$\bar{\Delta} = t_{i,k+1} - t_{i,k-2} = \Delta t_{i,k-1} + \cancelto{0}{\Delta t_{i,k}} + \Delta t_{i,k+1}.$$

Notice that $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ can take any values such that their sum remains $\bar{\Delta}$ while still ultimately communicating the same control profile.

Thus, there are an infinite number of combinations of parameters that represent an identical optimal solution. However, it is observed that changes to the two nonzero

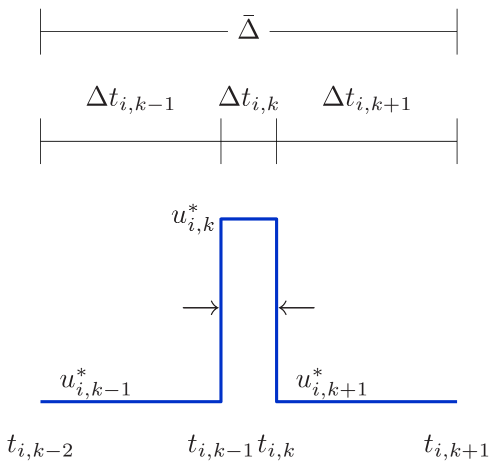


Figure 4.6: Scenario Resulting in Multiple Optimal Solutions

time durations have significant impact on the dynamic constraints, as the knot locations determined through $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ further determine the node locations at which states are defined in $\boldsymbol{x}_y$. This inertia deters arbitrary variations in time parameters, thus facilitating convergence. In other words, minor changes in the values of $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ require major changes in the elements of $\boldsymbol{Y}_{n_y,n_n,n_s}$ on the three corresponding segments in order to realize an equivalent optimal solution.

4.3.2 A Model Predictive Controller for Real-Time Implementation

One potential drawback of the FSCT method is that, while capable of producing optimal control histories for the finite set control problem, optimal control laws for real-time implementation are not immediately available. In Section 4.2, a control law is deduced for the optimal switching between dynamical modes, but for the general dynamical problem, there is no guarantee that an optimal control solution will imply a real-time law, $\boldsymbol{u} = \boldsymbol{u}(t, \boldsymbol{y})$. To compensate for this limitation, a process is now considered by which FSCT solutions may be implemented in conjunction with a model predictive control (MPC) design for real-time implementation of finite control. See Appendix A for relevant background on MPC theory and an extension to hybrid systems.

4.3.2.1 MPC Control Law

A model predictive controller can be derived by approximating the hybrid control system as a linear discrete-time model to develop easily calculated estimates for measureable state-like quantities at future instances of time. Let the estimate on the output at future time $t + j\Delta t$, given the states at time t, be denoted as $\hat{\boldsymbol{z}}(t + j\Delta t|t)$. In the appendix, it is shown that this estimate may be expressed as a function of $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$. Over a prediction interval defined by p, a series of estimates can be expressed as

$$\boldsymbol{Z} = \begin{bmatrix} \hat{\boldsymbol{z}}(t + \Delta t|t) \\ \vdots \\ \hat{\boldsymbol{z}}(t + j\Delta t|t) \\ \vdots \\ \hat{\boldsymbol{z}}(t + p\Delta t|t) \end{bmatrix} = \boldsymbol{G}\boldsymbol{y}(t) + \boldsymbol{K}\boldsymbol{u}(t),$$

with appropriate definitions for the matrices $\boldsymbol{G}$ and $\boldsymbol{K}$. Let the nominal output (corresponding to the FSCT trajectory) be expressed at the same discrete time intervals in the vector, $\boldsymbol{Z}_n$. A cost function of the form

$$\mathcal{J} = \frac{1}{2}(\boldsymbol{Z}_n - \boldsymbol{Z})^T \boldsymbol{Q} (\boldsymbol{Z}_n - \boldsymbol{Z}) + \frac{1}{2}\boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}$$

may be employed to penalize both deviations in trajectory states away from the nominal and excessive control usage. Thus, $\mathcal{J}$ can then be minimized according to user defined weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ to produce a tracking trajectory that also minimizes control over the prediction interval.

The model predictive control law for the hybrid system is defined according to

$$\boldsymbol{u} = \arg \min_{\boldsymbol{u} \in \mathbb{U}^{n_u}} \mathcal{J}.$$

Simply stated, the implemented control at the current time is the feasible control combination that minimizes the cost function. Thus, the control must consider each of the $\bar{m}$ possible control combinations, implementing the minimizing choice at each interval.

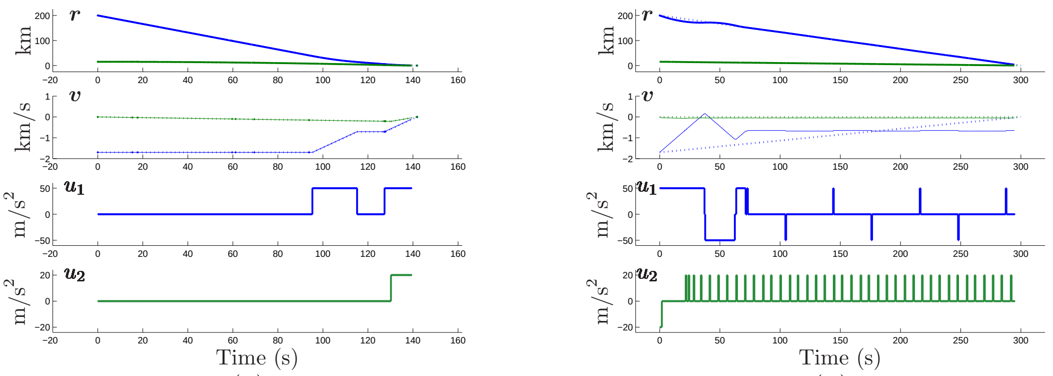
4.3.2.2 A Comparison of MPC Performance

The hybrid system model predictive controller is easily demonstrated in conjunction with the FSCT method using the minimum-acceleration solution of the lunar lander problem. With $\bar{m} = (3)(3) = 9$ possible control combinations, it is reasonable to assume that there exists a time interval, Δt, such that $\bar{m}$ evaluations of $\mathcal{J}$ can be compared to determine the minimizing control per interval. For this simulation, Δt is chosen such that there are 500 intervals between t_0 and t_f (less that four intervals per second for the minimum-acceleration trajectory). In addition, a prediction horizon is selected where $p = 10$, indicating that the controller calculates the estimates for the next 10 intervals of the output at each time step. In addition, a slightly modified cost function is evaluated, as

$$\mathcal{J} = \frac{1}{2}(\boldsymbol{Z}_n - \boldsymbol{Z})^T \boldsymbol{Q} (\boldsymbol{Z}_n - \boldsymbol{Z}) + \frac{1}{2}(\boldsymbol{u}_n - \boldsymbol{u})^T \tilde{\boldsymbol{R}} (\boldsymbol{u}_n - \boldsymbol{u}).$$

Here, the cost function penalizes deviations in the control from the FSCT solution, as opposed to penalizing control effort alone. Thus, the objective is to mimic the FSCT solution as close as possible via a real-time implementation. Since the a priori discovered optimal solution includes state and control values, it is logical to use all of this information in the controller. The weight matrices, $\boldsymbol{Q}$ and $\tilde{\boldsymbol{R}}$, are proportioned, however, to emphasize position state tracking over velocity or control tracking.

The results of a simulation implementing the real-time controller are depicted in Figure 4.7(a), displaying positions, velocities, and control values of the lunar lander. For



(a) (b)

Figure 4.7: Model Predictive Controller Simulations for the Lunar Lander

positions and velocities, both FSCT solution and MPC simulation states are plotted to demonstrate minimal deviations between the two. It is especially interesting to compare the control histories of Figure 4.5(b) and Figure 4.7(a): they are nearly identical. The primary observable difference between the MPC simulation and the FSCT solution is that the simulation has removed the instantaneous control switches that resulted from overparameterization in the FSCT formulation. Thus, with the FSCT solution in hand, it is possible to derive a real-time control law that very closely recreates the optimal trajectory.

As a point of comparison, a second simulation is conducted in which the MPC controller is implemented with a different nominal trajectory, shown in Figure 4.7(b). In this case, the nominal is identical to the initial guess imposed for both optimizations in Section 4.3.1. Here, the trajectory is truly arbitrary, and the associated control history corresponding to this trajectory is unknown. Furthermore, the velocity states do not satisfy a matching condition with the position states. Consequently, the cost function minimized in the control law is modified such that $\tilde{\boldsymbol{R}} = \boldsymbol{0}$ and elements of $\boldsymbol{Q}$ pertaining to velocity tracking are zero so that the control determination is solely based on position state deviations from the nominal. Note that for the arbitrary trajectory, reasonable position tracking is accomplished with this control law, while obeying the control constraints (finite set limitations) of the hybrid system. However, when comparing the control histories of both simulations, it is clear why it is beneficial to use the FSCT method for determining the nominal trajectory, since an arbitrary trajectory will necessarily require excessive control switches to minimize the MPC cost function. With a smaller value for Δt, control switches may be even more regular, requiring consideration of limitations such as minimum on-times for actuators.

The consistency between the FSCT minimum-acceleration solution and the hybrid system MPC simulation suggest the effectiveness of using the two methodologies in tandem. It is observed that the FSCT method offers control histories instead of implementable control laws. On the other hand, an MPC-derived controller may only be as good as the nominal trajectory selected for tracking. As a pair, however, it is possible to derive optimal trajectories and control histories, *and* implement them in a real-time context, where perturbations, modeling errors, and other unknowns are likely to arise. This example is intended to further illustrate the utility of the FSCT method when a control law, rather than a control history, is desired.

4.4 Small Spacecraft Attitude Control

In a final example, the FSCT method is applied to determine finite set control schedules for tracking an arbitary spacecraft orientation. The FSCT method is well-suited for the problem when only on-off actuation is available. This example is specifically motivated by spacecraft limited to commercial off-the-shelf actuator technologies that are inexpensive and readily available. The available literature[18–20,22] indicates a range of new thruster technologies for small spacecraft that are currently under development. Although these may offer wide ranges of thrust magnitudes and performance efficiencies, it is interesting to explore how the capability of traditional technologies can be stretched to maximize performance. The attitude control problem offers an exciting dynamic environment along with conceivable control limitations which make the FSCT method quite relevant.

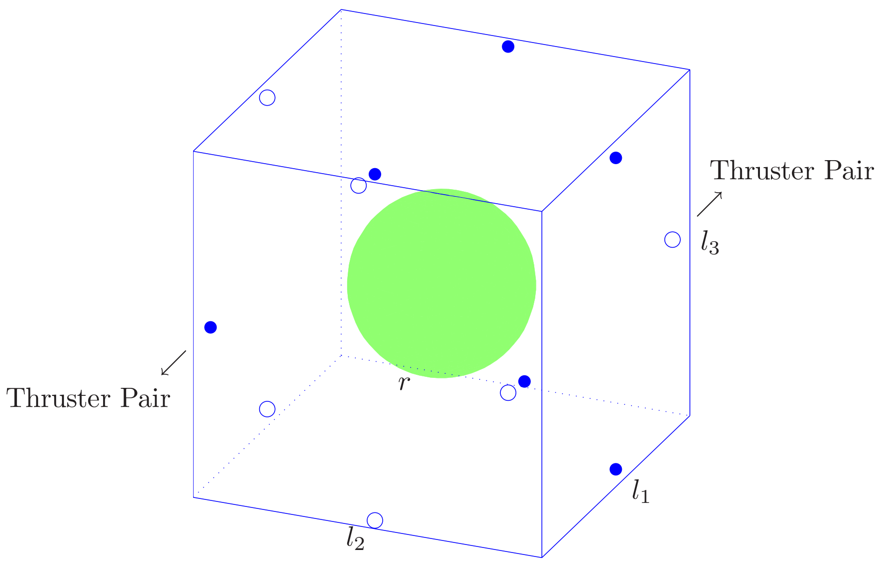


Figure 4.8: Micro-Satellite Illustration

Consider a low-cost micro-satellite employing a basic nitrogen cold gas propulsion system[39,40] for attitude control. Two scenarios are now investigated for this small spacecraft attitude control problem. In both of the scenarios, the spacecraft is equipped with six thruster-pairs situated on a cuboid spacecraft body to provide purely rotational control in each of the body's principal directions, positive and negative (see Figure 4.8). The propulsion system is supplied by a single N_2 propellant tank, centrally located in the spacecraft for simplicity. Temperature and pressure are regulated at each thruster nozzle to allow for constant thrust of 2 N. Pertinent specifications for the spacecraft and propulsion system are listed in Table 4.1.

The first scenario demonstrates the simplest control system to conceive. Each thruster pair is controlled by an on/off valve, and thrust magnitudes are limited to two values (2 N when on, 0 N when off). The second scenario explores a variable-thrust cold gas propulsion system in which the effective throat size of each thruster nozzle varies to alter propellant mass flow. However, the new problem can still be modeled in a finite set control formulation so that the FSCT method can be used. In transitioning between the two scenario formulations, it is suggested that *many* variable control problems are actually, at some level, finite control problems with an extended dynamic description.

For the dynamical relations that follow, it is necessary to identify the principal moments of inertia for the spacecraft. The spacecraft has dimension $l_1 \times l_2 \times l_3$ and a propellant tank with radius r. The cuboid volume, V_c, and propellant volume, V_p, are easily derived using these quantities. Assume a constant mass density within the propellant tank, and further assume a constant mass density in the remaining dry space of volume $V_d = V_c - V_p$. With dry mass, m_d, and propellant mass, m_p, it is possible to derive the principal moments. Let the cuboid mass, m_c, represent the total spacecraft mass if the mass density inside the tank were the same as the mass density outside. Then identify the unaccounted mass within the sphere as m_s, so that the principal moments for the spacecraft can be defined as:

$$\begin{aligned}
J_1 &= \frac{1}{12} m_c \left(l_2^2 + l_3^2\right) + \frac{2}{5} m_s r^2, \\
J_2 &= \frac{1}{12} m_c \left(l_1^2 + l_3^2\right) + \frac{2}{5} m_s r^2, \\
J_3 &= \frac{1}{12} m_c \left(l_1^2 + l_2^2\right) + \frac{2}{5} m_s r^2.
\end{aligned}$$

Specific definitions for derived quantities are also included in Table 4.1.

In each scenario, the dynamics are described using quaternion elements. Recall that the quaternion, $\boldsymbol{q} = \left[q_0 \; \boldsymbol{q}_v^T\right]^T$, is a 4-dimensional quantity, consisting of one scalar element and one 3-dimensional vector, that describes the rotation between two coordinate frames. The 3×3 coordinate transformation matrix used to rotate between two frames may be

Table 4.1: Physical Characteristics for a Micro-Satellite

Assumed Quantities				
Dimensions		l_1	1.00	m
		l_2	1.25	m
		l_3	1.50	m
		r	0.25	m
Masses	Dry Mass	m_d	15.00	kg
	Propellant Mass (at t_0)	m_p	5.00	kg
Cold Gas Propulsion	Specific Gas Constant (N_2)	R	296.80	$N \cdot m/(kg \cdot K)$
	Specific Heat (N_2)	γ	1.4	
	Storage Temperature	T	298.15	K
	Maximum Thrust	F_t	2.00	N
	Nozzle Throat Radius	r_t	2.50	mm
Derived Quantities				
Volumes	Cuboid Volume	V_c	$l_1 l_2 l_3$	
	Propellant Volume	V_p	$\frac{4}{3}\pi r^3$	
	Dry Volume	V_d	$V_c - V_p$	
Masses	Total Mass	m_t	$m_d + m_p$	
	Cuboid Mass	m_c	$m_d \ \frac{V_c}{V_d}$	
	Extra Sphere Mass	m_s	$m_t - m_c$	
Cold Gas Propulsion	Characteristic Velocity	c^*	434.439	m/s
	Exhaust Velocity	c	787.042	m/s
	Gas Density/Velocity Product at Throat	$\rho_t v_t$	129.42	$kg/(m^2 s)$

written as a function of the quaternion, as

$$\boldsymbol{C}(\boldsymbol{q}) = (q_0^2 - \boldsymbol{q}_v^T \boldsymbol{q}_v)\boldsymbol{I} + 2\boldsymbol{q}_v \boldsymbol{q}_v^T - 2q_0[\boldsymbol{q}_v \times]$$

where $[\boldsymbol{q}_v \times]$ is the cross product matrix,

$$[\boldsymbol{q}_v \times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}$$

which operates on a vector in the same way as a cross product. Thus, the quaternion and the coordinate transformation matrix are directly linked to describe the relationship between two reference frames. With no singularities, the quaternion is an effective means of representing attitude dynamics. For clarity, superscripts are used to describe the reference frames that a quaternion relates. For example, to transform a vector from reference frame

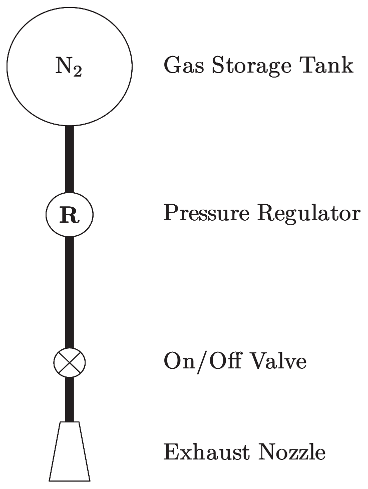


Figure 4.9: Fixed Thrust Propulsion

$\{\hat{\boldsymbol{a}}\}$ to reference frame $\{\hat{\boldsymbol{b}}\}$, the quaternion, ${}^{b}\boldsymbol{q}^{a}$, is used, where

$$\{\hat{\boldsymbol{b}}\} = \boldsymbol{C}({}^{b}\boldsymbol{q}^{a})\{\hat{\boldsymbol{a}}\}.$$

4.4.1 Low-Cost Cold Gas Thrusters: Fixed Thrust Attitude Control

Consider a micro-satellite with on-off actuation for its six attitude control thruster pairs. Each thruster delivers either 0 N or 2 N of thrust, depending on the state of each on/off valve. The basic propulsion system design is illustrated in Figure 4.9. Using the existing propulsion system, the objective in this scenario is to *track an arbitrary reference trajectory as well as possible, while minimizing fuel expenditure.*

Let the reference trajectory be described by the reference quaternion, ${}^{r}\boldsymbol{q}^{i}$, relating the inertial $\{\hat{\boldsymbol{\imath}}\}$ frame to the reference $\{\hat{\boldsymbol{r}}\}$ frame, while ${}^{r}\boldsymbol{\omega}^{i}$ indicates the angular velocity of the reference frame, written in the reference frame. With the initial quaternion, ${}^r\boldsymbol{q}^i{}_0 = [1\ 0\ 0\ 0]^T$, and angular velocity explicitly defined with respect to time as

$$
{}^r\boldsymbol{\omega}^i(t) = \begin{bmatrix} 0.3\cos t\left(1 - e^{-0.01t^2}\right) + (0.08\pi + 0.006\sin t)\, te^{-0.01t^2} \\ 0.3\sin t\left(1 - e^{-0.01t^2}\right) + (0.08\pi + 0.006\cos t)\, te^{-0.01t^2} \\ 1 \end{bmatrix} \text{rad/s}, \tag{4.5}
$$

the reference trajectory is completely specified, indicating the ideal attitude for the spacecraft at all times t. The reference angular velocity description is purely arbitrary, but it is observed that this reference trajectory offers interesting movement in angular position and velocity states over the fixed time $t \in [0\ 20]$ seconds and is an excellent test case for an FSCT optimization.

Define the state vector for this dynamical system according to

$$
\boldsymbol{y} = \begin{bmatrix} {}^b\boldsymbol{q}^i \\ {}^b\boldsymbol{\omega}^i \\ m_p \\ {}^r\boldsymbol{q}^i \\ p \end{bmatrix},
$$

where ${}^b\boldsymbol{q}^i$ indicates the quaternion relating the inertial frame and the spacecraft body frame, ${}^b\boldsymbol{\omega}^i$ is the corresponding angular velocity vector, and m_p is the depleting propellant mass when thrusters are activated. These first three components of $\boldsymbol{y}$ completely define the relevant states of the spacecraft. The remaining elements included in the state vector are strictly for computational convenience as these are useful for cost evaluation and control determination. The formulation allows the reference quaternion, ${}^r\boldsymbol{q}^i$, to be determined at each relevant instant in time by the FSCT method. The scalar state, p, measures an integral cost for deviations between reference and actual trajectories.

In the next step, the control vector is defined. Ultimately, $\boldsymbol{u}$ must indicate the position of the on/off valve for each of the twelve thrusters. Since thrusters, at a minimum, are assumed to act in pairs, it is logical to allow each control variable to indicate the valve position for at least two thrusters. However, it is also observed that each thruster pair has a corresponding thruster pair which acts in an opposing fashion such that their effects are cancelled when both pairs are on. Thus, consider the contol vector, $\boldsymbol{u} \in \mathbb{U}^3$ where

$$\mathbb{U} = \{-1, 0, 1\}.$$

Thus, $n_u = 3$, and each control variable is limited to three values, indicating for each principal axis, whether the positive-thrusting pair, the negative-thrusting pair, or neither is in the on position.

The state dynamics for the system are described by the following relations,

$$\dot{\boldsymbol{y}} = \begin{bmatrix} {}^b\dot{\boldsymbol{q}}^i \\ {}^b\dot{\boldsymbol{\omega}}^i \\ \dot{m}_p \\ {}^r\dot{\boldsymbol{q}}^i \\ \dot{p} \end{bmatrix} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) = \begin{bmatrix} \frac{1}{2}\boldsymbol{E}({}^b\boldsymbol{q}^i){}^b\boldsymbol{\omega}^i \\ -\boldsymbol{J}^{-1}\ {}^b\boldsymbol{\omega}^i \times \boldsymbol{J}{}^b\boldsymbol{\omega}^i + F_t\boldsymbol{J}^{-1}\boldsymbol{L}\boldsymbol{u} \\ -2\frac{F_t}{c}\sum_{i=1}^{3}|u_i| \\ \frac{1}{2}\boldsymbol{E}({}^r\boldsymbol{q}^i){}^r\boldsymbol{\omega}^i(t) \\ \left({}^b\boldsymbol{q}^i\right)^T \boldsymbol{H}^T\left({}^r\boldsymbol{q}^i\right)\boldsymbol{H}\left({}^r\boldsymbol{q}^i\right)\left({}^b\boldsymbol{q}^i\right) \end{bmatrix},$$

where $\boldsymbol{J} = \operatorname{diag}(J_1, J_2, J_3)$ is the inertia tensor, $\boldsymbol{L} = \operatorname{diag}(l_1, l_2, l_3)$ contains the spacecraft dimensions, and

$$\boldsymbol{E}(\boldsymbol{q}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \quad \boldsymbol{H}(\boldsymbol{q}) = \begin{bmatrix} q_1 & -q_0 & -q_3 & q_2 \\ q_2 & q_3 & -q_0 & -q_1 \\ q_3 & -q_2 & q_1 & -q_0 \end{bmatrix} = -\boldsymbol{E}^T(\boldsymbol{q}).$$

First, note that the quaternion dynamics, $\dot{\boldsymbol{q}}$, are a function of both $\boldsymbol{q}$ and $\boldsymbol{\omega}$. When relating $\{\hat{\boldsymbol{i}}\}$ to $\{\hat{\boldsymbol{b}}\}$, both vectors are contained within $\boldsymbol{y}$. When relating $\{\hat{\boldsymbol{i}}\}$ to $\{\hat{\boldsymbol{r}}\}$, however, the angular velocities are evaluated using Equation 4.5. Next, observe that the dynamics, ${}^b\dot{\boldsymbol{\omega}}^i$, are dramatically simplified by expressing angular velocities in the principal body frame, where the inertia tensor is an easily invertible diagonal matrix. In addition, one may see that the mass flow dynamic, $\dot{m}_p$, is nonzero only when one or more thruster pairs is on.

The cost dynamics, $\dot{p}$, are used for evaluating an integral cost. Here, it is desired to minimize deviations between the actual and reference coordinate frames. Consider the following relations:

$$\begin{aligned} \boldsymbol{C}({}^r\boldsymbol{q}^b) &= \boldsymbol{C}({}^r\boldsymbol{q}^i)\boldsymbol{C}^T({}^b\boldsymbol{q}^i), \\ {}^r\boldsymbol{q}_v{}^b &= \boldsymbol{H}\left({}^r\boldsymbol{q}^i\right)\left({}^b\boldsymbol{q}^i\right). \end{aligned}$$

Then if $\{\hat{\boldsymbol{b}}\}$ and $\{\hat{\boldsymbol{r}}\}$ are identical, ${}^r\boldsymbol{q}_v{}^b = \boldsymbol{0}$. A cost function that penalizes $\left({}^r\boldsymbol{q}_v{}^b\right)^T\left({}^r\boldsymbol{q}_v{}^b\right) > 0$ ensures minimal deviations between body and reference coordinate frames. This is equivalent to setting $p_0 = 0$ and minimizing p_f since

$$\begin{aligned} p_f - p_0 &= \int_{t_0}^{t_f} \dot{p}\, dt, \\ &= \int_{t_0}^{t_f} \left({}^r\boldsymbol{q}_v{}^b\right)^T \left({}^r\boldsymbol{q}_v{}^b\right)\, dt. \end{aligned}$$

The complete cost function weighs penalties on trajectory tracking deviations with the amount of propellant mass expelled in tracking the reference. Minimizing the total cost function,

$$\mathfrak{J} = \beta_1 p_f - \beta_2 m_{p_f} \tag{4.6}$$

is equivalent to minimizing tracking deviations and maximizing the final propellant mass when $\beta_1 > 0$ and $\beta_2 > 0$.

The problem is completely defined by identifying the remaining initial states for the optimization. Let the spacecraft begin along the reference trajectory. In this case, ${}^b\boldsymbol{q}_0^i = {}^r\boldsymbol{q}_0^i = [1\ 0\ 0\ 0]^T$ and ${}^b\boldsymbol{\omega}_0^i = {}^r\boldsymbol{\omega}^i(t_0)$. In addition, assume the initial propellant mass is $m_{p_0} = 5$ kg. These assumptions imply there is sufficient propellant available to achieve reasonable trajectory tracking for the interval from $t_0 = 0$ to $t_f = 20$ seconds.

FSCT Solution

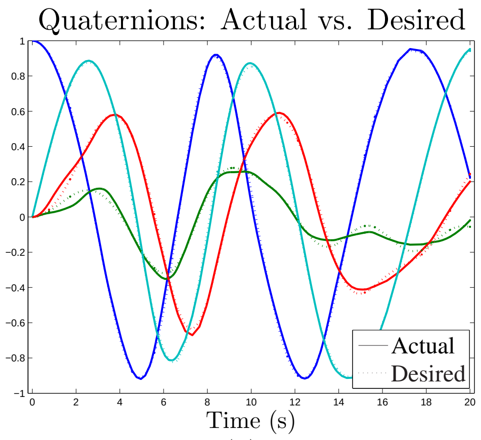
The fixed time optimal control problem detailed above is solved using the FSCT method to yield a feasible and locally optimal trajectory and control switching schedule. For this sample solution, the selected transcription parameters are $n_n = 5$ nodes per segment and $n_k = 20$ knots, allowing 20 control switches in each u_i over the time interval from t_0 to t_f. The pre-specified control values are selected based on the following law:

$$u^*_{i,k} = \cos\left(\frac{\pi}{2}(k-1)\right).$$

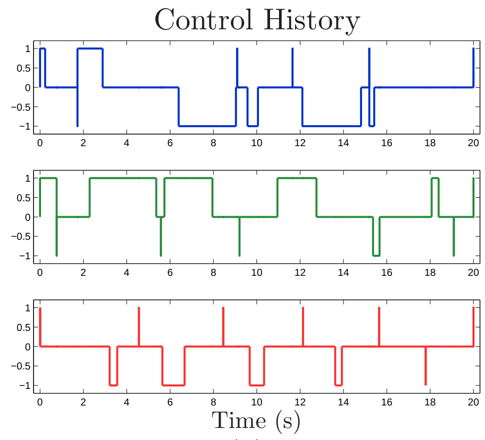
This control law alternates between positive-, zero- and negative-torque for each control variable. Clearly, the control sequence selection resembles that of the lunar lander problem in Section 4.3, as this seems to allow substantial flexibility to solve the underlying NLP problem.

The FSCT solution is depicted in Figure 4.10 when the cost function is set with equal penalty weights, $\beta_1 = \beta_2$. In Figure 4.10(a), the trajectory position (quaternion) histories for ${}^b\boldsymbol{q}^i$ and ${}^r\boldsymbol{q}^i$ are shown as the 'actual' and 'desired' trajectories, respectively. This illustration gives a visual sense of how well the trajectory can be tracked given finite value control limitations. In Figure 4.10(b), the resulting control history (switching schedule) is depicted. For each control variable, note that the durations of arcs associated with both $u_i = 1$ and $u_i = -1$ are reduced to zero. This indicates that the transcription formulation is not underparameterized.

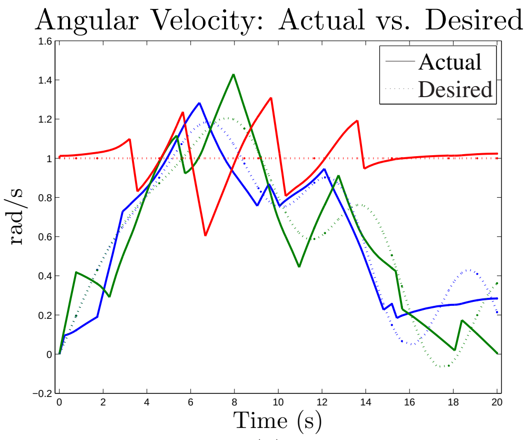
Finally, Figure 4.10(c) depicts the actual and desired angular velocities, ${}^b\boldsymbol{\omega}^i$ and ${}^r\boldsymbol{\omega}^i$. It is not unexpected that significantly more deviation is observable in this plot. Control restrictions clearly reduce the way in which velocity variables can change with time. More importantly, deviations in angular velocities are not penalized in the cost function, so the FSCT method does not attempt directly to drive velocities to match.



(a)



(b)



(c)

Figure 4.10: Fixed Thrust Attitude Control

4.4.2 Low-Cost Cold Gas Thrusters: Variable Thrust Attitude Control

The performance of the system above is clearly limited by on-off control actuation. Of course, fixed thrust control is an obvious choice when the intent is to apply the FSCT method to a real example. Indeed, the simplest, and perhaps least expensive, propulsion systems can benefit from the methodology for determining control strategies for reference tracking. A hybrid system model predictive controller, used in conjunction with FSCT solutions may be as successful in this case as it was for the lunar lander presented earlier. Here, however, another scenario is presented to expand the class of applications available to the FSCT methodology.

The most straightforward way of improving upon the solutions of the first attitude control scenario is to expand the solution space to include variable magnitude control inputs. This improves performance through better tracking, less fuel expenditure, or both. Thus, this scenario explores the possibility of a variable amplitude controller with a modified cold gas propulsion system. The purpose of the development that follows is to demonstrate that the variable control problem can *still* be interpreted, on a higher level, as a finite set control problem. Extrapolating further, many, if not most, dynamical systems with variable control inputs can be extended to reveal discrete components. Consider, for example, a control system whose varying inputs are determined by a digital computer. At the highest level, everything is reduced to a binary description, '0's and '1's, not unlike the discrete control inputs shown in the examples so far.

A previously developed variable amplitude cold gas propulsion system[41] serves as the inspiration for the following development. Here, the nitrogen propellant system is modified to allow variation in the effective dimension of the nozzle throat and, subsequently, the propellant mass flow. Consider the illustration of Figure 4.11. A valve core rod lies near the throat of the thruster nozzle, and has controlled motion up and down. Let the variable, d_i, indicate the position of the valve core rod for the i^{th} thruster. In addition, define r_t as

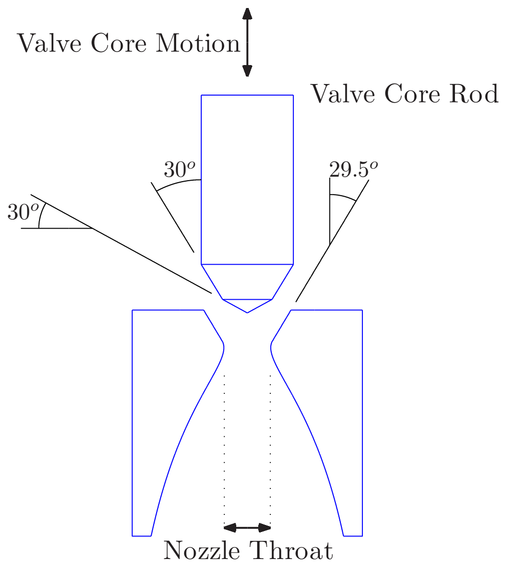


Figure 4.11: A Variable Amplitude Thruster Nozzle

the radius of the nozzle throat. The effective throat area is a function of the rod position, expressed as

$$A_t(d_i) = \pi r_t^2 - \pi \left(r_t - \frac{1}{2} d_i \right)^2 , \tag{4.7}$$

where $0 \leq d_i \leq 2r_t$. Note that if the rod position is such that $d_i > 2r_t$, no effect is expected on thruster performance, and $(A_t)_{\max} = \pi r_t^2$.

Because the throat area directly affects the mass flow through the nozzle (assuming constant propellant density and velocity), it has a direct effect on the magnitude of thrust. Assuming, as before, that the maximum thrust available is $(F_t)_{\max} = 2$ N, then

$$\rho_t v_t = \frac{(F_t)_{\max}}{(A_t)_{\max}\, c},$$

which can be evaluated using the constants in Table 4.1. Now, the amplitude of control for each thruster is a function of one discrete variable, indicating the position of the on/off valve, and one continuous variable, indicating the valve core rod position. To describe the dynamical system, it is necessary to understand how the rod position, d_i, is controlled. Surely, there are many ways of doing this, all affecting the nature of the dynamics. Assume then, for the sake of this argument, that each rod is driven by a constant-acceleration motor. Thus, the rod position and its velocity, v_i, are continuous variables, while its acceleration, a_i, may take only a discrete number of values.

If the valve core rod positions and velocities are included as state variables, a hybrid system ensues consistent with the formulation in Equation 1.1, with only continuous states and discrete controls. While this is not the only formulation for the variable amplitude control problem, this formulation demonstrates that it is possible to extend a variable amplitude control device into a combination of continuous states and discrete decision variables. In this case, states are defined for the physical elements that allow for thrust amplitude variations.

The state vector for this scenario, then, includes the same quantities as the previous scenario, now adding core rod positions and velocities to the set. Recall that, at a minimum, the twelve thrusters of the micro-satellite are combined into six thruster pairs. In the first scenario, two pairs providing torque along the same axis of rotation were considered together. In this scenario, the dynamic relations dictate that only thruster pairs can be considered to act in harmony. Thus, let the vectors, $\boldsymbol{d}$ and $\boldsymbol{v}$, which contain the individual rod dynamics, have six components each. Thus, a thruster pair shares the same rod positions and velocities to ensure that translational accelerations cancel at all times.

The state vector for the dynamical system takes the form,

$$\boldsymbol{y} = \begin{bmatrix} {}^b\boldsymbol{q}^i \\ {}^b\boldsymbol{\omega}^i \\ m_p \\ \boldsymbol{d} \\ \boldsymbol{v} \\ {}^r\boldsymbol{q}^i \\ p \end{bmatrix}.$$

The control vector is now

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{a} \end{bmatrix},$$

where $\boldsymbol{w}$ and $\boldsymbol{a}$ are each vectors composed of 6 elements (corresponding to the number of thruster pairs), where

$$w_i \in \{0, 1\}$$

indicates whether the i^{th} thruster pair is on or off, and

$$a_i \in \{-1, 0, 1\}$$

indicates the acceleration of the valve core rods of the i^{th} thruster pair, which can be negative, zero, or positive. The dynamics of the system are described by

$$\dot{\boldsymbol{y}} = \begin{bmatrix} {}^b\dot{\boldsymbol{q}}^i \\ {}^b\dot{\boldsymbol{\omega}}^i \\ \dot{m}_p \\ \dot{\boldsymbol{d}} \\ \dot{\boldsymbol{v}} \\ {}^r\dot{\boldsymbol{q}}^i \\ \dot{p} \end{bmatrix} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) = \begin{bmatrix} \frac{1}{2}\boldsymbol{E}({}^b\boldsymbol{q}^i){}^b\boldsymbol{\omega}^i \\ -\boldsymbol{J}^{-1}\,{}^b\boldsymbol{\omega}^i \times \boldsymbol{J}{}^b\boldsymbol{\omega}^i + \rho_t v_t c \boldsymbol{J}^{-1}\boldsymbol{L}\boldsymbol{A}(\boldsymbol{d})\boldsymbol{w} \\ -2\rho_t v_t \tilde{\boldsymbol{A}}^T(\boldsymbol{d})\boldsymbol{w} \\ \alpha_2 \boldsymbol{v} \\ \alpha_3 \boldsymbol{a} \\ \frac{1}{2}\boldsymbol{E}({}^r\boldsymbol{q}^i){}^r\boldsymbol{\omega}^i(t) \\ \left({}^b\boldsymbol{q}^i\right)^T \boldsymbol{H}^T\left({}^r\boldsymbol{q}^i\right)\boldsymbol{H}\left({}^r\boldsymbol{q}^i\right)\left({}^b\boldsymbol{q}^i\right) \end{bmatrix},$$

where the previously defined quantities $\boldsymbol{J}$, $\boldsymbol{L}$, $\boldsymbol{E}$, and $\boldsymbol{H}$ are unchanged, and

$$\begin{aligned}
\boldsymbol{A}(\boldsymbol{d}) &= \begin{bmatrix} A_t(d_1) & -A_t(d_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & A_t(d_3) & -A_t(d_4) & 0 & 0 \\ 0 & 0 & 0 & 0 & A_t(d_5) & -A_t(d_6) \end{bmatrix}, \\
\tilde{\boldsymbol{A}}(\boldsymbol{d}) &= \begin{bmatrix} A_t(d_1) & A_t(d_2) & A_t(d_3) & A_t(d_4) & A_t(d_5) & A_t(d_6) \end{bmatrix}^T, \\
A_t(d_i) &= \frac{1}{\alpha_1^2}\left[\pi r_t^2 - \pi\left(r_t - \frac{1}{2}d_i\right)^2\right]. \qquad (4.8)
\end{aligned}$$

Notice immediately that Equation 4.8 differs from Equation 4.7 by the scaling factor, α_1. Additional scaling factors, α_2 and α_3, are present in the valve core rod dynamics, as well, so that all state variables remain $\mathcal{O}(10^0)$ to improve the convergence of the underlying NLP problem. In this case, $\alpha_1 = 10^3$ so that r_t and d_i are presented in mm. Likewise, $\alpha_2 = 10^1$ so $\boldsymbol{v}$ is in 10^{-4} m/s, and $\alpha_3 = 10^0$ so that $a_i = 1$ indicates that the i^{th} rod is accelerating by 10^{-4} m/s^2.

Clearly, $\boldsymbol{A}(\boldsymbol{d})$ and $\tilde{\boldsymbol{A}}(\boldsymbol{d})$ represent the effective throat cross-sectional area for each thruster pair, listed in matrix form and vector form, respectively. These facilitate the new definitions for ${}^b\dot{\boldsymbol{\omega}}^i$ and $\dot{m}_p$. Thus, the effective control torque, evaluated by $\rho_t v_t c \boldsymbol{J}^{-1}\boldsymbol{L}\boldsymbol{A}(\boldsymbol{d})\boldsymbol{w}$ and measured in rad/s^2, as well as the total mass flow defined by $\dot{m}_p$, are determined by the current throat area and the state of the on/off valve.

The cost function for this scenario is identical to before.

$$\mathcal{J} = \beta_1 p_f - \beta_2 m_{p_f} \qquad (4.6)$$

Again, $\beta_1 = \beta_2$ to allow for a direct comparison between results.

FSCT Solution

In as many ways as possible, the optimization of the variable-thrust attitude control scenario is set up identically to the fixed-thrust scenario. The majority of the details, then, are not repeated. In this transcription formulation, $n_n = 5$ and $n_k = 20$ again, but with additional control variables ($n_u = 12$, instead of 3), the total number of segments, and thereby nodes, is significantly increased.

The pre-specified controls for the formulation follow a standard structure.

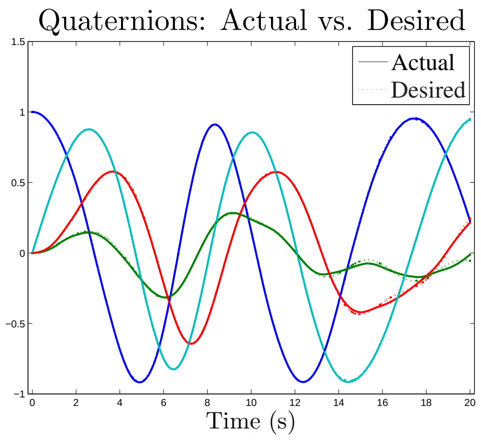
$$\begin{aligned} w^*_{i,k} &= \frac{1}{2} + \frac{1}{2}(-1)^{k-1}, \\ a^*_{i,k} &= \cos\left(\frac{\pi}{2}(k-1)\right) \end{aligned}$$

Notice that each w_i alternates between the values 1 and 0, while a_i alternates between 1, 0, and -1.

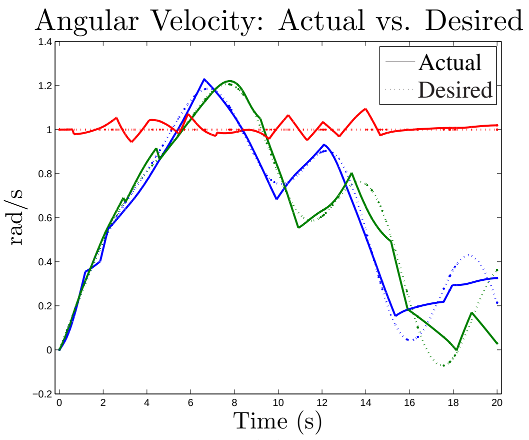
The results of the FSCT optimization are presented in full in Figures 4.12 and 4.13. Immediately, Figure 4.12(a-b) can be compared to Figure 4.9 to show how quaternions and angular velocities match the reference trajectories in each scenario. As expected, the variable thrust formulation offers more flexibility, and consequently better tracking, for the attitude control problem.

Figure 4.12(c-d) record the control histories that produce this trajectory. For each thruster pair, the controls indicate whether the thruster switch is on or off, and whether the motors driving the valve core rod are accelerating the rod. For completeness, Figure 4.12(e) shows the position history of the valve core rods for each thruster pair. Notice that the positions remain within the bounds $0 \leq d_i \leq 5$ mm, where the rod position has an effect on the resulting mass flow through the nozzle.

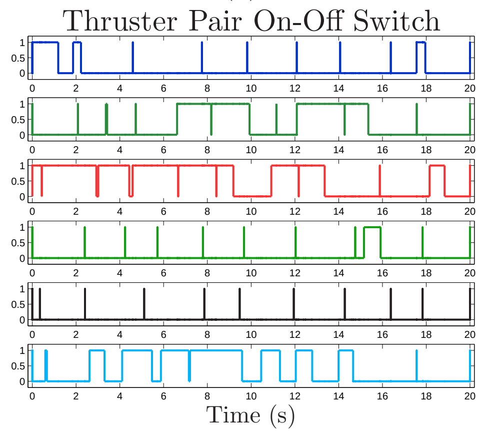
Figure 4.13 examines the effective control torque history for the system. When the effects of all of the finite value control variables are considered along with new dynamic states ($\boldsymbol{d}$ and $\boldsymbol{v}$), one can extract the actual control torque, measured in rad/s^2, that is applied to the spacecraft at any time. Figure 4.13(a) illustrates this. Note that the zero-duration 'dots' contained in the figure are artifacts of zero-duration segments that naturally result in an FSCT solution. For the sake of this discussion, they can effectively be ignored. As a comparison to this solution, Figure 4.13(b) illustrates the control torque



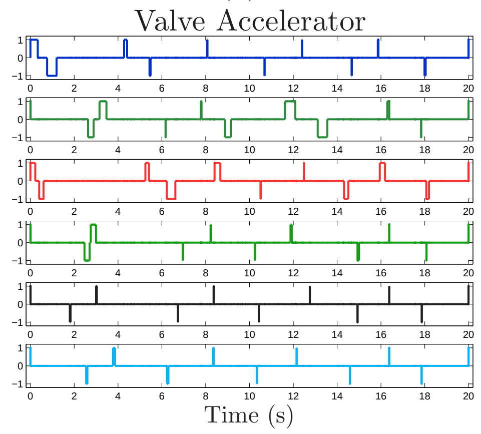
(a)



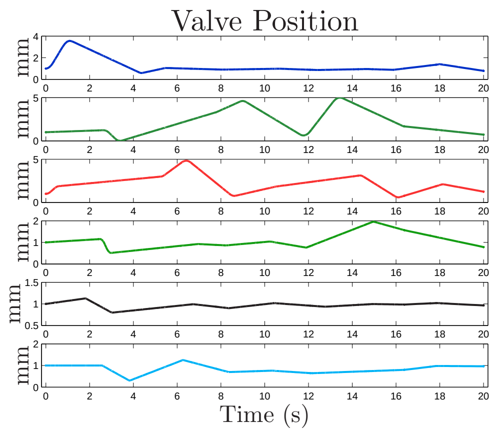
(b)



(c)



(d)



(e)

Figure 4.12: Variable Thrust Attitude Control

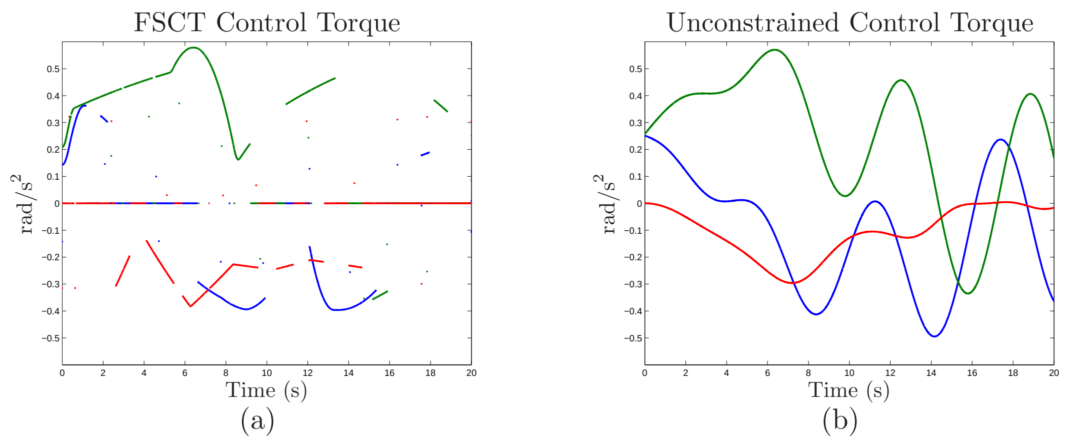


Figure 4.13: FSCT Variable Thrust Attitude Control Torque vs. Unconstrained Attitude Control Torque

for an *unconstrained control* system which tracks the reference trajectory perfectly. The unconstrained control torque is derived using a continuous Lyapunov-based control law which guarantees perfect tracking of quaternions and angular velocities (since initial states are along the reference). Some distinct similarities are easy to observe by looking at the plots together. Certainly, control torque magnitudes are similar, but there are also points at which the derived control torque from the finite set formulation very closely mimics the behavior of the purely continuous control.

This is viewed as a significant result which demonstrates how a detailed hybrid system formulation can approach a continuous formulation. While it is a step backwards to use a finite control formulation if control inputs are truly continuous, perhaps it is reasonable to argue that *many* systems, if not most, are truly hybrid systems, modeled as continuous. Often, it is easier to model the continuous system, as numerous methodologies exist for treating such systems. However, if a system has discrete components, it is ideal to treat them as such. Thus, the FSCT method offers an avenue for modeling maybe inevitably present discrete components, at whatever level they appear in the dynamics.

4.5 Summary

The Finite Set Control Transcription method is demonstrated for the determination of optimal solutions to hybrid control problems. The intent is to explore the range of applications of the FSCT method. Although many of the applications in this investigation are particularly relevant to aerospace engineering, the applicability of the method extends to all engineering disciplines.

Three example applications are used to provide context for the method. First, the FSCT method is used on a simple two-state system with two individually-stable dynamical modes. In a number of different sources, the method of multiple Lyapunov functions is utilized to treat the hybrid system with one decision variable. Here, results from the FSCT method are analyzed to demonstrate how optimal control laws may be extracted whose performance exceeds those derived using a Lyapunov argument.

Next a simple lunar lander problem is addressed by the FSCT method. A primary feature of the FSCT method is its ability to manage multiple independent decision inputs simultaneously. In this two-dimensional example, two control variables are included in the optimal control formulation to illustrate how the FSCT method can be applied when multiple independent control variables are considered. Solutions derived via the FSCT method are further utilized in conjunction with a hybrid system model predictive control scheme. For the hybrid system with a reasonable number of possible decision inputs at any given time, the MPC formulation offers real-time decision-making for the hybrid system. When the two methods are used in tandem, optimized control schedules can be realized in the context of potential perturbations or other unknowns.

Finally, the FSCT method tackles an attitude control problem presented in two different formulations. In the first, a small spacecraft is assumed to be limited to finite thrust magnitudes for a cold gas propulsion attitude control system. The second scenario explores a variable-thrust propulsion system, still modeled as a hybrid system. This investigation argues that even a system traditionally modeled with continuous control inputs may be more accurately described as a system ultimately relying on discrete decision variables. Continuous control variables may often be extended into a set of continuous state variables and discrete inputs. The scenario considers *how* an actuator may actually vary the control magnitude it applies. This process generates additional dynamics that can be modeled within the system. It is conceivable that, at some level, many systems can be thought of as hybrid systems with completely continuous states, and completely discrete control variables.

Chapter 5

Libration Point Formations

Relatively benign applications of the FSCT method are explored in the previous chapter. Although many benefits of the methodology are demonstrated using these academic examples, it is also important to illustrate the capability of solving more realistic, complex problems. This chapter is dedicated to such an application. Incidentally, the application presented here represents the motivation for much of the FSCT method's development.

This chapter and Appendix B address the topic of controlled formation flight of spacecraft near the libration points of the Sun-Earth/Moon system. The problem description and results (supported by background in the appendix) are detailed for readers interested in this application and related proposed future space missions. In-depth use of the FSCT method provides insight into the limitations of existing technologies that may guide the development of feasible mission requirements. First, motivation is presented for pursuing this application in the context of finite set control.

5.1 Motivation

Interest in space-based interferometry has motivated many investigations[42–57] into the feasibility of spacecraft formations near the libration points of the Sun-Earth/Moon system. A survey of many of these efforts is provided for refererence in Appendix B. The sensitivity of the dynamical regime near the libration points is apparent in the precise control requirements for non-natural formations derived in these investigations.

While some studies[18–22] focus on reducing the lower performance bounds on thruster technology, unconstrained control laws in this regime often require thrust levels *below* the limits of the technology presently available. The associated control accelerations, though small, cannot be neglected if precision formation keeping is desired.

Missions like TPF[15,16] and MAXIM[17] require precision tracking during data collection phases, often with the expectation that this be done in the absence of simultaneous actuation. Previous studies suggest this is an unrealistic expectation. At the same time, high precision tracking is achievable only through precise unconstrained control. This chapter employs the FSCT method to derive constrained control solutions for the purpose of determining realistic expectations for libration point formations in the context of the sensitive dynamic environment and the current state of propulsive technology.

Although unconstrained continuous control solutions offer perfect tracking to any trajectory, there is generally a disconnect when considering their implementation with actual hardware. To illustrate this, Figure 5.1 demonstrates an arbitrary continuous control solution in (a), with a proposed finite burn implementation in (b). The figure suggests that an actuator may not be able to produce precise thrust levels, and the continuous control solution is only approximated by some discrete thrust values. However, if an actuator limitation is known a priori, then it should be considered in the process of deriving a control solution.

Specific to the libration point formation control problem, it is probable that the control magnitudes of an unconstrained continuous solution may actually be smaller than the minimum thrust magnitude of the actuator (depicted in Figure 5.1(c)). In this case, the unconstrained solution is not useful. However, it is clear that feasible control solutions are necessarily restricted to a finite number of values, representing zero actuation and minimum-magnitude thrusting only. Therefore, a finite set control solution method is quite applicable for this scenario.

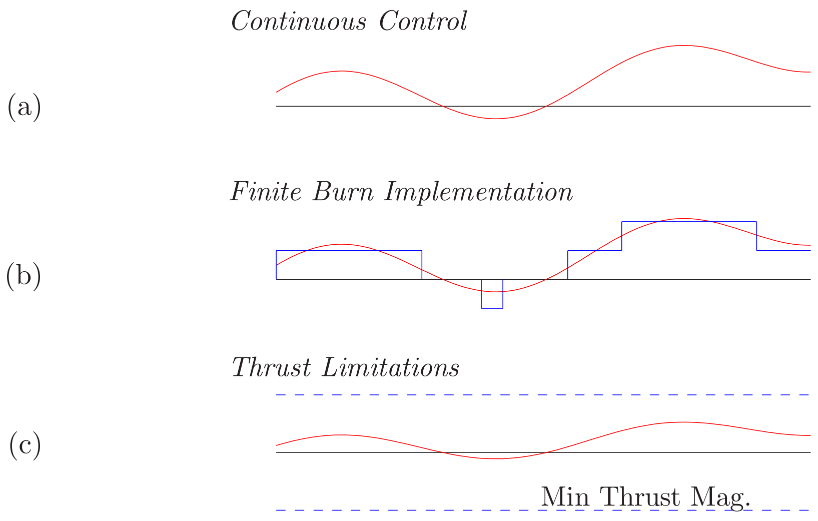


Figure 5.1: Implementing a Continuous Control Solution with Finite Burns

Thus, constrained finite burn solutions, achievable with currently available technology, are the focus of this chapter. Specifically, solutions are constrained to account for the minimum possible magnitudes available for thruster technology. See Appendix B for an extended discussion on the current propulsive capability, leading to the assumed values employed here.

Note that formation control problems are also constrained by mission requirements. Interferometry applications are of particular interest in the development of actuator constraints consistent with potential capabilities of member satellites, leading to significant spacecraft orientation restrictions. The restrictive—but realistic—assumptions applied here seek to establish reasonable expectations for tracking accuracy for libration point formations.

Organization

This chapter is organized as follows.

- Section 5.2 delivers the appropriate background and context for the libration point formation problem. The dynamics are developed for the Circular Restricted Three Body Problem (CR3BP) and several assumptions are presented to identify the appropriate constraints and objectives for an optimal control problem.
- Section 5.3 specifies how the Finite Set Control Transcription method is used and enhanced to treat this complicated problem.
- Section 5.4 presents a sample solution to illustrate the method's effectiveness.
- Section 5.5 presents the results from many optimizations intended to explore the solution space. Surveys are conducted for general formations to establish trends associated with formation pointing, and thrusting capability.

5.2 Problem Description

5.2.1 CR3BP Equations of Motion and the Reference Orbit

Assume that the motion of a formation of spacecraft is described relative to an artificial reference point, c, that evolves along a Halo orbit in the CR3BP. Conceptually, this point may represent a central 'chief' spacecraft in the formation. Subsequently, all other vehicles are referred to as 'deputies,' and their motion is determined relative to the state of the chief in the synodic rotating frame of the CR3BP.

5.2.1.1 Equations of Motion

Define an inertial frame, $\mathcal{I} \equiv \{\hat{\boldsymbol{x}}_{\mathcal{I}}, \hat{\boldsymbol{y}}_{\mathcal{I}}, \hat{\boldsymbol{z}}_{\mathcal{I}}\}$, with origin at the barycenter, B, of the two primary masses (in this case, the Sun and the Earth/Moon) such that the orthogonal

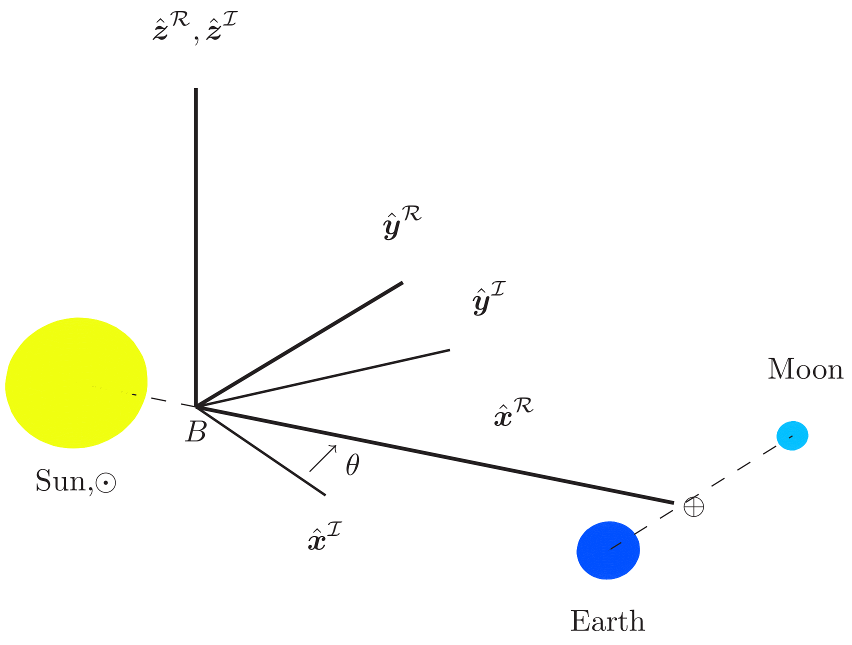

$\hat{\boldsymbol{z}}^{\mathcal{R}}, \hat{\boldsymbol{z}}^{\mathcal{I}}$
$\hat{\boldsymbol{y}}^{\mathcal{R}}$
$\hat{\boldsymbol{y}}^{\mathcal{I}}$
Moon
$\hat{\boldsymbol{x}}^{\mathcal{R}}$
B
θ
Sun,⊙
⊕
$\hat{\boldsymbol{x}}^{\mathcal{I}}$
Earth

Figure 5.2: CR3BP Frame

unit vectors $\hat{\boldsymbol{x}}_{\mathcal{I}}$ and $\hat{\boldsymbol{y}}_{\mathcal{I}}$ define the plane of motion of the primary bodies. Subsequently, the unit vector $\hat{\boldsymbol{z}}_{\mathcal{I}}$ is normal to the primary plane, and the primaries rotate about the $+\hat{\boldsymbol{z}}_{\mathcal{I}}$ axis.

In the CR3BP, it is convenient to describe the motion of a spacecraft in terms of a rotating coordinate system, $\mathcal{R}$, also centered at B, whose rotation coincides with the mean motion of the primaries. Let the orthogonal unit vectors $\{\hat{\boldsymbol{x}}_{\mathcal{R}}, \hat{\boldsymbol{y}}_{\mathcal{R}}, \hat{\boldsymbol{z}}_{\mathcal{R}}\}$ define the rotating frame, $\mathcal{R}$, where $\hat{\boldsymbol{z}}_{\mathcal{R}} = \hat{\boldsymbol{z}}_{\mathcal{I}}$, and $\hat{\boldsymbol{x}}_{\mathcal{R}}$ is directed from the Sun to the Earth/Moon barycenter. The remaining axis, $\hat{\boldsymbol{y}}_{\mathcal{R}}$, completes the right-handed system such that $\hat{\boldsymbol{y}}_{\mathcal{R}} = \hat{\boldsymbol{z}}_{\mathcal{R}} \times \hat{\boldsymbol{x}}_{\mathcal{R}}$. The $\mathcal{R}$ frame rotates about $\hat{\boldsymbol{z}}_{\mathcal{R}}$ at a rate of ω, equal to the mean motion of the primaries. The system is shown in Figure 5.2.

Let the position of the chief spacecraft, in terms of the $\mathcal{R}$-frame unit vectors, be defined as $\boldsymbol{r}_c^{\mathcal{R}} \equiv [x_c \; y_c \; z_c]^T$. In addition, assume that

$m_\odot$	$=$	mass of the Sun,
$m_\oplus$	$=$	mass of the combined Earth/Moon system,
$r_{\odot c}$	$=$	distance between the Sun and the chief, and
$r_{\oplus c}$	$=$	distance between the Earth/Moon barycenter and the chief.

Given the potential function,

$$U = \frac{(x_c^2 + y_c^2)\omega^2}{2} + \frac{Gm_\odot}{r_{\odot c}} + \frac{Gm_\oplus}{r_{\oplus c}}$$

the chief dynamics are described as,

$$\begin{aligned}
\begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \\ \ddot{z}_c \end{bmatrix} &= \begin{bmatrix} U_{x_c} \\ U_{y_c} \\ U_{z_c} \end{bmatrix} + 2\omega \begin{bmatrix} \dot{y}_c \\ -\dot{x}_c \\ 0 \end{bmatrix} + \begin{bmatrix} u_{x_c} \\ u_{y_c} \\ u_{z_c} \end{bmatrix} \\
&= -\frac{Gm_\odot}{r_{\odot c}^3} \begin{bmatrix} x_c + r_\odot \\ y_c \\ z_c \end{bmatrix} - \frac{Gm_\oplus}{r_{\oplus c}^3} \begin{bmatrix} x_c - r_\oplus \\ y_c \\ z_c \end{bmatrix} \\
&\quad + 2\omega \begin{bmatrix} \dot{y}_c \\ -\dot{x}_c \\ 0 \end{bmatrix} + \omega^2 \begin{bmatrix} x_c \\ y_c \\ 0 \end{bmatrix} + \begin{bmatrix} u_{x_c} \\ u_{y_c} \\ u_{z_c} \end{bmatrix},
\end{aligned} \tag{5.1}$$

where G is the gravitational constant, and $\boldsymbol{u}_c = [u_{x_c} \; u_{y_c} \; u_{z_c}]^T$ is the external control acceleration. Let the state of the chief be described by the vector

$$\boldsymbol{y}_c \equiv \left[\boldsymbol{r}_c^T \; \dot{\boldsymbol{r}}_c^T\right]^T = [x_c \; y_c \; z_c \; \dot{x}_c \; \dot{y}_c \; \dot{z}_c]^T .$$

Equation 5.1 is easily transformed into first order form,

$$\dot{\boldsymbol{y}}_c = \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{u}_c) = \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{0}),$$

where $\boldsymbol{u}_c = \boldsymbol{0}$ since the chief is assumed to evolve along a natural arc.

The general form of the motion in Equation 5.1 applies to the states of the l^{th} deputy, such that the state space representation is given by

$$\dot{\boldsymbol{y}}_{d_l} = \tilde{\boldsymbol{f}}(\boldsymbol{y}_{d_l}, \boldsymbol{u}_l).$$

The motion of the l^{th} deputy relative to the chief is determined first by identifying the relative state vector,

$$\boldsymbol{y}_l \equiv \boldsymbol{y}_{d_l} - \boldsymbol{y}_c. \tag{5.2}$$

Subsequently, differentiating Equation 5.2 implies that

$$\begin{aligned}
\dot{\boldsymbol{y}}_l &= \tilde{\boldsymbol{f}}(\boldsymbol{y}_{d_l}, \boldsymbol{u}_l) - \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{0}), \\
&= \tilde{\boldsymbol{f}}(\boldsymbol{y}_c + \boldsymbol{y}_l, \boldsymbol{u}_l) - \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{0}), \qquad (5.3) \\
&= \boldsymbol{f}(t, \boldsymbol{y}_l, \boldsymbol{u}_l). \qquad (5.4)
\end{aligned}$$

Notice that the functions $\tilde{\boldsymbol{f}}$ and $\boldsymbol{f}$ are not the same. It is apparent from Equation 5.3 that the relative dynamics of the l^{th} deputy depend on the absolute state of the chief, the relative state of the deputy, and the control effort exerted by the deputy. Through manipulation of the arguments in Equation 5.3, it is observed that the velocity terms related to the chief, $\dot{\boldsymbol{r}}_c$, cancel away. Since the chief satellite is assumed to follow a natural trajectory from some epoch and the path is predetermined, $\tilde{\boldsymbol{f}}$ reduces to $\boldsymbol{f}$ in Equation 5.4, which depends on time, the relative state of the deputy, and the external control of the deputy. Equation 5.4 is the dynamical model used to describe the motion of a deputy spacecraft.

Note that traditional formulations of the CR3BP equations employ nondimensional variables.[58] The formulation presented here, however, uses dimensional variables to allow increased flexibility in the scaling of the numerical algorithm.

5.2.1.2 Reference Halo Orbit

The chief spacecraft is assumed to evolve along an L_1 Halo Orbit, as determined in the Sun-Earth/Moon CR3BP. The initial state at the reference epoch is defined in $\mathcal{R}$-frame

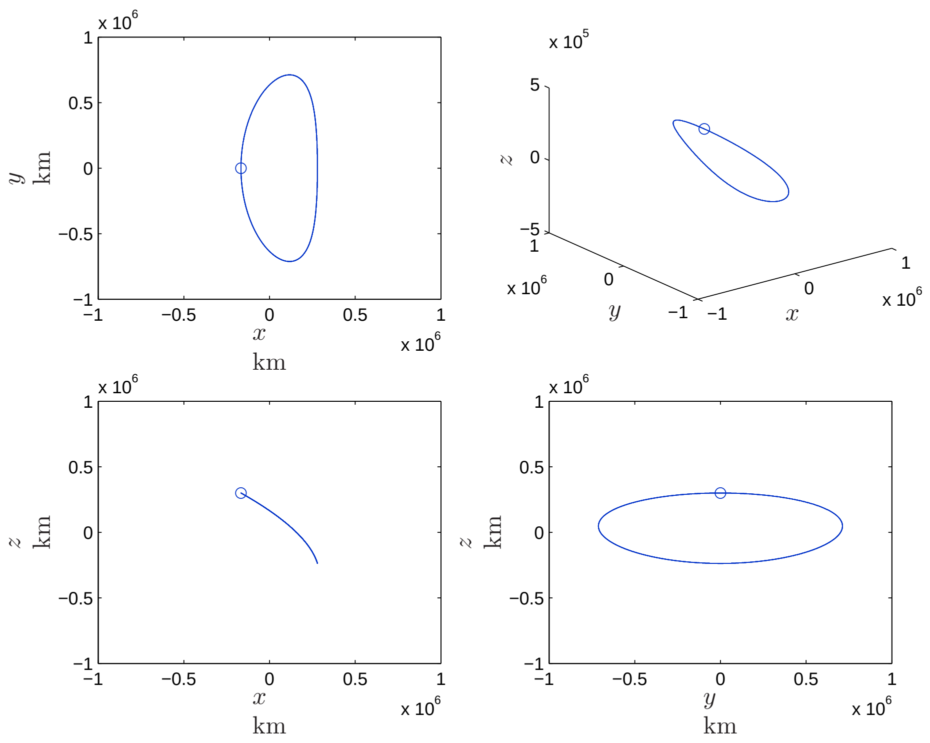


Figure 5.3: Reference Halo Orbit for Chief Spacecraft with Origin at L$_1$

coordinates as

$$\boldsymbol{r}_c(t_0) = \boldsymbol{r}_{L_1} + \begin{bmatrix} -166,783.32 \\ 0 \\ 300,000.00 \end{bmatrix} \text{ km} \tag{5.5}$$

$$\dot{\boldsymbol{r}}_c(t_0) = \begin{bmatrix} 0 \\ 281.28 \\ 0 \end{bmatrix} \text{ m/s.} \tag{5.6}$$

The position vector, $\boldsymbol{r}_{L_1}$, is the position of L$_1$ from the Sun-Earth/Moon barycenter, approximately 148×10^6 km in the $\hat{\boldsymbol{x}}_{\mathfrak{R}}$-direction. Thus, the Halo orbit begins at its northern most point, $300,000$ km north of the libration point. The Halo orbit is displayed in Figure 5.3. The trajectory shown takes place over one orbital period, approximately 178 days.

5.2.2 General Requirements for Interferometry Missions

Considering deep-space imaging formations as the application, it is reasonable to assume that constraints may be imposed on

- the size, shape, and orientation of the formation,
- the orientation of each member of the formation (deputy spacecraft),
- the thruster capability and thruster location on each spacecraft body.

Formationkeeping in the presense of all these constraints is naturally a difficult task. For example, although a specified thrust magnitude can be limiting by itself, when the thrust direction is also restricted by requirements on spacecraft orientation, precision control is *nearly* impossible. Certainly, not all of these requirements can be met simultaneously. Thus, the optimal control problem seeks to minimize deviations when constraints cannot feasibly be enforced. First, the requirements are formally identified. In Section 5.2.3, some are relaxed into objectives.

5.2.2.1 Formation Constraints

Define the scalar value $r^*_{cd_l}$ to be the required radial distance between the chief spacecraft (reference origin) and the l^{th} deputy spacecraft. In addition, the scalar value $r^*_{d_\lambda d_l}$ is defined as the generalized required distance between deputy λ and deputy l. Using the notation of relative states for the l^{th} deputy as

$$\boldsymbol{y}_l = \begin{bmatrix} \boldsymbol{r}_l \\ \boldsymbol{v}_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{d_l} - \boldsymbol{r}_c \\ \boldsymbol{v}_{d_l} - \boldsymbol{v}_c \end{bmatrix},$$

the distance between two deputies is $\boldsymbol{r}_{\lambda l} = \boldsymbol{r}_{d_l} - \boldsymbol{r}_{d_\lambda}$, and formation constraints may be expressed as

$$\begin{aligned} \boldsymbol{r}_l^T \boldsymbol{r}_l - \left(r^*_{cd_l}\right)^2 &= 0, \;\; l = 1, ..., n_d, \\ \boldsymbol{r}_{\lambda l}^T \boldsymbol{r}_{\lambda l} - \left(r^*_{d_\lambda d_l}\right)^2 &= 0, \;\; l = 1, ..., n_d - 1, \;\; \lambda = l + 1, ..., n_d, \end{aligned}$$

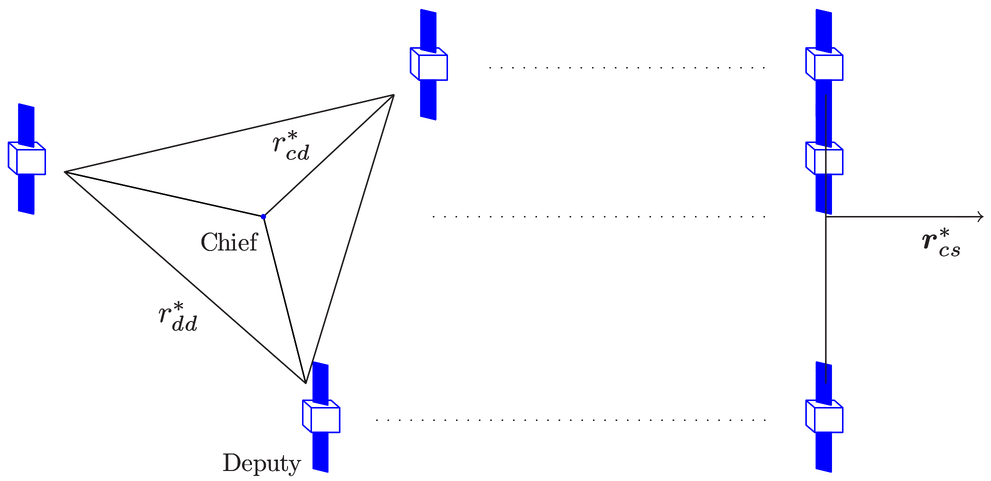


Figure 5.4: Formation Pointing

where n_d indicates the number of deputy spacecraft in the formation.

Furthermore, let $\boldsymbol{r}^*_{cs}$ define a vector pointing from the chief to a specified target point in space. If the target is sufficiently distant from the spacecraft, $\boldsymbol{r}^*_{cs}$ essentially points along an inertially fixed direction such that $\boldsymbol{r}^{*\mathcal{I}}_{cs}$ = constant. This is not a necessary restriction, however. In general, $\boldsymbol{r}^*_{cs}$ is written in the rotating frame as a function of time: $\boldsymbol{r}^{*\mathcal{R}}_{cs} = \boldsymbol{r}^{*\mathcal{R}}_{cs}(t)$. For imaging a distant point, the plane of the formation is assumed perpendicular to $\boldsymbol{r}^*_{cs}$, that is,

$$\boldsymbol{r}^T_{\lambda l}\boldsymbol{r}^*_{cs} = 0, \quad l = 1, ..., n_d - 1, \quad \lambda = l + 1, ..., n_d.$$

In Figure 5.4, both the formation size and orientation requirements are illustrated. Here, $n_d = 3$, and specified formation distances the same for each deputy force the formation to appear as an equilateral triangle. The formation may take on many configurations with enough deputy spacecraft and through proper definition of the formation distances.

5.2.2.2 Spacecraft Orientation

Let the orientation of the l^{th} deputy spacecraft be defined by the body-fixed coordinate frame,

$$\mathcal{B}_l \equiv \{\hat{\boldsymbol{x}}_{\mathcal{B}_l}, \hat{\boldsymbol{y}}_{\mathcal{B}_l}, \hat{\boldsymbol{z}}_{\mathcal{B}_l}\}$$

where the unit vectors, $\hat{\boldsymbol{x}}_{\mathcal{B}_l}$, $\hat{\boldsymbol{y}}_{\mathcal{B}_l}$, and $\hat{\boldsymbol{z}}_{\mathcal{B}_l}$ can be written in terms of any coordinate system. Most likely, they are expressed in the frame in which the states are integrated, i.e. the CR3BP rotating frame, $\mathcal{R}$.

For the subset of interferometry missions considered here, each deputy spacecraft points along a specified direction for the duration of the mission or phase of interest. Therefore, define the first axis of the body-fixed frame in the direction of the imaged point.

$$\hat{\boldsymbol{x}}_{\mathcal{B}_l} \equiv \frac{\boldsymbol{r}_{cs}}{r_{cs}} \tag{5.7}$$

Note that the scalar $r_{cs} = |\boldsymbol{r}_{cs}|$ normalizes the unit vector. This axis is normal to the face of the spacecraft containing an imaging payload. Now define the other coordinate axes as

$$\hat{\boldsymbol{y}}_{\mathcal{B}_l} \equiv \frac{\boldsymbol{r}_l \times \hat{\boldsymbol{x}}_{\mathcal{B}_l}}{|\boldsymbol{r}_l \times \hat{\boldsymbol{x}}_{\mathcal{B}_l}|}, \tag{5.8}$$

$$\hat{\boldsymbol{z}}_{\mathcal{B}_l} \equiv \hat{\boldsymbol{x}}_{\mathcal{B}_l} \times \hat{\boldsymbol{y}}_{\mathcal{B}_l}. \tag{5.9}$$

The vector $\hat{\boldsymbol{y}}_{\mathcal{B}_l}$ is roughly parallel to the direction of motion, and $\hat{\boldsymbol{z}}_{\mathcal{B}_l}$ is approximately aligned with the radial distance from the chief. This statement is based on the assumption of equal spacing among deputies. Note, however, that this coordinate system definition is valid even if the requirements specified by r^*_{cd} and r^*_{dd} are not met.

5.2.2.3 Spacecraft Design

For simplicity, assume that each deputy has a cube-shaped structure and is equipped with a translational thruster on each side, where the body frame, $\mathcal{B}_l$, is aligned with the principal axes of the spacecraft. The attitude of each spacecraft is independently controlled to maintain the proper alignment with the target.

It is not the intent of this work to suggest requirements on the structural design of a spacecraft. The assumptions made here are generalized for the problem under consideration. As a result, it is assumed the vehicles are capable of thrusting in all three body-fixed directions (although not necessarily at the same magnitudes). This configuration represents a simple way of implementing the thrusting requirements for the selected example. It is also one of the most limiting implementations. The dynamical model can be configured to consider any structural characteristics, but this implementation is a logical way of demonstrating the methods and the effects that highly constrained spacecraft orientations may have on performance.

5.2.2.4 Control Constraints

Consider the vector $\boldsymbol{u}_l$ that defines the control acceleration for the l^{th} spacecraft. In the body-frame,

$$\boldsymbol{u}_l^{\mathcal{B}} \equiv \begin{bmatrix} u_{\hat{\boldsymbol{x}}_{\mathcal{B}_l}} \\ u_{\hat{\boldsymbol{y}}_{\mathcal{B}_l}} \\ u_{\hat{\boldsymbol{z}}_{\mathcal{B}_l}} \end{bmatrix},$$

and the components of the vector are constrained to the values in the set $\mathbb{U} \equiv \{-T^*, 0, T^*\}$. Note that T^* indicates the minimum available thrust from the actuator, and this can be applied in each principal direction, positive or negative. This constraint must be imposed at all times, as this represents the true thrusting capability of the spacecraft.

5.2.3 Objectives for the Libration Point Formation Problem

If all of the requirements of Section 5.2.2 were implemented to determine a control solution, the resulting control problem would be overconstrained. It is impossible to limit the control to a finite number of values and still expect that relative distance and orientation requirements can be met. Instead, the libration point formation is treated as an optimal control problem. While control requirements are treated as constraints, some requirements are relaxed. Instead of enforcing them as constraints, deviations are minimized in the cost function.

Specifically, the spacecraft orientation relative to the formation and finite burn control limitations are treated as constraints. Thus, the resulting control options (directions and magnitudes) are completely specified, as if perfect attitude control is available to each spacecraft, and translational thrusters are only throttled to their minimum value (T^*). Consequently, formation size, shape, and orientation requirements are reduced to elements in the cost function. In addition, a cost is assigned to the duration of all burn segments to minimize the effort (thrusting) over the course of the trajectory. In general the cost function weighs three separate sets of costs.

$$\mathcal{J} = w_1\mathcal{J}_1 + w_2\mathcal{J}_2 + w_3\mathcal{J}_3$$

The weights, w_i, are selected to balance the costs of fuel and formation deviations. The cost indices are formulated as integrals of the following form:

$$\mathcal{J}_1 = \sum_{l=1}^{n_d} \int_{t_0}^{t_f} \boldsymbol{u}_l^T \boldsymbol{u}_l \, dt \tag{5.10}$$

$$\begin{aligned} \mathcal{J}_2 = & \sum_{l=1}^{n_d} \int_{t_0}^{t_f} \left(\boldsymbol{r}_l^T \boldsymbol{r}_l - \left(r^*_{cd_l}\right)^2 \right)^2 \, dt \\ & + \sum_{l=1}^{n_d-1} \sum_{\lambda=l+1}^{n_d} \int_{t_0}^{t_f} \left(\boldsymbol{r}_{\lambda l}^T \boldsymbol{r}_{\lambda l} - \left(r^*_{d_\lambda d_l}\right)^2 \right)^2 \, dt \end{aligned} \tag{5.11}$$

$$\mathcal{J}_3 = \sum_{l=1}^{n_d-1} \sum_{\lambda=l+1}^{n_d} \int_{t_0}^{t_f} \left(\boldsymbol{r}_{\lambda l}^T \boldsymbol{r}_{cs} \right)^2 \, dt \tag{5.12}$$

The cost, $\mathcal{J}_1$, is a quadratic cost on the thrust; $\mathcal{J}_2$ minimizes deviations in formation size and shape; and $\mathcal{J}_3$ minimizes deviations in the formation plane.

5.3 Unique Implementation Details

In Section 5.2 the libration point formation problem is described. The dynamics, constraints, and objective are identified. The problem can now be formulated to be solved using the FSCT method of Chapter 3. Certainly, the implementation concepts for constraints such as initial conditions, time constraints, and continuity constraints are unchanged. However, some characteristics of this problem require some enhancements to the method for successful implementation.

A primary consideration is that the formation problem has multiple dynamic bodies. Although the states of each body could be collected in a single state vector, $\boldsymbol{y}$, the key features of the FSCT method motivate an alternative where each dynamic body is treated separately. By doing this, the switching characteristics of multiple independent controls are still localized to the relevant dynamic body. Thus, the optimization parameters representing the controls $\boldsymbol{u}_l$ and $\boldsymbol{u}_\lambda$ do not interact, creating superfluous dependencies. This formulation decision has a resulting effect on the formulation of constraints and the objective function.

Some consideration is also necessary regarding the underlying NLP problem. Specifically, the size and the scaling of the parameters in $\boldsymbol{x}$ are addressed. When the problem is large (in this case, $n = \mathcal{O}(10^3)$), efficiency in calculations is crucial. Ensuring that all parameters are scaled on the same order of magnitude also improves efficiency.

Finally, due to the complicated nature of the dynamic model, it is worthwhile to directly express the derivative elements for the dynamics function, $\boldsymbol{f}$. Chapter 3 demonstrates how $\frac{\partial \boldsymbol{f}}{\partial t}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$ are implemented in the derivative calculations for continuity constraints. Now, specifics are developed for the calculation of these partials for this problem.

5.3.1 Transcriptions with Multiple Dynamic Bodies

The results of Chapter 3 are now extended for applications with multiple dynamic bodies. Let n_d designate the number of bodies, or in this case, deputy spacecraft in the formation. Thus, n_y indicates the number of states per body, and n_u the controls per body. The quantities n_n (nodes per segment), n_k (control switches), and $n_s = n_u n_k + 1$ (segments) are defined as before. However, control switches are defined independently, and consequently, segment boundary definitions differ per body.

With multiple bodies, the number of optimization parameters becomes

$$n = n_y n_n n_s n_d + n_u(n_k + 1)n_d + 2.$$

In this formulation, there are $n_y n_n n_s n_d$ state variables and $n_u(n_k + 1)n_d$ time durations. The initial and final times contribute the remaining parameters. The parameter vector is subsequently defined as,

$$\boldsymbol{x} = [\cdots\ y_{i,j,k,l}\ \cdots\ \cdots\ \Delta t_{i,k,l}\ \cdots\ t_0\ t_f]^T\,, \tag{5.13}$$

where the state indices are $i = 1, \ldots, n_y$, $j = 1, \ldots, n_n$, $k = 1, \ldots, n_s$, and $l = 1, \ldots, n_d$, and the time indices are $i = 1, \ldots, n_u$, $k = 1, \ldots, n_k + 1$, and $l = 1, \ldots, n_d$.

The constraint vector for this formulation is the same as that listed in Equation 3.6, except there are now multiple time constraints, $\boldsymbol{c}_t$. The time restriction must hold for each axis and for each spacecraft, so $n_t = n_u n_d$. The total number of constraints in $\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$ is

$$n_c = n_{\psi_0} + n_{\psi_f} + n_\beta n_n n_s n_d + n_y(n_n - 1)n_s n_d + n_y(n_s - 1)n_d + n_u n_d.$$

The sizes for the initial and final conditions, $\boldsymbol{c}_{\psi_0}$ and $\boldsymbol{c}_{\psi_f}$, are n_{ψ_0} and n_{ψ_f}, respectively. These dimensions are likely increased relative to the previous formulation since there are now n_d dynamic bodies. The path constraints, $\boldsymbol{c}_\beta$, are imposed at every node, along each segment, and for each body. The vector is sized accordingly. The dynamic constraints, $\boldsymbol{c}_{\dot{y}}$, are now of size $n_y(n_n - 1)n_s n_d$ to enforce continuity between the nodes for every state of every body along each segment. Continuity is enforced at the knots in $\boldsymbol{c}_s$ by $n_y(n_s - 1)n_d$ constraints for every state of every body. Notice that the number of transitions between segments equals the number of total knots, $n_s - 1 = n_u n_k$. Finally, there are $n_u n_d$ time constraints contained in $\boldsymbol{c}_t$ to ensure that all axis durations sum to $t_f - t_0$.

5.3.2 Splines for Data Available A Priori

Because of the size (numbers of parameters and constraints) of the transcribed optimization problem that results through collocation, it is important to limit overhead computing as much as possible. There are two places where this is done by providing existing data as an input into the optimization routines. Specifically, in this problem formulation, the trajectory of the chief spacecraft, which lies along a natural orbit, does not change. In addition, the designated pointing direction for the formation is also known, and does not need to be manipulated during the optimization process.

To reduce overhead computing, this data can simply be splined with sufficient interpolating points to provide the information necessary for solving the problem. Because the data from which splines are evaluated is not changing (they are input through a data file), the coefficients need only be evaluated once. It could be extremely costly (in computing time) to perform this operation on each iteration (or worse, each function evaluation). Careful implementation uses separate subroutines for determining the coefficients and for calling the coefficients to find an interpolated point. The former is accomplished before beginning the iterative optimization process.

5.3.2.1 Chief Spacecraft Position and Velocity

Recall that the dynamics of a spacecraft are presented relative to the chief spacecraft, and consequently, the position of the chief enters into the dynamics of Equation 5.4 through the imbedded function $\boldsymbol{r}_c(t)$. In addition, when time derivatives are calculated (to determine gradients), the function $\dot{\boldsymbol{r}}_c(t)$ also enters into the equations through

$$\frac{\partial \boldsymbol{f}}{\partial t} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{r}_c(t)} \frac{\partial \boldsymbol{r}_c(t)}{\partial t} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{r}_c(t)} \dot{\boldsymbol{r}}_c(t).$$

The trajectory described by $\left[\boldsymbol{r}_c(t)^T \dot{\boldsymbol{r}}_c(t)^T\right]^T$ most likely does not have an analytic solution. If, for example, the trajectory describes a periodic Halo orbit, it is best to provide data from which spline coefficients can be evaluated, so that both chief position and velocity can be extrapolated at any time since knot and node locations change throughout the optimization.

For this problem, enough data points are splined to cover more than a period of the periodic trajectory. This is more than enough, as the time duration of the optimization problems examined here is generally considerably less.

5.3.2.2 Deputy Spacecraft Pointing Direction

The pointing direction from the chief spacecraft to a specified point in space is $\boldsymbol{r}_{cs}^{*}$. In inertial space, this may be a constant vector, but not necessarily. Because this pointing direction is needed in the frame in which the dynamics are integrated, a general statement can be made that the pointing direction is a function of time, that is, $\boldsymbol{r}_{cs}(t)$. Likewise, it has a vector rate, $\dot{\boldsymbol{r}}_{cs}(t)$.

In many cases, these functions could have an analytic form. For example, if $\boldsymbol{r}_{cs}^{*\mathcal{I}}$ is constant, then $\boldsymbol{r}_{cs}^{*\mathcal{R}}(t)$ may be expressed in terms of sines and cosines. However, for the more general problem, where the pointing direction is arbitrarily specified, it is useful to treat the pointing in the same manner as the chief spacecraft position and velocity. For generality, then, the pointing direction and rate, used to calculate the coordinate transformation, ${}^{\mathcal{R}}\boldsymbol{C}^{\mathcal{B}_l}$, and its rate, are stored as interpolating points to generate spline coefficients.

5.3.3 Scaling

NLP algorithms are more efficient when all of the parameters of $\boldsymbol{x}$ are on the same order of magnitude. Most of the problems addressed in this investigation are already sufficiently scaled. That is, all parameters are on $\mathcal{O}(10^0)$. However, the sensitive dynamic environment near libration points requires some unique scaling considerations. Specifically, as opposed to evaluating dynamic equations in conventional units, such as km and sec, a good practice is to define a new distance unit, DU, and time unit, TU, such that, in the new units, the states and times are along the same order of magnitude.

For example, consider a formation with a required relative distance between chief and deputies approximately 1 km. A relative position may be $[0.707, 0, 0.707]$. If the distance unit is defined as

$$1 \text{ DU} \equiv 100 \text{ m},$$

then the position states will generally range in magnitude from 1 to 10 DU. With relative velocities less than 1 μm/s, it is logical to define

$$1 \text{ TU} \equiv 10^6 \text{ sec}$$

to ensure that velocity states range in magnitude from 1 to 10 DU/TU. The time unit is now approximately 10+ days, and it is reasonable to run optimal control problems with duration of approximately 1/3 of a year to keep the time durations within a general magnitude range less than, or close to, 10 TU.

5.3.4 Objective Implementation

In Section 5.2.3, the objective function is described as a summed cost penalizing accumulated thrust, formation size and shape requirement deviations, and formation pointing requirement deviations. When input as a function of the parameters $\boldsymbol{x}$, the cost function takes the form,

$$F(\boldsymbol{x}) = w_1 F_1(\boldsymbol{x}) + w_2 F_2(\boldsymbol{x}) + w_3 F_3(\boldsymbol{x}).$$

Although a parameterized thrust cost is easily derived, the formation deviation costs are more difficult to evaluate. Primarily, these are a function of the position-states associated with each of the n_d dynamic bodies. Unfortunately, these positions are only known at the nodes, and the locations of the nodes are different for each body. That is, the times for the

j^{th} node of the k^{th} segment are not necessarily the same for each body, or

$$t_{j,k,l} \neq t_{j,k,\lambda}.$$

This is overcome through interpolation. The implementation of the parameterized subcosts follow.

5.3.4.1 Thrust

The control in each axis of each spacecraft is broken into $n_k + 1$ axis segments, whose durations are $\Delta t_{i,k,l}$. In addition, the control values are pre-specified for each segment. For the solutions derived in this chapter, the controls are pre-specified a priori as

$$u^*_{i,k,l} = T^* \cos\left(\frac{\pi}{2}(k-1)\right),$$

where $i = 1, \ldots, n_u$, $k = 1, \ldots, n_k + 1$, and $l = 1, \ldots, n_d$. Thus, the control cost is simply,

$$F_1(\boldsymbol{x}) = \sum_{l=1}^{n_d} \sum_{k=1}^{n_k+1} \sum_{i=1}^{n_u} \Delta t_{i,k,l} \left(u^*_{i,k,l}\right)^2.$$

5.3.4.2 Formation Size and Shape Deviation

For the formation costs, added complexities stem from the node distribution from t_0 to t_f. Specifically, the node distribution varies for each spacecraft, yet the deviation in the shape of the formation depends on the position of each spacecraft at a given time. A simple solution is to perform an interpolation of the nodes for each spacecraft. Using interpolated values, a representative number of points over the trajectory is selected to approximate the integrals of Equation 5.11. Define the interpolated set of states as $\boldsymbol{y}_{j,l}$ where $j = 1, ..., n_g$, and n_g is sufficiently large to generate a uniformly spaced grid of points that describe the trajectories of the $l = 1, ..., n_d$ spacecraft. The interpolated states are distinguished from those in Equation 5.13 by not having a k subscript, as they are uniformly distributed between t_0 and t_f independent of knot locations. Then

$$\begin{aligned}F_2(\boldsymbol{x}) &= \sum_{l=1}^{n_d}\sum_{j=1}^{n_g-1}\frac{t_{j+1}-t_j}{2}\left\{\left(\left[\sum_{i=1}^{3}y_{i,j,l}^2\right]-\left(r^*_{cd_l}\right)^2\right)^2+\left(\left[\sum_{i=1}^{3}y_{i,j+1,l}^2\right]-\left(r^*_{cd_l}\right)^2\right)^2\right\}\\ &+\sum_{l=1}^{n_d-1}\sum_{\lambda=l+1}^{n_d}\sum_{j=1}^{n_g-1}\frac{t_{j+1}-t_j}{2}\left\{\left(\left[\sum_{i=1}^{3}\left(y_{i,j,l}-y_{i,j,\lambda}\right)^2\right]-\left(r^*_{d_\lambda d_l}\right)^2\right)^2\right.\\ &\left.+\left(\left[\sum_{i=1}^{3}\left(y_{i,j+1,l}-y_{i,j+1,\lambda}\right)^2\right]-\left(r^*_{d_\lambda d_l}\right)^2\right)^2\right\}\end{aligned}$$

computes deviations from specified distances between chief-deputy and deputy-deputy pairs.

5.3.4.3 Formation Plane Deviation

The cost associated with deviations in the formation plane exhibits the same complexities as the formation shape cost. Once again, this difficulty is overcome through the use of interpolated points, $\boldsymbol{y}_{j,l}$. The resulting cost index is

$$\begin{aligned}F_3(\boldsymbol{x}) &= \sum_{l=1}^{n_d-1}\sum_{\lambda=l+1}^{n_d}\sum_{j=1}^{n_g-1}\frac{t_{j+1}-t_j}{2}\left\{\left(\left[\sum_{i=1}^{3}\left(y_{i,j,l}-y_{i,j,\lambda}\right)r^*_{(cs)_i}(t_j)\right]\right)^2\right.\\ &\left.+\left(\left[\sum_{i=1}^{3}\left(y_{i,j+1,l}-y_{i,j+1,\lambda}\right)r^*_{(cs)_i}(t_{j+1})\right]\right)^2\right\}.\end{aligned}$$

This cost function uses a trapezoidal approximation of the integral of Equation 5.12 to accumulate penalties in plane deviation.

5.3.5 Constraint Implementation

Finally, some considerations of the libration point formation constraints are addressed. Recall that formation requirements (size, shape, and orientation) cannot be achieved throughout the entire time interval $[t_0\ t_f]$. However, it is possible to ensure that at t_0 and t_f, the formation requirements are satisfied. The solutions obtained in this chapter employ initial condition constraints that specify the initial states of each deputy such that the formation requirements are met. For the states at t_f, however, it is overly restrictive to completely specify their exact values. Instead, formation constraints are implemented at the final time only. The form of these constraints is below.

Additionally, this section outlines the partial derivatives of the dynamics function, a necessary input for calculating derivatives of the continuity constraints analytically.

5.3.5.1 Final Formation Size and Shape

Let the specified distance between the chief and the lth deputy be $r^*_{cd_l}$ and the specified distance between spacecraft l and spacecraft λ be $r^*_{d_\lambda d_l}$. Then the number of final formation constraints is

$$n_{\psi_f} = n_d + \left(\begin{array}{c} n_d \\ 2 \end{array} \right)$$

and the constraints have the form

$$\boldsymbol{c}_{\psi_f}(\boldsymbol{x}) = \left[\begin{array}{c} \left[\sum_{i=1}^{3} y^2_{i,n_n,n_s,1}\right] - \left(r^*_{cd_1}\right)^2 \\ \vdots \\ \left[\sum_{i=1}^{3} y^2_{i,n_n,n_s,l}\right] - \left(r^*_{cd_l}\right)^2 \\ \vdots \\ \left[\sum_{i=1}^{3} y^2_{i,n_n,n_s,n_d}\right] - \left(r^*_{cd_{n_d}}\right)^2 \\ \left[\sum_{i=1}^{3} \left(y_{i,n_n,n_s,1} - y_{i,n_n,n_s,2}\right)^2\right] - \left(r^*_{d_2 d_1}\right)^2 \\ \vdots \\ \left[\sum_{i=1}^{3} \left(y_{i,n_n,n_s,l} - y_{i,n_n,n_s,\lambda}\right)^2\right] - \left(r^*_{d_\lambda d_l}\right)^2 \\ \vdots \\ \left[\sum_{i=1}^{3} \left(y_{i,n_n,n_s,n_d-1} - y_{i,n_n,n_s,n_d}\right)^2\right] - \left(r^*_{d_{n_d} d_{n_d-1}}\right)^2 \end{array} \right].$$

The Jacobian elements for the formation constraints can be written as

$$\frac{\partial c_{\psi_{f_l}}}{\partial x_\gamma} = \left\{ \begin{array}{cl} 2y_{i,n_n,n_s,l}, & x_\gamma \equiv y_{i,n_n,n_s,l} \\ 0, & \text{otherwise} \end{array} \right.$$

for $l = 1, \ldots, n_d$ and

$$\frac{\partial c_{\psi_{f_{n_d+q}}}}{\partial x_\gamma} = \begin{cases} 2\left(y_{i,n_n,n_s,l} - y_{i,n_n,n_s,\lambda}\right), & x_\gamma \equiv y_{i,n_n,n_s,l} \\ -2\left(y_{i,n_n,n_s,l} - y_{i,n_n,n_s,\lambda}\right), & x_\gamma \equiv y_{i,n_n,n_s,\lambda} \\ 0, & \text{otherwise} \end{cases}$$

for $l = 1, \ldots, n_d - 1$ and $\lambda = l+1, \ldots, n_d$, where

$$q = \left(\sum_{\mathfrak{L}=1}^{l-1} n_d - \mathfrak{L}\right) + (\lambda - l)\,. \tag{5.14}$$

5.3.5.2 Final Formation Plane

The final formation constraints ensure that the plane of the formation of spacecraft is perpendicular to the pointing direction at the final time. Recall that the specified pointing direction is $\boldsymbol{r}^*_{cs}(t)$, which, in this case is assumed to be expressed in the rotating $\mathcal{R}$ frame. The final formation constraints use the dot product between this pointing direction and the relative distances between each spacecraft. That is, there are

$$n_{\psi_f} = \binom{n_d}{2}$$

plane constraints of the form

$$\boldsymbol{c}_{\psi_f}(\boldsymbol{x}) = \begin{bmatrix} \sum_{i=1}^{3}\left(y_{i,n_n,n_s,1} - y_{i,n_n,n_s,2}\right) r^*_{(cs)_i}(t_f) \\ \vdots \\ \sum_{i=1}^{3}\left(y_{i,n_n,n_s,l} - y_{i,n_n,n_s,\lambda}\right) r^*_{(cs)_i}(t_f) \\ \vdots \\ \sum_{i=1}^{3}\left(y_{i,n_n,n_s,n_d-1} - y_{i,n_n,n_s,n_d}\right) r^*_{(cs)_i}(t_f) \end{bmatrix}.$$

Jacobian elements take the form

$$\frac{\partial c_{\psi_{f_q}}}{\partial x_\gamma} = \begin{cases} r^*_{(cs)_i}(t_f), & x_\gamma \equiv y_{i,n_n,n_s,l} \\ -r^*_{(cs)_i}(t_f), & x_\gamma \equiv y_{i,n_n,n_s,\lambda} \\ \sum_{i=1}^{3}\left(y_{i,n_n,n_s,l} - y_{i,n_n,n_s,\lambda}\right) \dot{r}^*_{(cs)_i}(t_f), & x_\gamma \equiv t_f \\ 0, & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, 3$, $l = 1, \ldots, n_d - 1$, and $\lambda = l+1, \ldots, n_d$, where q is again defined by Equation 5.14.

5.3.5.3 Partial Derivatives for the Dynamics Function

One may observe from Equation 5.1 that the partials for the absolute equations, $\tilde{\boldsymbol{f}}$, taken for the chief states, can be expressed as,

$$\begin{aligned}
\boldsymbol{A}(\boldsymbol{r}_c) &\equiv \frac{\partial \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{u}_c)}{\partial \boldsymbol{y}_c} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{U}_{xx} & 2\boldsymbol{\Omega} \end{bmatrix} \\
\boldsymbol{B} &\equiv \frac{\partial \tilde{\boldsymbol{f}}(\boldsymbol{y}_c, \boldsymbol{u}_c)}{\partial \boldsymbol{u}_c} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix},
\end{aligned}$$

where $\boldsymbol{U}_{xx} = \boldsymbol{U}_{xx}(\boldsymbol{r}_c)$ is the second partial of the potential function taken with respect to the positions, $\boldsymbol{r}_c$, and

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

is a constant matrix. Notice that because the velocity states only appear linearly in Equation 5.1, the state Jacobian is only dependent upon the chief position states. In addition, the control argument in $\tilde{\boldsymbol{f}}$ is written in the $\mathcal{R}$ frame, as the states are. This is an important distinction, as the formulation for this problem has the states, $\boldsymbol{y}_l$, in the $\mathcal{R}$ frame, while the controls, $\boldsymbol{u}_l \equiv \boldsymbol{u}_l^{\mathcal{B}_l}$ are necessarily written in the $\mathcal{B}_l$ frame so that they remain constant over each segment. For clarity, the equivalence of Equation 5.4 may be expressed as

$$\dot{\boldsymbol{y}}_l = \boldsymbol{f}(t, \boldsymbol{y}_l, \boldsymbol{u}_l^{\mathcal{B}_l}) = \bar{\boldsymbol{f}}(t, \boldsymbol{y}_l, \boldsymbol{u}_l^{\mathcal{R}}) = \bar{\boldsymbol{f}}(t, \boldsymbol{y}, {}^{\mathcal{R}}\boldsymbol{C}^{\mathcal{B}_l}\boldsymbol{u}^{\mathcal{B}_l}).$$

Note that the relationship,

$$\boldsymbol{u}_l^{\mathcal{R}} = {}^{\mathcal{R}}\boldsymbol{C}^{\mathcal{B}_l}\boldsymbol{u}_l^{\mathcal{B}_l}, \tag{5.15}$$

is implied, where the coordinate transformation matrix is defined by

$${}^{\mathcal{R}}\boldsymbol{C}^{\mathcal{B}_l} = \begin{bmatrix} \hat{\boldsymbol{x}}_{\mathcal{B}_l}^{\mathcal{R}} & \hat{\boldsymbol{y}}_{\mathcal{B}_l}^{\mathcal{R}} & \hat{\boldsymbol{z}}_{\mathcal{B}_l}^{\mathcal{R}} \end{bmatrix}.$$

Interestingly, the relative equations described by $\bar{\boldsymbol{f}}$ yield the same Jacobians as $\tilde{\boldsymbol{f}}$, with a slight modification in the position argument.

$$\begin{aligned} \frac{\partial \bar{\boldsymbol{f}}(t, \boldsymbol{y}_l, \boldsymbol{u}_l^{\mathcal{R}})}{\partial \boldsymbol{y}_l} &= \boldsymbol{A}(\boldsymbol{r}_c + \boldsymbol{r}_l) && (5.16) \\ \frac{\partial \bar{\boldsymbol{f}}(t, \boldsymbol{y}_l, \boldsymbol{u}_l^{\mathcal{R}})}{\partial \boldsymbol{u}_l^{\mathcal{R}}} &= \boldsymbol{B}. && (5.17) \end{aligned}$$

Note that the control Jacobian matrix remains constant. The state Jacobian matrix, however, is a function of the time and the relative position between chief and deputy. In addition,

$$\begin{aligned} \frac{\partial \bar{\boldsymbol{f}}(t, \boldsymbol{y}_l, \boldsymbol{u}_l^{\mathcal{R}})}{\partial t} &= \frac{\partial \dot{\boldsymbol{y}}_l}{\partial \boldsymbol{r}_c} \dot{\boldsymbol{r}}_c \\ &= (\boldsymbol{A}(\boldsymbol{r}_c + \boldsymbol{r}_l) - \boldsymbol{A}(\boldsymbol{r}_c)) \begin{bmatrix} \dot{\boldsymbol{r}}_c \\ \boldsymbol{0} \end{bmatrix}. && (5.18) \end{aligned}$$

Equations 5.16-5.18 can ultimately be used to calculate the relevant partials, $\frac{\partial \boldsymbol{f}}{\partial t}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}_l}$, as

$$\begin{aligned} \frac{\partial \boldsymbol{f}}{\partial t} &= \frac{\partial \bar{\boldsymbol{f}}}{\partial t} + \frac{\partial \bar{\boldsymbol{f}}}{\partial \boldsymbol{u}_l^{\mathcal{R}}} \frac{\partial \boldsymbol{u}_l^{\mathcal{R}}}{\partial t}, \\ \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}_l} &= \frac{\partial \bar{\boldsymbol{f}}}{\partial \boldsymbol{y}_l} + \frac{\partial \bar{\boldsymbol{f}}}{\partial \boldsymbol{u}_l^{\mathcal{R}}} \frac{\partial \boldsymbol{u}_l^{\mathcal{R}}}{\partial \boldsymbol{y}_l}. \end{aligned}$$

It remains to derive $\frac{\partial \boldsymbol{u}_l^{\mathcal{R}}}{\partial t}$ and $\frac{\partial \boldsymbol{u}_l^{\mathcal{R}}}{\partial \boldsymbol{y}_l}$. Using Equation 5.15, it is clear that the time derivative must consider how the transformation matrix changes with time.

$$\frac{\partial \boldsymbol{u}_l^{\mathcal{R}}}{\partial t} = \dot{\boldsymbol{u}}_l^{\mathcal{R}} = {}^{\mathcal{R}}\dot{\boldsymbol{C}}^{\mathcal{B}_l} \boldsymbol{u}_l^{\mathcal{B}}$$

The time derivative of the transformation matrix can be derived element by element. Recall that the matrix is composed of the body frame unit vectors written in the rotating frame. Their derivatives are based on the definitions for the body frame axes given in Equations

5.7-5.9:

$$\begin{aligned}
{}^{\mathcal{R}}\dot{\boldsymbol{C}}^{\mathcal{B}_l} &= \begin{bmatrix} \dot{\hat{\boldsymbol{x}}}^{\mathcal{R}}_{\mathcal{B}_l} & \dot{\hat{\boldsymbol{y}}}^{\mathcal{R}}_{\mathcal{B}_l} & \dot{\hat{\boldsymbol{z}}}^{\mathcal{R}}_{\mathcal{B}_l} \end{bmatrix}, \\
\dot{\hat{\boldsymbol{x}}}^{\mathcal{R}}_{\mathcal{B}_l} &= \frac{1}{\tilde{x}}\dot{\tilde{\boldsymbol{x}}} - \frac{\dot{\tilde{x}}}{\tilde{x}^2}\tilde{\boldsymbol{x}}, \\
\dot{\hat{\boldsymbol{y}}}^{\mathcal{R}}_{\mathcal{B}_l} &= \frac{1}{\tilde{y}}\dot{\tilde{\boldsymbol{y}}} - \frac{\dot{\tilde{y}}}{\tilde{y}^2}\tilde{\boldsymbol{y}}, \\
\dot{\hat{\boldsymbol{z}}}^{\mathcal{R}}_{\mathcal{B}_l} &= \dot{\hat{\boldsymbol{x}}}^{\mathcal{R}}_{\mathcal{B}} \times \hat{\boldsymbol{y}}^{\mathcal{R}}_{\mathcal{B}} + \hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}} \times \dot{\hat{\boldsymbol{y}}}^{\mathcal{R}}_{\mathcal{B}},
\end{aligned}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{r}_{cs}$, $\tilde{x} = |\tilde{\boldsymbol{x}}|$, $\tilde{\boldsymbol{y}} = \boldsymbol{r}_l \times \hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l}$ and $\tilde{y} = |\tilde{\boldsymbol{y}}|$.

Likewise, the changes in the control with respect to the states are a factor of how the body frame (and, therefore, the coordinate transformation matrix) change based on the states. That is,

$$\begin{aligned}
\frac{\partial \boldsymbol{u}^{\mathcal{R}}_l}{\partial \boldsymbol{y}_l} &= (\boldsymbol{u}^{\mathcal{B}_l}_l)_x \frac{\partial \hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l}}{\partial \boldsymbol{y}_l} + (\boldsymbol{u}^{\mathcal{B}_l}_l)_y \frac{\partial \hat{\boldsymbol{y}}^{\mathcal{R}}_{\mathcal{B}}}{\partial \boldsymbol{y}_l} + (\boldsymbol{u}^{\mathcal{B}}_l)_z \frac{\partial \hat{\boldsymbol{z}}^{\mathcal{R}}_{\mathcal{B}}}{\partial \boldsymbol{y}_l}, \\
\frac{\partial \hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l}}{\partial \boldsymbol{y}_l} &= \boldsymbol{0}_{3\times 6}, \\
\frac{\partial \hat{\boldsymbol{y}}^{\mathcal{R}}_{\mathcal{B}_l}}{\partial \boldsymbol{y}_l} &= \begin{bmatrix} -\left[\hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l} \times\right] & \boldsymbol{0}_{3\times 3} \end{bmatrix}, \\
\frac{\partial \hat{\boldsymbol{z}}^{\mathcal{R}}_{\mathcal{B}_l}}{\partial \boldsymbol{y}_l} &= \left[\hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l} \times\right] \frac{\partial \hat{\boldsymbol{y}}^{\mathcal{R}}_{\mathcal{B}_l}}{\partial \boldsymbol{y}_l},
\end{aligned}$$

where $\left[\hat{\boldsymbol{x}}^{\mathcal{R}}_{\mathcal{B}_l} \times\right]$ is the 3×3 skew symmetric matrix that performs the same operation on a 3×1 vector as the cross product.

With this, the implementation of the spacecraft formation problem is completely defined. Next, results obtained through this formulation are demonstrated and explored.

5.4 Sample Solutions

A candidate formation consists of $n_d = 3$ deputy spacecraft, equally spaced as in Figure 5.4 with distance $r^*_{cd} = 1$ km between the chief and each deputy and $r^*_{dd} = 1.73$

km between two deputies. The formation seeks to collect data on a distant star located along the inertial pointing direction, $\boldsymbol{r}_{cs}^{*\mathcal{I}} = [1\ 0\ 0]^T$. Solutions are found with several sets of parameters. A baseline solution is found with $n_n = 4$ nodes per segment and $n_k = 10$ interior switch times per control axis, allotting for six thrusting segments per control axis per satellite (three positive, three negative). A trajectory is devised with fixed initial states, and final conditions dictate the formation size and plane constraints be met at the final time. Since the CR3BP is time invariant, the initial time is set as $t_0 = 0$. The formation nominally rotates about the reference with the same period as the Halo orbit. By fixing the final time, t_f, at a third of the reference Halo orbit period (approximately 5.1183×10^6 seconds), the three deputies each complete a third of a rotation about the chief.

5.4.1 The Initial Guess

In numerical optimization problems, the initial guess, $\boldsymbol{x}_0$, is often a determining factor in whether an optimal solution is successfully identified. A gradient-based numerical algorithm can only be trusted to find *locally* optimal solutions, most likely in the vicinity of the starting point. Thus, identical problems can easily converge on different solutions if starting from distant points. One way to manage this characteristic is to use several different values of $\boldsymbol{x}_0$ to find solutions, choosing the minimum of the locally optimal solutions as a candidate for the globally optimal solution. For this effort, the successful identification of any feasible and locally optimal solution to the highly constrained formation problem demonstrates the capability of the FSCT method.

Collocation methods, where discretized states are included as parameters in the optimization, allow the user to select an initial set of parameters that *guides* the optimization process towards a particular trajectory. Although $\boldsymbol{x}_0$ does not satisfy the dynamical constraints, its states may represent the vicinity of the desired solution. Through iteration, the optimizer modifies the current point until feasibility and optimality tolerances are achieved.

In this mindset, a baseline initial guess for this candidate formation appears in Figure 5.5, depicting the trajectories of three spacecraft in the inertial and rotating frames, along with a nominal control profile of the three spacecraft in their respective body frames. As seen in the inertial frame, the trajectories satisfy the formation requirements (size and plane orientation) as presented above, maintaining a plane perpendicular to the inertial pointing direction, $\boldsymbol{r}_{cs}^{*\mathcal{I}}$. This set of states represent a perfectly circular planar formation. Ideally, an unconstrained control profile is capable of delivering the associated accelerations. The constrained controls shown here, however, do not produce these states. Instead, the burn durations chosen for the initial guess simply distribute the time of each of the $n_u n_k + 1 = 31$ segments equally. This choice, while somewhat arbitrary, frees the optimizer to search the solution space for the proper durations of burning and coasting segments, while keeping the states fairly close to their ideal placement.

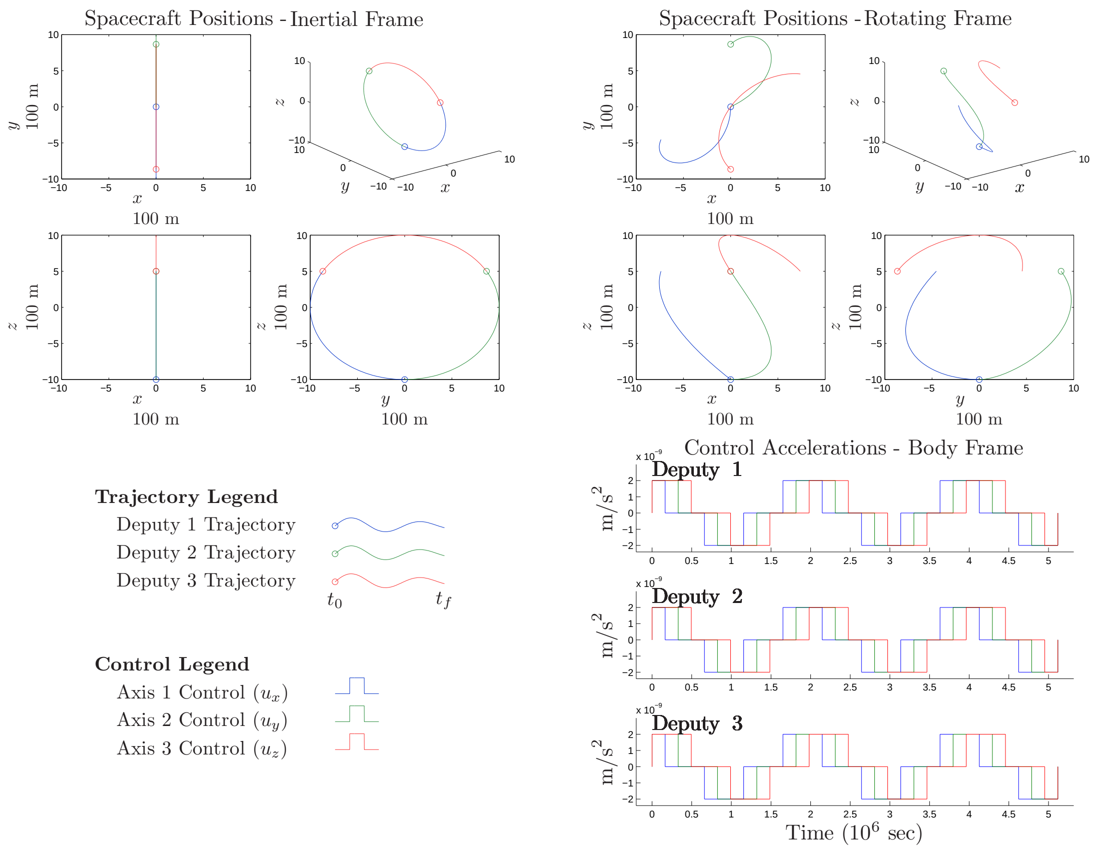


Figure 5.5: Baseline Initial Guess

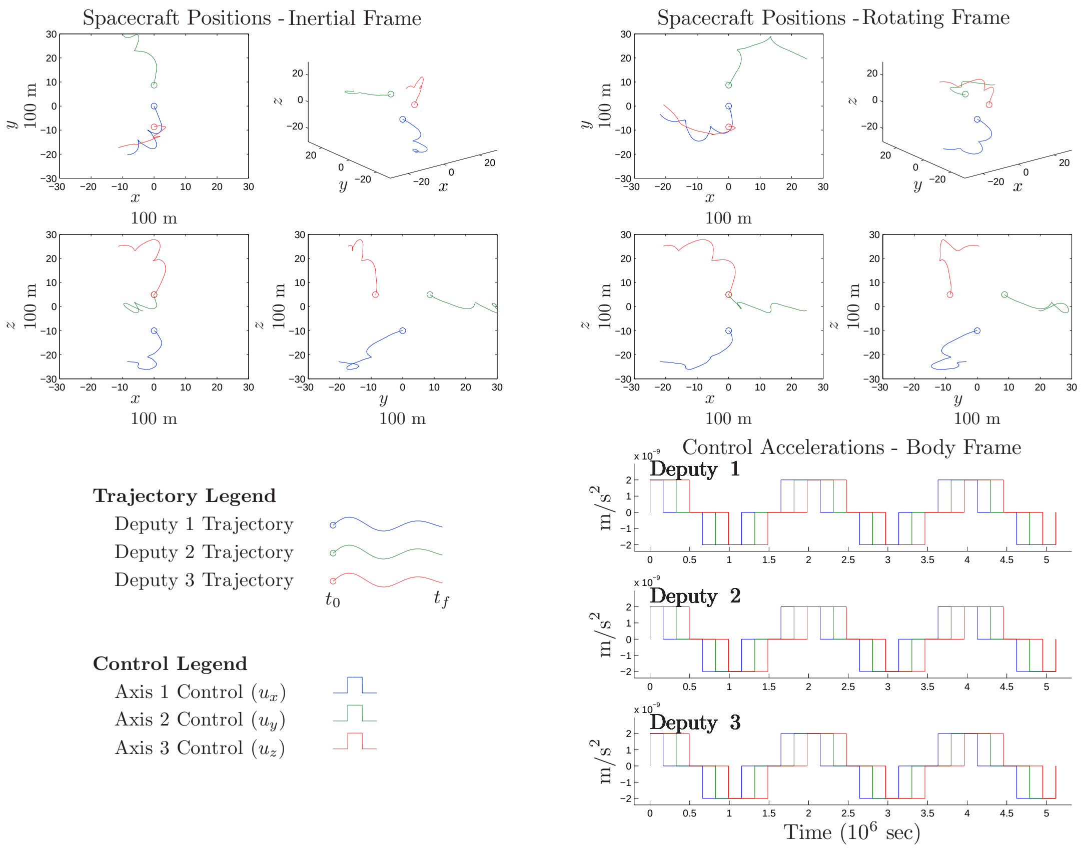


Figure 5.6: Feasible Initial Guess

As an alternative, an initial guess can be generated that is feasible by propagating a guessed control profile, say, that of the baseline guess. In this case, the initial guess satisfies the dynamical constraints immediately, but the formation constraints are clearly not satisfied. In addition, deviations from the desired formation characteristics are large. Thus, the cost is also large at the first iteration of the optimization procedure. In Figure 5.6, a set of states is shown that is the true propagation of the arbitrary control profile used in the baseline guess. It is apparent that the trajectories of three spacecraft driven by this control profile are far from meeting any formation requirements.

Although this is a dramatic example, the difference between the baseline and feasible initial guess communicates an advantage to using a collocation method over an indirect or direct shooting method. Shooting methods essentially 'shoot' a guess for the control to the terminal time, making corrections on each iteration to meet constraints. However, because of the extreme sensitivities of this dynamic regime, the quality of a shooting guess is critical to the success of the process. A collocation method is far more flexible in the initial guess.

5.4.2 Baseline Solution

The baseline initial guess is optimized via the fomulation presented in this work to yield the solution depicted in Figure 5.7. The trajectories of the three spacecraft are shown in the inertial and rotating reference frames, along with the control profile for each of the spacecraft.

First, it is observed that the optimization has modified the guessed segment durations to specify proper burn times and burn durations. Burn segments are shortened and coast segments are lengthened according to feasibility constraints (driving dynamic constraint residuals to zero) and optimality considerations (minimizing fuel expenditure). Notice also that several burn segments have zero duration, depicted as a single line separating two coasting arcs. Recall that the formulation pre-specifies thrusting arcs. However, if a solution does not require that thrusting arc, its duration is reduced to zero in the optimization. The structure of the solution does not change, and a segment still exists in that place (occurring over zero time). Additionally, n_n nodes exist at a single point in time. Since the solution implies the given burn segment is superfluous, many of the optimization variables have no effect (these are the states along the zero segment). However, since no feasible control profile was known a priori, it is helpful to include *more* segments than ultimately necessary until some insight into a feasible trajectory is extracted.

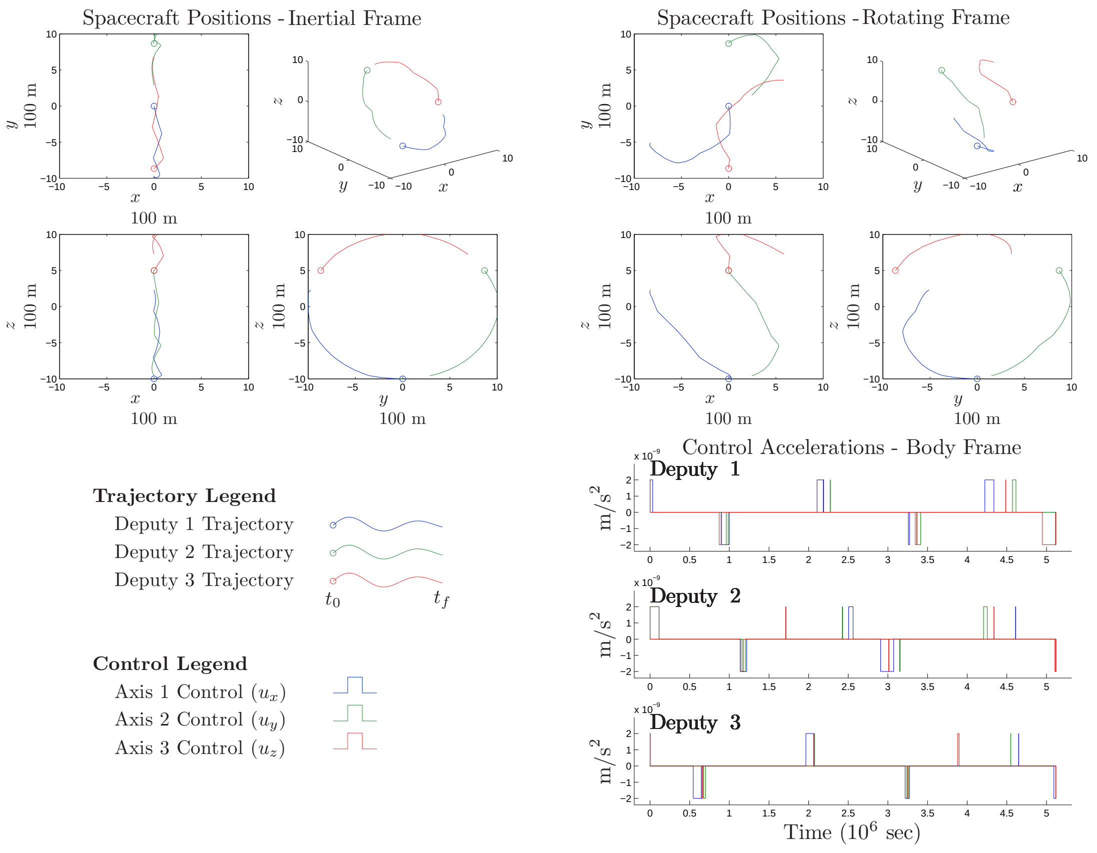


Figure 5.7: Baseline Solution

Next, it is apparent that the paths of each of the deputy spacecraft resemble the initial guess to a significant extent. Certainly, deviations away from the 'perfect' trajectory are visible, but the solution implies that those deviations are necessary in order to identify a feasible trajectory given the specified control restrictions. More importantly, the trajectory now coincides with the control history. For example, the control discontinuities between segments result in corners in the velocity states for each spacecraft (not shown).

One can also extract from the baseline solution an optimal rotation rate for the formation. By comparing the inertial trajectories of the baseline guess and solution, it is apparent in the optimal solution that the spacecraft traverse less distance over the time interval. This suggests that for a 1 km (radius) formation, the optimal rotation rate is slightly less than the baseline guess of 2π/period.

The baseline solution demonstates the effectiveness of the solution technique on a highly constrained optimal control problem. The sample mission requires fixed spacecraft orientation, fixed thruster locations, and a specified thrust acceleration magnitude. Under these constraints, trajectories are sought to maintain formation size, shape, and orientation restrictions. An unconstrained control solution that maintains the formation configuration requires acceleration levels below those deliverable by existing technology. In the actuator constrained case, the proposed methodology determines reasonable performance expectations in the presense of these system constraints.

5.4.3 Varying Parameters to Obtain Different Solutions

The methodology presented in this formulation serves to establish reasonable expectations for libration point missions under the constraints imposed by the current state of technology. The generalized method can be applied to any number of potential scenarios during the mission planning process. To demonstrate the flexibility of the method, some formulation parameters are changed here to observe their effect on the solution. The final time, the number of nodes and knots, the initial guess, and the thrust magnitude are varied to determine four other solutions to be compared with the baseline in Table 5.1. The results vary in size, solution convergence, and cost performance.

By doubling the fixed final time (now two thirds of an orbital period, or 10.2366×10^6 seconds) without any other changes, it is evident that the cost is significantly increased from the baseline. Of course, it would be reasonable to expect the integral costs to be proportional to the final time. However, the solution is over an order of magnitude greater in cost. That

Table 5.1: Comparison of Solutions with Various Parameters

	Baseline	t_f	n_k, t_f	Feasible Guess	Thrust
n_n	4	4	6	4	4
n_k	10	10	20	10	10
t_f (10^6 sec)	5.1183	10.2366	10.2366	5.1183	5.1183
Guess	Baseline	Baseline	Baseline	Feasible	Baseline
w_1 (Thrust)	$\frac{1}{400}$	$\frac{1}{400}$	$\frac{1}{400}$	$\frac{1}{400}$	$\frac{1}{1600}$
w_2 (Distance)	0.1	0.1	0.1	0.1	0.1
w_3 (Plane)	1	1	1	1	1
T^* (km/s^2)	2.0e-12	2.0e-12	2.0e-12	2.0e-12	4.0e-12
n	2333	2333	6779	2333	2333
# Iterations	45	157	194	95	272
Computational Time (sec)	253.91	956.41	3000.35	575.88	1570.04
Weighted Thrust Cost	1.6233	5.8133	6.6838	10.9247	1.1324
Weighted Formation Cost	16.1818	634.3255	67.7092	35.5675	6.9286
Weighted Plane Cost	0.8591	84.6949	4.6852	4.9035	1.0632
Total Cost	18.6643	724.8338	79.0782	51.3956	9.1242

is because for a longer trajectory, a good solution necessarily requires more maneuvers, and, subseqently, more knots (switching opportunities) in the problem formulation.

The next example maintains the longer final time, now including twice as many knots as the baseline. In addition the number of nodes is increased to improve the fidelity of the solution. As a result, the dimension of the parameter optimization problem increases significantly (from 2333 to 6779 variables). This requires additional computational time for each iteration and an overall longer convergence time. However, including more knots does lead to an improved solution relative to the previous example.

The feasible initial guess is also compared in performance to the baseline initial guess (using the original final time and numbers of nodes and knots). In contrast to the baseline, the feasible guess does not lead to an improved solution. Clearly, the baseline guess is in the vicinity of a smaller locally optimal point than the feasible guess. This is expected since the feasible initial guess does not produce trajectories that meet the formation requirements.

Finally, a solution is identified using a larger thrust magnitude than the baseline. Although the total cost is less than the baseline, this takes into account a different weight on the thrust portion of the cost index. The weight is selected to normalize the cost, dividing away the square of the thrust magnitude (in scaled distance and time units). Each of the thrust acceleration magnitudes and cost function weights in this investigation are selected arbitrarily. However, it is clear that these parameters can be tailored to values that effectively represent a realistic mission scenario.

5.5 Spacecraft Formation Pointing Survey

In this section, the sample solutions of Section 5.4 are extended. Specifically, many similar solutions are obtained, varying the pointing direction of the $n_d = 3$ spacecraft formation to gather intuition regarding the relative difficulty of maintaining formation requirements in different orientations. The results that follow are derived from up to 290 optimizations on each of two sample weight distributions for the cost function. The optimizations vary the inertial pointing defined by $\boldsymbol{r}_{cs}^{*\mathfrak{I}}$ to determine the minimum cost for maintaining a formation in each direction.

The objective weights are selected to emphasize formation size and shape requirements in the first survey and formation orientation or plane in the second. This is simply accomplished by altering weights w_2 and w_3, respectively. Of course, a limited fuel resource is always a concern for space missions, thus a penalty on excessive fuel expenditure is also enforced. The weight distributions for these surveys are included in Table 5.2 for 'formation emphasis' and 'plane emphasis' surveys. Observe that, compared to each other, the first survey emphasizes $\mathfrak{J}_2 = F_2(\boldsymbol{x})$ in the total cost function, while the second emphasizes $\mathfrak{J}_3 = F_3(\boldsymbol{x})$.

5.5.1 Data Samples

An interferometry mission may require a spacecraft formation to perform imaging on distant stars or planets located literally anywhere in the universe. Thus, the formation must

Table 5.2: Weight Distribution for Two Pointing Surveys

	Formation	Plane
w_1	$\frac{1}{(T^*)^2}$	$\frac{1}{(T^*)^2}$
w_2	0.1	0.01
w_3	1.0	10.0
	$T^* = 20$ DU/TU^2	

be able to maintain formation requirements while pointing in any direction. By sampling a sufficient number of different pointing directions, some trends may be established regarding the various orientations. It is clear that some orientations result in trajectories for the member spacecraft that are easier to maintain in the underlying dynamic scheme than others.

In this section, the varying parameter is a 3-dimensional vector representing the pointing direction perpendicular to the formation plane. The pertinent output for comparison for each data sample is the resulting minimized cost, $\mathcal{J} = F(\boldsymbol{x})$. Illustrating results in four dimensions (three independent variables and one output) is somewhat difficult. To accomplish this, a thermal plot is selected. The pointing direction is represented as a point on a unit sphere, and the resulting cost of the optimization conducted in that direction is depicted as a color at that point. Thus, survey results are analogous to a global temperature indicator showing the current temperature of every place around the world. The 'temperature' is presented as a color on the globe, highlighting hot spots and cold spots wherever they may exist.

Since FSCT optimizations must be conducted for each pointing direction, only a sampling of data points may be obtained. Pointing directions are selected by first defining

$$\boldsymbol{r}_{cs}^{*\mathcal{I}} \equiv \frac{\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T}{r_1^2 + r_2^2 + r_3^2}.$$

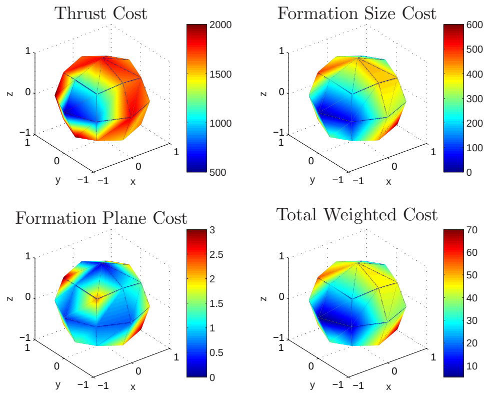
If $r_i \in \{-1, 0, 1\}$, then the total number of unique pointing directions is 26. Likewise, if $r_i \in \{-2, -1, 0, 1, 2\}$, there are 98 data points, and $r_i \in \{-3, -2, -1, 0, 1, 2, 3\}$ yields 290 points. Thermal plots are demonstrated below with each set of data points.

Note that for each optimization solution obtained in the survey results that follow, a new initial guess is generated to initiate the FSCT method. The baseline initial guess in Section 5.4.1 is modeled for the guesses here. An automated procedure produces starting points for each optimization representing an ideal trajectory for each of the 290 sample pointing directions.

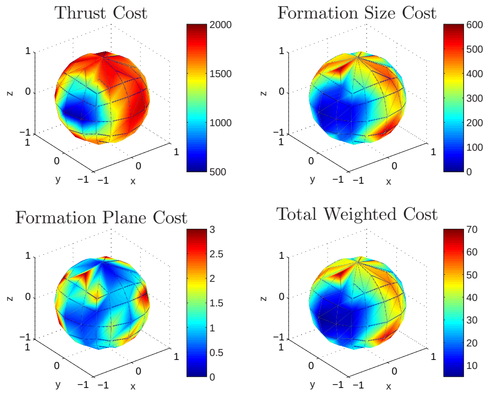
5.5.2 Pointing Survey: Formation Emphasis

Using the weight distribution associated with the 'formation emphasis' in Table 5.2, a survey of costs is revealed using 26, 98, and 290 data points distributed fairly evenly over a unit sphere representing all possible pointing directions for the spacecraft formation. The results are summarized in Figure 5.8.

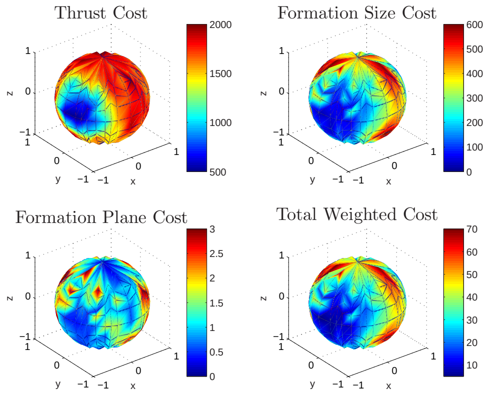
Note that for each data set, four thermal plots are displayed, indicating the costs associated with thrust ($\mathcal{J}_1$), formation size/shape ($\mathcal{J}_2$), formation orientation ($\mathcal{J}_3$), and the total weighted cost when all are combined ($\mathcal{J}$). Each subplot is scaled to accentuate the variation per cost, with red indicating higher costs and blue lower costs. Also, the colors are interpolated between data points to smooth out the image.



(a) 26 Points

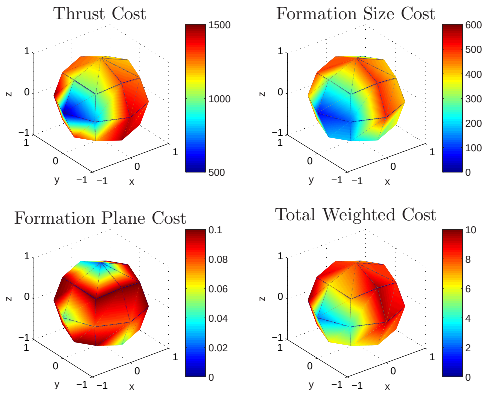


(b) 98 Points

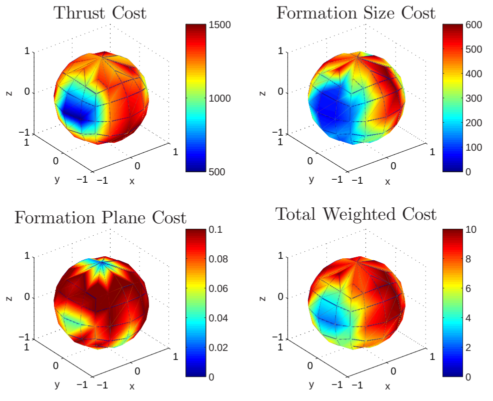


(c) 290 Points

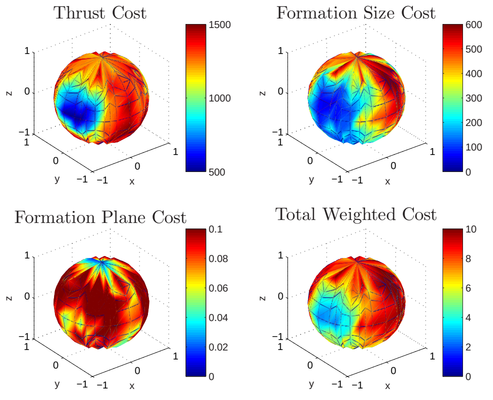
Figure 5.8: Formation Emphasis: Comparison of 26, 98, 290 Points



(a) 26 Points



(b) 98 Points



(c) 290 Points

Figure 5.9: Plane Emphasis: Comparison of 26, 98, 290 Points

A broad comparison between Figures 5.8(a), (b), and (c) indicates the value of using more data points when determining cost trends. This is most easily seen by comparing the thrust cost for each set of data points. Although a trend is apparent with only 26 points in (a) that $\boldsymbol{r}_{cs}^{*\mathcal{I}} = [\pm 1\ 0\ 0]^T$, or a formation restricted to the $\hat{y} - \hat{z}$ plane, results in a lower thrust cost than any others, this trend is undeniable, and well articulated with 290 data points in (c).

On the other hand, what appears as a trend in the formation size/shape cost in (a) is not as obvious in (c). One potential cause of this is the presence of outliers in either data set. An outlier, in the case of FSCT optimization, is a solution that converged on a larger local minimum. Thus, in the figures, a red spot surrounded by a field of blue suggests that, for numerical reasons, the FSCT iterative procedure terminated earlier than it may have if either a different initial guess were used, or tighter optimality tolerances were enforced.

As the title of this section implies, the total weighted cost subplot is nearly identical to the formation size/shape cost subplot. With a higher value for w_2, the cost associated with size and plane deviations in the formation dominates the total cost.

5.5.3 Pointing Survey: Plane Emphasis

Next, the 'plane emphasis' weight distribution of Table 5.2 is employed for a total of 290 separate optimization jobs of the FSCT method. Again, thrust cost, formation size/shape deviations, formation orientation deviations, and total weighted cost are illustrated for 26, 98, and 290 data points in Figure 5.9. The trajectories that yield these results assume, in general, that it is more important that the spacecraft formation remain in the plane perpendicular to the pointing direction than it is for the spacecraft to maintain constant spacing relative to each other.

Note, first, that the scaling in Figure 5.9 is different than Figure 5.8, and consequently, each color is assigned a new range for each subplot. The range of values corresponding to the color spectrum is selected to accentuate the deviations the best over the data set.

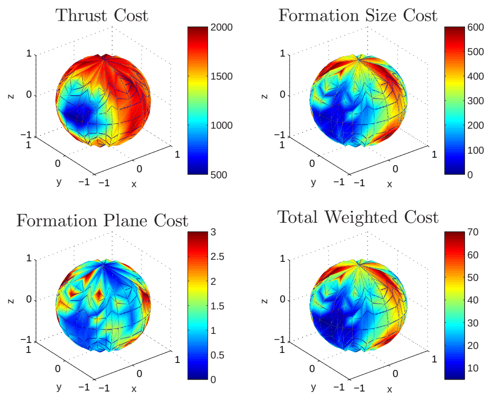
Observing the formation plane cost for each data set, a subtle trend emerges that implies that a formation confined to the $\hat{x} - \hat{y}$ plane (perpendicular to $\boldsymbol{r}_{cs}^{*\mathcal{I}} = [0\ 0\ \pm 1]^T$) is easier to maintain. However, even with changes in w_2 and w_3 totaling two orders of magnitude, deviations are more noticeable in the distance between spacecraft.

In each survey, it is apparent that the thermal plots possess, at least to some extent, some symmetry. This makes sense, as two opposing pointing directions for $\boldsymbol{r}_{cs}^{*\mathcal{I}}$ still result in a spacecraft formation confined to the same perpendicular plane. It should be noted, though, based on the momentum provided by the initial guesses (recalling that a final solution remains in the same vicinity), the spacecraft are always moving around $\boldsymbol{r}_{cs}^{*\mathcal{I}}$ with a negative (clockwise) rotation. Thus the solutions for two opposing pointing directions are different, but it is validating to see similar costs on opposite sides of the globe (observable throughout).

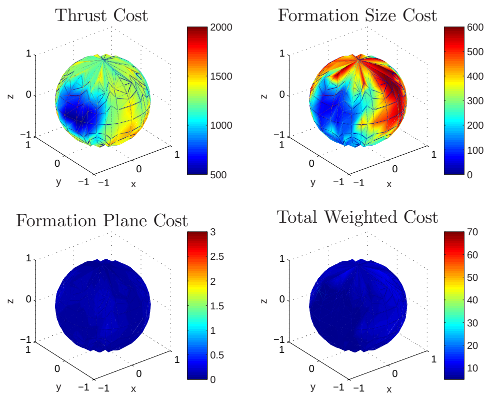
5.5.4 Formation vs. Plane Emphasis Comparison

In Figures 5.8 and 5.9, the cost ranges for the color spectrum are altered, making it difficult to compare the results of both surveys (formation emphasis and plane emphasis). Thus, it is useful to repeat the results of each survey, now with consistent scaling. In Figures 5.10 and 5.11, the 290 point data sets for each of the two weight distributions are again illustrated. In the first figure, the color spectrum is assigned to show an informative range of cost values for the first survey, and the second figure corresponds to the cost range of the second survey. Some interesting observations can be made by studying each of these figures.

Although it was not as obvious in Section 5.5.3, Figure 5.10(b) clearly demonstrates how the total cost of this survey is dominated by the formation plane deviation cost, as



(a) Formation Emphasis



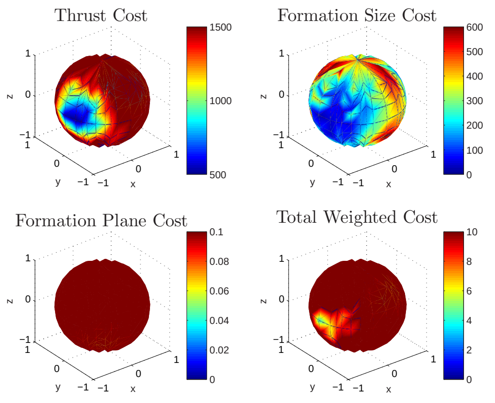
(b) Plane Emphasis

Figure 5.10: Formation vs. Plane Emphasis: Scaling for Formation Emphasis

these two subplots more obviously show a similar color manifold.

Notice, also, that the individual costs of formation size deviations and formation plane deviations are smaller when they are emphasized in the total cost function. This is intuitive: as the optimizer seeks to improve the total cost through iteration, it will naturally affect the higher-weighted elements of the cost. This is most obvious with the plane cost. In Figure 5.10, when the plane cost is emphasized (b), it is smaller across all data points, as compared to when it is not emphasized in the total cost function (a).

It is already observed that it is more difficult to maintain formation size than formation plane. Another indication of this fact can be seen by comparing the thrust cost between the two surveys. It is clear that more actuation is required when the formation size and shape are emphasized in the cost function, since the thrust cost subplots show more red in (a) than (b) for each of the figures comparing the two surveys. Thus, when it becomes more important to ensure consistent distances between spacecraft, more (or longer) thruster firings are required, increasing fuel expenditure.


(a) Formation Emphasis

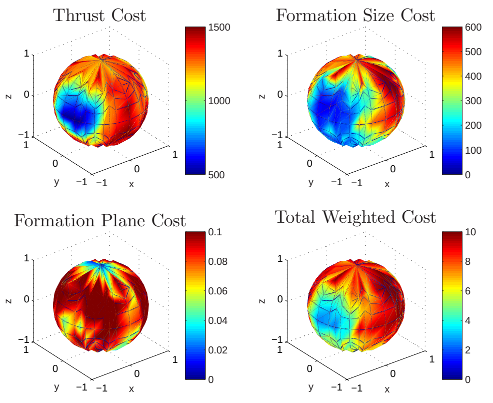

(b) Plane Emphasis

Figure 5.11: Formation vs. Plane Emphasis: Scaling for Plane Emphasis

5.6 Summary

In this chapter, the FSCT method is employed to determine locally optimal control solutions for libration point formations restricted to highly constrained actuation requirements. Some minor modifications and enhancements are implemented to address the specific problem, mostly rooted in the necessity for handling multiple dynamic bodies.

The solutions presented demonstrate the method's capability to minimize selected costs when determining a feasible trajectory. A number of mission parameters are used to demonstrate a variety of solutions. Some formation characteristics considered include

- formation configuration, size, orientation, and rotation rate,
- thruster capability and placement,
- dynamic model and reference trajectory, and
- initial and terminal conditions.

Surveys are conducted using many obtained solutions to gather intuition regarding the challenges for spacecraft formations conducting deep-space imaging from restricted planes of motion. Weight distributions for the elements of the objective function are varied to help the mission planner gain intuition when prioritizing requirements that cannot be perfectly met via constraints.

The purpose of this exploration is to demonstrate how this methodology can serve as an effective mission planning tool. The FSCT solution process can be employed for analysis directed towards establishing requirements and capabilities for highly constrained formations.

Chapter 6

Extended Applications

Recall that the FSCT method is designed to treat problems defined by the general problem statement of Section 1.2. That is, the FSCT method optimizes performance for systems with state variables, $\boldsymbol{y}$, subject to the ordinary differential relation,

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \tag{1.1}$$

where

$$\begin{array}{rcl} \boldsymbol{y} & = & \left[y_1 \ \cdots \ y_{n_y}\right]^T, \\ y_i & \in & \mathbb{C}^1, \end{array} \tag{1.2}$$

and control variables, $\boldsymbol{u}$, limited to discrete sets as

$$\begin{array}{rcl} \boldsymbol{u} & = & \left[u_1 \ \cdots \ u_{n_u}\right]^T, \\ u_i & \in & \mathbb{U}_i = \left\{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i}\right\}. \end{array} \tag{1.3}$$

Up to this point, all of the applications have fallen directly within the confines of this system description. Indeed, there are a number of problems for which the FSCT method is immediately useful. Further, in Section 4.4, it is demonstrated that some systems ordinarily described with continuously-varying control elements may be reformulated with additional states to have discrete control variables only, thereby conforming to the system description above. Thus, the general problem statement is not overly restrictive.

However, it is interesting and useful to explore ways in which the FSCT method may be extended to broaden its scope of applications. With slight alterations to the methodology, it is possible to consider new classes of the hybrid control problem. The bounds set by Equations 1.1, 1.2, and 1.3 are now addressed systematically. Specifically, this chapter considers how the FSCT method can be applied with *continuous control* variables, *discrete state* variables, and *partial differential* state constraints. With continuous control variables, constraints in the underlying parameter optimization problem are relaxed; with discrete states, more constraints are added. For partial differential constraints, a discretization in the non-time differentiation variable allows a partial differential equation to be approximated by multiple ordinary differential equations.

Notational Considerations

The objective in each of the following sections is to transform a new class of hybrid problems into a formulation that is consistent with the FSCT methodology. For clarity, the variables $\boldsymbol{y}$ and $\boldsymbol{u}$ remain restricted by Equations 1.1, 1.2, and 1.3. The definitions of the state and control vectors alter, however, according to the type of transformation under consideration. The original system, formulated outside of the FSCT method bounds, is distinguished by the notation $(\cdot)'$.

6.1 Systems with Continuous Control Variables

Consider the system with both discrete and continuous control variables of the following form:

$$\begin{aligned}
\dot{\boldsymbol{y}}' &= \boldsymbol{f}'(t, \boldsymbol{y}', \boldsymbol{u}', \boldsymbol{v}'), \qquad (6.1)\\
&\left\{\begin{array}{lcl} \boldsymbol{y}' &=& \begin{bmatrix} y'_1 & \cdots & y'_{n_y}\end{bmatrix}^T, \\ y'_i &\in& \mathbb{C}^1, \end{array}\right.\\
&\left\{\begin{array}{lcl} \boldsymbol{u}' &=& \begin{bmatrix} u'_1 & \cdots & u'_{n_{u'}}\end{bmatrix}^T, \\ u'_i &\in& \mathbb{U}_i = \{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i}\}, \end{array}\right.\\
&\left\{\begin{array}{lcl} \boldsymbol{v}' &=& \begin{bmatrix} v'_1 & \cdots & v'_{n_{v'}}\end{bmatrix}^T, \\ v'_i &\in& \mathbb{C}^0, \end{array}\right.
\end{aligned}$$

where $n_{u'} + n_{v'}$ is the total number of control variables, including both discrete and continuous. In this form, this system cannot be immediately implemented with the FSCT formulation. The $n_{v'}$ continuous control variables, indicated by v'_i, must be free to vary with time over a continuous spectrum, but the FSCT method does not include the framework to optimize these variables. As presently formulated, the FSCT approach considers all control variables to be piecewise constant and optimizes the time elapsed between switches by incorporating these intervals as control parameters in $\boldsymbol{x}$.

When both continuous and discrete control variables are simultaneously considered, the continuous control variables are more appropriately treated as pseudo-states in the FSCT formulation. The states, of course, are continuously-varying elements whose values are optimized at each node. Certainly, v'_i may be described similarly. The primary difference between a traditional state, y'_i, and a continuous control, v'_i, then, is that the former is subject to dynamical constraints in Equation 6.1. Thus, continuous control variables can simply be treated as states without dynamical constraints. In other words, v'_i is a continuous independent variable that can be represented as a continuous dependent variable with relaxed dependencies.

Define the FSCT state vector as

$$\boldsymbol{y} \equiv \left[(\boldsymbol{y}')^T \ (\boldsymbol{v}')^T \right]^T, \tag{6.2}$$

and the FSCT control vector as

$$\boldsymbol{u} \equiv \boldsymbol{u}'. \tag{6.3}$$

Note that the continuous control variables are included in the state vector as pseudo-states. Now, $\boldsymbol{y}$ is continuous and $\boldsymbol{u}$ is discrete, consistent with all other formulations in this investigation. The state continuity constraints are

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) \equiv \begin{bmatrix} \boldsymbol{f}'(t, \boldsymbol{y}', \boldsymbol{u}') \\ \text{—} \\ \vdots \\ \text{—} \end{bmatrix}, \tag{6.4}$$

where $n_{v'}$ elements in the dynamics function are undefined, as there are no dynamical constraints on the continuous control variables. The definitions in Equations 6.2-6.4 represent the complete tranformation.

6.1.1 Relaxing Constraints for Continuous Control Variables

The new hybrid problem is nearly identical to the general problem statement of Section 1.2. Now, though, there are no formal dynamical relations for continuous independent (control) variables. One approach to handling this in the context of the FSCT formulation is to modify the state constraint equations so that only the traditional states, and not the augmented pseudo-states, are bound by continuity relations. Another approach is to formulate constraint equations for each element in $\boldsymbol{y}$, and simply *relax* the constraints associated with the pseudo-states.

Recall that traditional states may be bound by any number of different constraints. The common types of constraints include initial/final conditions, knot continuity constraints, and dynamical constraints imposed in this work using the Hermite-Simpson integration equations. The nature of the problem may dictate how these constraints are treated for the $n_{v'}$ pseudo-states. For example, it is possible that one may still want to specify an initial value for a control variable, in which case the pseudo-state may be treated identically to a traditional state. In most cases, though, all of the constraints associated with the pseudo-states must be relaxed.

Each constraint identified in Chapter 3 is presented as an equality constraint. That is, any constraint is $c_i(\boldsymbol{x}) = 0$. In the implementation with the optimizer, this may actually be formulated as

$$0 \leq c_i(\boldsymbol{x}) \leq 0.$$

Thus, to relax a constraint, the lower and upper bounds are changed to $\pm\infty$, as

$$-\infty \leq c_i(\boldsymbol{x}) \leq \infty.$$

The constraint is effectively eliminated from the parameter optimization problem. With this change to all $c_i(\boldsymbol{x})$ constraining pseudo-states, the continuous control variables are effectively unconstrained.

Consider the dynamic constraints derived from Equation 6.4. Since the constraint is relaxed with infinity bounds, it is possible to define the dynamics function as simply

$$\boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{f}'(t, \boldsymbol{y}', \boldsymbol{u}') \\ \boldsymbol{0} \end{bmatrix},$$

where the value of zero is placed in every undefined element of the dynamics function. Although these elements can be set to anything without affecting the resulting solution, it is wise to define them with a dummy value to avoid confusion within the underlying parameter optimization problem.

6.1.2 Example: Zermelo Navigation

An example application with both continuous and discrete controls is presented in Section 3.1.1. In that variation of the Zermelo navigation problem, two control variables exist. The thrust magnitude is limited to a finite set of values, while the thrust direction is free to vary over a continuous spectrum. In the presentation of Section 3.1.1, a simplified formulation is implied for the sake of developing the methodology in steps. With the complete FSCT formulation, the Zermelo problem can be transformed as described above to yield an equivalent solution.

Formulate the FSCT problem with states

$$\boldsymbol{y} \equiv [y_1 \;\; y_2 \;\; y_3 \;\; y_4 \;\; \theta]^T ,$$

where

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, u) = \begin{bmatrix} y_3 \\ y_4 \\ u \cos\theta \\ u \sin\theta - y_3 \cos y_1 \\ 0 \end{bmatrix} , \tag{6.5}$$

and the single discrete control is $u \in \{0, 2\}$. Note that in Equation 6.5, the differential continuity condition $\dot{y}_5 = \dot{\theta} = 0$ is not enforced, and zero is only used as a dummy value. When all constraints on y_5 are relaxed, the FSCT problem may be solved.

The solution with 3 segments (2 knots) is shown in Figure 3.2. When the problem is solved using the present formulation, the solution is equivalent to that 3-segment solution. Choosing more knots in the present formulation ($n_k = 14$, for example), results in many zero-duration segments, with the equivalent of three non-zero segments representing a thrust-coast-thrust sequence. For brevity, figures are not repeated here.

6.2 Systems with Discrete State Variables

Now consider the system with both discrete and continuous state variables of the following form:

$$\begin{aligned}
\dot{\boldsymbol{y}}' &= \boldsymbol{f}'(t, \boldsymbol{y}', \boldsymbol{\eta}', \boldsymbol{u}'), \\
\boldsymbol{\eta}' &= \boldsymbol{g}'(t, \boldsymbol{y}', \boldsymbol{\eta}', \boldsymbol{u}'), \\
&\left\{ \begin{array}{rcl} \boldsymbol{y}' &=& \left[y_1' \; \cdots \; y_{n_y}' \right]^T, \\ y_i' &\in& \mathbb{C}^1, \end{array} \right. \\
&\left\{ \begin{array}{rcl} \boldsymbol{\eta}' &=& \left[\eta_1' \; \cdots \; \eta_{n_{\eta'}}' \right]^T, \\ \eta_i' &\in& \mathbb{Y}_i = \left\{ \tilde{\eta}_{i,1}, \ldots, \tilde{\eta}_{i,\mu_i} \right\}, \end{array} \right. \\
&\left\{ \begin{array}{rcl} \boldsymbol{u}' &=& \left[u_1' \; \cdots \; u_{n_{u'}}' \right]^T, \\ u_i' &\in& \mathbb{U}_i = \left\{ \tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i} \right\}. \end{array} \right.
\end{aligned} \tag{6.6}$$

In this hybrid system, $\boldsymbol{\eta}$ represents the set of discrete state variables. That is, each η_i is limited to a finite set, but it is also subject to a dynamical relation indicating its value as a function of all of the states and controls. As an example, a system in which collisions may occur may fall into this class of hybrid system. A collision status, represented by η_i, is a discrete value, but it is dependent upon the other variables of a system (i.e. a collision between two objects can only occur if they are collocated).

In this case, discrete state variables become pseudo-controls in the FSCT formulation. Specifically,

$$\boldsymbol{y} \equiv \boldsymbol{y}', \tag{6.7}$$

$$\boldsymbol{u} \equiv \left[(\boldsymbol{u}')^T (\boldsymbol{\eta}')^T \right]^T, \tag{6.8}$$

$$\boldsymbol{f} \equiv \boldsymbol{f}'. \tag{6.9}$$

Thus, the discrete states are treated similarly to control variables during the optimization. Just like $\boldsymbol{u}'$, the values of $\boldsymbol{\eta}'$ are pre-specified by the user according to the finite value sets available for each variable, and the optimizer uses time duration parameters to determine the optimal switching times for both independent and dependent discrete components.

However, the constraints describing the dependencies of the discrete states in Equation 6.6 must be included in the list of constraints imposed on the underlying parameter optimization problem. That is, *additional* constraints must be devised to represent how the component, η'_i, is related to the values of t, $\boldsymbol{y}'$, $\boldsymbol{\eta}'$, and $\boldsymbol{u}'$. These, of course, depend on the specifics of the system in question.

It is observed that adding constraints is much more difficult than relaxing them. First, it is important to ensure that in adding constraints to appropriately represent the problem, the underlying parameter optimization problem does not become overconstrained. Second, care must be taken to account for how switching dependencies affect the overall description of dependent state constraints.

Further development of hybrid system optimization with discrete state variables is beyond the scope of the present investigation. Although an initial framework is established in Equations 6.7-6.9 for using the FSCT method for this class of system, it is left to future investigations to determine how appropriate constraints can be defined to effectively implement the dependencies of Equation 6.6.

6.3 Systems Bound by Partial Differential Equations

Finally, consideration is given to the problem whose dynamics are described by partial differential equations (PDEs), instead of the ordinary differential relations of Equation 1.1. Note that, in the development that follows, there is an intentional lack of rigor regarding the types and classifications of PDEs that exist. Instead, the following is intended as a sample exercise on how PDE systems with finite controls may be handled in the FSCT setting.

Introduce a new time-like quantity, τ, as a second differentiable variable. In the classical heat equation, for example, τ indicates the distance along a one-dimensional rod. Consider a first-order PDE of the form

$$\frac{\partial y'}{\partial t} + \frac{\partial f'(t, y', \boldsymbol{u}')}{\partial \tau} = 0. \tag{6.10}$$

An appropriate solution describes the scalar state as a function of the two differentiable variables, that is, $y' = y'(t, \tau)$. The state is a continuous variable, and its dynamics are related to the values of a discrete control vector, $\boldsymbol{u}'$.

$$\begin{aligned} y' &\in \mathbb{C}^{1\times 1} \\ \boldsymbol{u}' &= \begin{bmatrix} u'_1 & \cdots & u'_{n_{u'}} \end{bmatrix}^T \\ u'_i &\in \mathbb{U}_i = \{\tilde{u}_{i,1}, \ldots, \tilde{u}_{i,m_i}\} \end{aligned}$$

The transformation of this system into an FSCT-suitable formulation involves a discretization of y' over the non-time differentiation variable. Consider the FSCT state vector defined by

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n_y} \end{bmatrix} = \begin{bmatrix} y'(t, \tau_1) \\ \vdots \\ y'(t, \tau_i) \\ \vdots \\ y'(t, \tau_{n_y}) \end{bmatrix}. \tag{6.11}$$

Thus, the elements of $\boldsymbol{y}$ represent the values of y' at discrete points along $\tau \in \begin{bmatrix} \tau_1 & \tau_{n_y} \end{bmatrix}$. For simplification, assume that $\tau_1 = \tau_{\min}$, $\tau_{n_y} = \tau_{\max}$, and τ_i are uniformly distributed between the boundaries. In addition, define the FSCT control vector as simply

$$\boldsymbol{u} = \boldsymbol{u}'. \tag{6.12}$$

Using first-order forward differencing, one approximation for Equation 6.10 at τ_i becomes,

$$\begin{aligned} \left.\frac{\partial y'}{\partial t}\right|_{\tau=\tau_i} &= -\left.\frac{\partial f'}{\partial y'}\frac{\partial y'}{\partial \tau}\right|_{\tau=\tau_i} \\ &\approx -\left.\frac{\partial f'}{\partial y'}\right|_{\tau=\tau_i} \frac{1}{\Delta\tau}\left(y'\big|_{\tau=\tau_{i+1}} - y'\big|_{\tau=\tau_i} \right), \end{aligned}$$

where $\Delta\tau = \frac{\tau_{\max}-\tau_{\min}}{n_y-1}$. Relating this approximation with the FSCT state definition in Equation 6.11,

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \tag{6.13}$$

$$f_i = -\frac{\partial f'(t, y_i, \boldsymbol{u})}{\partial y_i} \frac{y_{i+1} - y_i}{\Delta\tau}, \quad i = 1, \ldots, n_y - 1. \tag{6.14}$$

If a backward differencing approximation is used at τ_{n_y}, it is observed that $f_{n_y} = f_{n_y-1}$. Thus, the partial differential relation is approximated as a vector of ordinary differential equations whose size is determined by the chosen discretization grid in τ.

The FSCT formulation is completely defined for the transformed PDE. Note that, in general, higher-order finite differencing schemes can improve the accuracy of the approximation. Many of these schemes are addressed in Chapter 2, utilizing the differentiation matrix, $\boldsymbol{D}$.

Example: Traffic Flow Management

A basic and relevant application of the FSCT method to a PDE system involves traffic flow through intersections.[59] Consider a system describing the density of traffic through two intersections, as shown in Figure 6.1. Four lanes of traffic are controlled by two traffic lights, whose conditions are indicated by the discrete controls, $\boldsymbol{u} = [u_1 \; u_2]^T$, where $u_i \in \{0, 1\}$. If $u_i = 0$, then East- and West-bound traffic see red lights. Likewise, $u_i = 1$ indicates a red light for South-bound ($i = 1$) or North-bound ($i = 2$) traffic.

In the absence of traffic lights, the density of traffic, ρ, measured in vehicles/distance unit, exhibited in a single lane is described by

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial \tau} = 0,$$

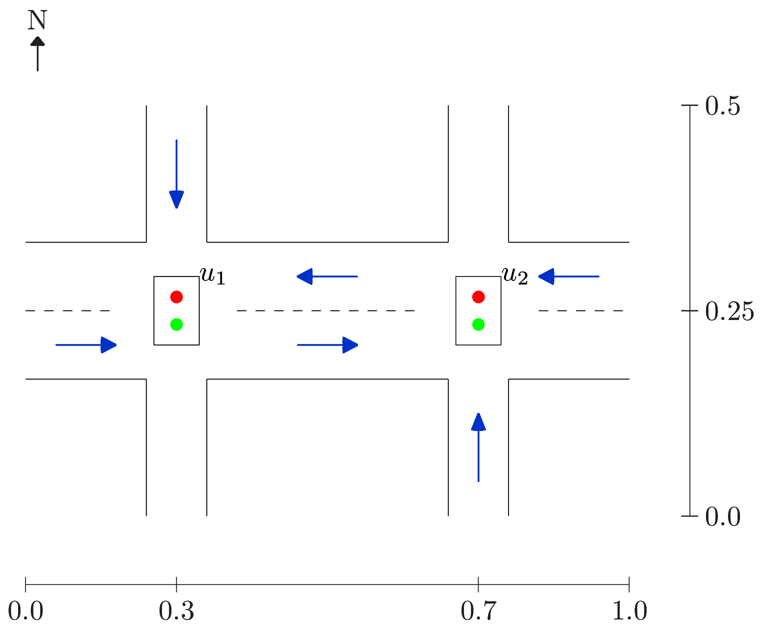


Figure 6.1: Traffic Flow Problem: Four Lanes, Two Intersections

where τ measures the position along the lane and v is the velocity of the traffic, indirectly related to the density through the relation

$$v = v_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right).$$

To simulate the effects of a traffic light, the flux,

$$f'(\rho) = \rho v = \rho v_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right)$$

is set to zero immediately after a red light. For simplicity, let $v_{\min} = \rho_{\min} = 0.0$ and $v_{\max} = \rho_{\max} = 1.0$ for this example. Thus, in this example, $f' = f'(t, \rho, \boldsymbol{u})$ is only directly dependent on ρ, and its dependence on $\boldsymbol{u}$ is limited to the points located closest to the intersections.

Discretize the East- and West-bound traffic density at 30 evenly distributed nodes between $\tau = 0.0$ and $\tau = 1.0$ distance unit. Likewise, the South- and North-bound are discretized with 20 evenly distributed nodes between $\tau = 0.0$ and $\tau = 0.5$. Let the densities at each node be placed in vectors corresponding to each lane: $\boldsymbol{\rho}_{\mathrm{E}}$, $\boldsymbol{\rho}_{\mathrm{W}}$, $\boldsymbol{\rho}_{\mathrm{S}}$, and $\boldsymbol{\rho}_{\mathrm{N}}$. Finally, define a scalar cost state, q, which accumulates over time the amount of traffic flowing through all lanes. Then the FSCT state vector is

$$\boldsymbol{y} \equiv \left[\begin{array}{ccccc} \boldsymbol{\rho}_{\mathrm{E}}^T & \boldsymbol{\rho}_{\mathrm{W}}^T & \boldsymbol{\rho}_{\mathrm{S}}^T & \boldsymbol{\rho}_{\mathrm{N}}^T & q \end{array} \right]^T,$$

and $n_y = 101$. The ordinary differential dynamics resulting from a first-order numerical approximation of the τ-differentiation becomes

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) = \left[\begin{array}{ccccc} \boldsymbol{f}_{\rho_{\mathrm{E}}}^T & \boldsymbol{f}_{\rho_{\mathrm{W}}}^T & \boldsymbol{f}_{\rho_{\mathrm{S}}}^T & \boldsymbol{f}_{\rho_{\mathrm{N}}}^T & f_q \end{array} \right]^T,$$

where in each lane,

$$f_{\rho_i} = \frac{1}{\Delta\tau}\left(F_{i+\frac{1}{2}}(\boldsymbol{\rho}) - F_{i-\frac{1}{2}}(\boldsymbol{\rho})\right),$$

and

$$F_{i+\frac{1}{2}}(\boldsymbol{\rho}) = \begin{cases} f'\left(\frac{\rho_{\max}+\rho_{\min}}{2}\right) & , \rho_i > \frac{\rho_{\max}+\rho_{\min}}{2} \text{ and } \rho_{i+1} < \frac{\rho_{\max}+\rho_{\min}}{2} \\ \min\limits_{\rho \in [\rho_i \; \rho_{i+1}]} f'(\rho) & , \rho_i \le \rho_{i+1} \\ \max\limits_{\rho \in [\rho_i \; \rho_{i+1}]} f'(\rho) & , \rho_i > \rho_{i+1} \end{cases} \quad , i = 1, \ldots, n_\rho - 1.$$

In the above sequence of equations, $n_\rho = 30$ for the East- and West-bound lanes, $n_\rho = 20$ for North- and South-bound lanes and $\Delta\tau = \frac{\tau_{\max}-\tau_{\min}}{n_\rho - 1}$. Additionally, $F_{\frac{1}{2}}(\boldsymbol{\rho}) = f'(\rho_1)$ and $F_{n_\rho+\frac{1}{2}}(\boldsymbol{\rho}) = f'(\rho_{n_\rho})$. This set of definitions represents the flow of traffic without the influence of traffic lights. The control influences the dynamics by updating flux values $F_{i+\frac{1}{2}}(\boldsymbol{\rho}) = 0$ for the grid points immediately after a red light. Finally, the dynamics definition is complete by specifying

$$f_q = f'(\rho_{\mathrm{E}_{15}}) + f'(\rho_{\mathrm{W}_{15}}) + f'(\rho_{\mathrm{S}_{15}}) + f'(\rho_{\mathrm{N}_{15}}),$$

indicating that the cost state accumulates the traffic flux for each lane at its 15th grid point. Setting the initial value of the cost state, $q_0 = 0$, the cost function $\mathcal{J} = -q_f$ maximizes the accumulated traffic flow in each lane over a specified time interval $t \in [t_0 \; t_f]$.

Sample solutions have been obtained using the formulation outlined here. By transforming the PDE into, in this case, 100 ordinary differential equations to express the time evolution of traffic density across four lanes of traffic, it is possible to determine optimal switching times for the two traffic lights. Initial conditions, indicating the distribution of traffic in each lane at t_0, have a significant effect on the switching schedule. There is also a distinct coupling between the two lights. Clearly, East-West traffic can only flow when both $u_1 = u_2 = 1$, and solutions show a synchronization between the two lights (with delay, according to which direction has larger flow). Illustrations of solutions are excluded due to the complexity inherent in a PDE system (i.e. the number of resulting state variables, for example).

Futher analysis of resulting solutions is not necessarily beneficial in the context of this investigation. For example, the sample problem illustrated in Figure 6.1 is perhaps overly simplified by using only two traffic lights. More interesting conclusions may be drawn with additional traffic lights affecting the existing North- and South-bound lanes. However, this example satisfies the current objective, establishing a framework for future analysis. This simplified example opens up a realm of PDE systems that may be explored in future investigations.

6.4 Summary

This chapter explores some extensions to the the general problem statement of Section 1.2 to expand the class of hybrid control systems whose performance can effectively be optimized using the FSCT method. The focus of this chapter is on the processes by which hybrid control problems can be transformed to fit within the confines of the FSCT formulation. It is demonstrated that this methodology easily extends to problems with continuous controls or partial differential dynamics. Discrete states are also addressed, but the success of the FSCT method on problems with dependent discrete variables may be specific to the problem at hand.

Examples are presented for dual continuous-discrete control and partial differential systems to illustrate the process of transforming the problem so that the FSCT method can be effectively applied. Results from these examples are limited to emphasize more the process. As illustrative examples, they demonstrate the direction one might take to solve more complicated or relevant hybrid control problems that do not immediately fall within the general problem statement.

Chapter 7

Conclusions

This investigation is dedicated to the development of a new methodology for treating hybrid control problems. The specific focus is on systems whose states are continuously varying and subject to differential constraints, and whose controls are discrete and limited to a finite set of values. The methodology is termed here the Finite Set Control Transcription (FSCT) method, and it is effectively demonstrated in a number of applications throughout this investigation.

The FSCT method is rooted in direct collocation and is a modified transcription of the optimal control problem into a parameter optimization problem that can be solved using any standard nonlinear programming (NLP) algorithm. In this work, SNOPT is the selected NLP solver due to the facility of solving large optimization problems with a relatively sparse Jacobian structure.

A primary distinction of this methodology is how discrete control elements of the hybrid system are optimized in the continuous context of nonlinear programming. Because all of the parameters of the underlying NLP problem are optimized over a continuous spectrum (while potentially bounded and subject to constraints), the FSCT method optimizes switching times of control variables instead of directly optimizing their values. By pre-specifying a sequence of feasible control values over a trajectory, the time durations that each control variable spends at each value is determined. The time durations for non-optimal control values are driven to zero to eliminate them from the control history. As with other collocation methods, state values at specified nodes are also included in the parameterized problem. Thus, the FSCT method optimizes states and times over a continuous spectrum to arrive at solutions to hybrid control problems.

A unique advantage of the FSCT method is that it is capable of treating problems with multiple independent control variables with ease. Complications necessarily arise when multiple control variables are optimized over their feasible sets. In this methodology, the result is a phenomenon of switching dependencies for constraint functions at each iteration of the optimization. That is, a constraint equation involving some optimization parameters at one iteration may involve a different set of parameters at the next iteration. It is shown in this investigation that steps can be taken to ensure that this does not adversely affect the convergence process of the underlying problem.

A number of different applications are explored to showcase the capability of the methodology. Some simple examples are employed to demonstrate the basic utility of the FSCT method. One- and two-dimensional problems demonstrate the method's effectiveness. Examples with complex dynamic sets are also presented to reveal the robustness and relevance of the FSCT method. Although most applications in this investigation are rooted in aerospace engineering, the methodology has applications in many other engineering and non-engineering fields.

The scope of hybrid control problems that benefit from the FSCT method is also investigated. The relevant class of hybrid systems is stretched by considering both discrete and continuous controls, both continuous and discrete states, and dynamics described by partial differential equations instead of ordinary differential equations. Successful solutions are found with continuous controls and partial differential dynamics.

Future Work

This research effort opens the door to several areas in which future efforts may further the results of this investigation. The following list summarizes some of them.

- Further Development of the Methodology
  - The Finite Set Control Transcription is presented in this investigation as an extension of a traditional collocation (direct) method. However, the same extensions may also be applied to a direct or indirect shooting method. That is, similar modifications can be made to shooting methods to treat controls that are limited to finite sets. Some interesting insight might be gained by investigating the advantages and disadvantages of FSCT-extended indirect methods.
  - An investigation in mesh refinement may reveal the effects of node distribution on FSCT solutions. In this work, an equal number of nodes is assumed for each segment. A modified formulation may allow for a specified node count for each segment. This may improve the accuracy of the ensuing solution, especially over segments that are relatively long in duration.
  - The FSCT methodology can be explored in more detail to understand its capabilities with extended classes of hybrid systems. In this investigation, some considerations are presented with regard to how a combination of continuous and discrete states in the hybrid system may be treated (Section 6.2). Further analysis may reveal new and more effective ways of optimizing systems with discrete dependent variables.
- Further Application of the Methodology
  - Many of the applications in this investigation merely demonstrate how the FSCT method can be used for effective optimization. Further applications may investigate these systems with more specific objectives. The spacecraft attitude control system presented in Chapter 4 and the traffic flow problem in Chapter 6, for example, are two applications that can be developed into more detailed analyses.

– An additional relevant application exists in the field of petroleum engineering. Smart technologies are in development for optimizing oil extraction from wells. Finite set limited variables are used to control oil output. The extension of the FSCT method to PDE systems is specifically directed as a stepping stone to this more complicated problem.

Appendices

Appendix A

A Model Predictive Controller for Real-Time Implementation

In Chapter 4, a Model Predictive Control (MPC) scheme is proposed to supplement the FSCT method. Specifically, since the latter produces trajectory and control histories, MPC techniques offer a methodology for real-time implementation of an optimal trajectory. In this appendix, a basic background in MPC is presented. More complex, and perhaps more robust, control designs are beyond the scope of this work, although developed in the theory on Model Predictive Control.[60]

First, a linear discete-time model is derived from the generalized continuous-time hybrid control system. A model predictive controller exploits the linear discrete-time model to express a real-time continuous controller, which is derived subsequently. Finally, an extension is suggested for hybrid systems subject to finite set control.

A.1 Linear Discrete-Time Model

MPC offers a method for tracking an arbitrary reference trajectory while optimizing a performance index over a finite time horizon. A basic model predictive controller is derived using a discrete-time, linear dynamic model of the form,

$$\boldsymbol{y}(t+\Delta t) = \boldsymbol{A}(t)\boldsymbol{y}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t), \tag{A.1}$$

$$\boldsymbol{z}(t) = \boldsymbol{C}\boldsymbol{y}(t). \tag{A.2}$$

Here, $\boldsymbol{z}(t)$ is the measured output from the linear system. In general, then, the nonlinear continuous dynamics,

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}), \tag{1.1}$$

must be transformed into the form of Equations A.1-A.2 through appropriate definitions of $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, and $\boldsymbol{C}$.

Note that a standard linearization of Equation 1.1 about the origin yields

$$\dot{\boldsymbol{y}}(t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t)\boldsymbol{y}(t) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(t)\boldsymbol{u}(t). \tag{A.3}$$

Then, applying the state transition matrix to the linearized dynamics of Equation A.3,

$$\begin{aligned} \boldsymbol{y}(t+\Delta t) &= \boldsymbol{\Phi}(t+\Delta t, t)\boldsymbol{y}(t) + \int_t^{t+\Delta t} \boldsymbol{\Phi}(t+\Delta t, \tau)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\tau)\boldsymbol{u}(\tau)\, d\tau && \text{(A.4)} \\ &= \boldsymbol{\Phi}(t+\Delta t, t)\boldsymbol{y}(t) + \left[\int_t^{t+\Delta t} \boldsymbol{\Phi}(t+\Delta t, \tau)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\tau)\, d\tau\right]\boldsymbol{u}(t), && \text{(A.5)} \end{aligned}$$

where

$$\begin{aligned} \dot{\boldsymbol{\Phi}}(t+\Delta t, t) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t+\Delta t)\boldsymbol{\Phi}(t+\Delta t, t), \\ \boldsymbol{\Phi}(t, t) &= \boldsymbol{I}, \end{aligned}$$

and $\boldsymbol{u}(t)$ is assumed constant over Δt to transition from Equation A.4 to Equation A.5. Comparing Equation A.5 to Equation A.1,

$$\begin{aligned} \boldsymbol{A}(t) &= \boldsymbol{\Phi}(t+\Delta t, t), \\ \boldsymbol{B}(t) &= \int_t^{t+\Delta t} \boldsymbol{\Phi}(t+\Delta t, \tau)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\tau)\, d\tau \\ &\approx \boldsymbol{\Phi}(t+\Delta t, t)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(t)\Delta t \quad \text{(for small } \Delta t\text{)}. \end{aligned}$$

Finally, $\boldsymbol{C}$ is defined according to the observable characteristics of the system. For example, consider a set of states representing positions and velocities of an object, as $\boldsymbol{y} = \left[\boldsymbol{r}^T\ \boldsymbol{v}^T\right]^T$. If only the object's positions are measurable outputs available to the controller, then $\boldsymbol{C} = [\boldsymbol{I}\ \boldsymbol{0}]$. If all states are available to determine the control law, then $\boldsymbol{C} = \boldsymbol{I}$, such that the output of the discrete-time model is a linearized approximation of all of the system states. That is, $\boldsymbol{C}$ is simply the Jacobian of $\boldsymbol{z}(t)$ with respect to $\boldsymbol{y}(t)$.

A.2 Control Law Derivation

The MPC law exploits the linear, discrete-time model to develop estimates for the observation variables, $\boldsymbol{z}(t)$, at future time intervals, given the current state values. The output *predictions* are determined for a finite horizon of future times, and current control values are chosen so that these output estimates are as close as possible to desired values of a nominal trajectory. In the traditional sense, the linear, discrete formulation allows controls to be determined by solving a linear equation, holding constant the control values over the increment Δt.

Let the estimate on the output at time $t + \Delta t$, given the states at time t, be denoted as $\hat{\boldsymbol{z}}(t + \Delta t|t)$ such that

$$\begin{aligned}
\hat{\boldsymbol{z}}(t + \Delta t|t) &= \boldsymbol{C}\hat{\boldsymbol{y}}(t + \Delta t|t) \\
&= \boldsymbol{C}\left[\hat{\boldsymbol{A}}(t|t)\hat{\boldsymbol{y}}(t|t) + \hat{\boldsymbol{B}}(t|t)\boldsymbol{u}(t|t)\right] \\
&= \boldsymbol{C}\left[\boldsymbol{A}(t)\boldsymbol{y}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t)\right].
\end{aligned}$$

In this notation, the symbol, $\hat{\cdot}$, indicates that the variable is estimated for a future time value. Notice, however, that since the states are known at time t, so are all other definitions at time t. Thus, $\hat{\boldsymbol{A}}(t|t) = \boldsymbol{A}(t)$, $\hat{\boldsymbol{B}}(t|t) = \boldsymbol{B}(t)$, $\hat{\boldsymbol{y}}(t|t) = \boldsymbol{y}(t)$, and $\boldsymbol{u}(t|t) = \boldsymbol{u}(t)$. This notation becomes more important for estimates further in the future. Consider the estimates at time $t + j\Delta t$:

$$\begin{aligned}
\hat{\boldsymbol{z}}(t + j\Delta t|t) &= \boldsymbol{C}\hat{\boldsymbol{y}}(t + j\Delta t|t) \\
&= \boldsymbol{C}\Big[\hat{\boldsymbol{A}}(t + (j-1)\Delta t|t)\hat{\boldsymbol{y}}(t + (j-1)\Delta t|t) \\
&\qquad + \hat{\boldsymbol{B}}(t + (j-1)\Delta t|t)\boldsymbol{u}(t + (j-1)\Delta t|t)\Big] \\
&= \cdots \\
&= \boldsymbol{C}\left[\prod_{i=0}^{j-1}\hat{\boldsymbol{A}}(t + i\Delta t|t)\right]\boldsymbol{y}(t) \\
&\qquad + \boldsymbol{C}\sum_{k=0}^{j-2}\left[\prod_{i=0}^{j-2-k}\hat{\boldsymbol{A}}\left(t + (j-1-i)\Delta t|t\right)\right]\hat{\boldsymbol{B}}(t + k\Delta t|t)\boldsymbol{u}(t + k\Delta t|t) \\
&\qquad + \boldsymbol{C}\hat{\boldsymbol{B}}(t + (j-1)\Delta t|t)\boldsymbol{u}(t + (j-1)\Delta t|t) \qquad \text{(A.6)}
\end{aligned}$$

Equation A.6 implies that any future estimate may be expressed as a function of $\boldsymbol{y}(t)$, the controls $\boldsymbol{u}(t)$ through $\boldsymbol{u}(t + (j-1)\Delta t)$, and the estimated values of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices through time $t + (j-1)\Delta t$. Clearly, Equation A.6 can be simplified dramatically by assuming that over the interval from t to $t + j\Delta t$, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant. If so, all estimates $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ can be replaced by $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$, respectively, and a closed form can be realized. Depending on the length of the prediction interval, this assumption may improve computational efficiency dramatically. In any case, it is possible to define predicted outputs at discrete time intervals as

$$\boldsymbol{Z} = \begin{bmatrix} \hat{\boldsymbol{z}}(t + \Delta t|t) \\ \vdots \\ \hat{\boldsymbol{z}}(t + j\Delta t|t) \\ \vdots \\ \hat{\boldsymbol{z}}(t + p\Delta t|t) \end{bmatrix},$$

where p is the *prediction interval* over which observation variables are compared to nominal values. Likewise, the controls at the discrete time intervals are collected as

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}(t) \\ \vdots \\ \boldsymbol{u}(t + j\Delta t) \\ \vdots \\ \boldsymbol{u}(t + (p-1)\Delta t) \end{bmatrix}.$$

Then the output can be expressed as

$$\boldsymbol{Z} = \boldsymbol{G}\boldsymbol{y}(t) + \boldsymbol{F}\boldsymbol{U}. \tag{A.7}$$

In Equation A.7, a linear relationship is defined between the current states, the current and future control values, and estimates of the future output. The matrices $\boldsymbol{G}$ and $\boldsymbol{F}$ are defined according to Equation A.6. Next, consider the nominal trajectory, $\boldsymbol{y}_n(t)$, to which the control law must track. Then, $\boldsymbol{C}\boldsymbol{y}_n(t)$ may be evaluated at the discrete intervals t through $t + p\Delta t$ and placed in a vector denoted as $\boldsymbol{Z}_n$.

Define a cost function of the form

$$\mathcal{J} = \frac{1}{2}(\boldsymbol{Z}_n - \boldsymbol{Z})^T \boldsymbol{Q}(\boldsymbol{Z}_n - \boldsymbol{Z}) + \frac{1}{2}\boldsymbol{U}^T \boldsymbol{R}\boldsymbol{U}.$$

It is observed that, with appropriate definitions of $\boldsymbol{Q}$, $\boldsymbol{R}$, $\tilde{\boldsymbol{Q}}$ and $\tilde{\boldsymbol{R}}$, this cost function is a linear approximation of

$$\tilde{\mathcal{J}} = \int_t^{t+p\Delta t} (\boldsymbol{y}_n - \boldsymbol{y})^T \tilde{\boldsymbol{Q}}(\boldsymbol{y}_n - \boldsymbol{y}) + \boldsymbol{u}^T \tilde{\boldsymbol{R}}\boldsymbol{u}\, d\tau,$$

which penalizes both deviations in trajectory states away from the nominal and excessive control usage. Thus, $\mathcal{J}$ can then be minimized according to user defined weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ to produce a tracking trajectory that also minimizes control over the prediction interval. To implement the model predictive controller in the traditional sense, the minimizing control,

$$\boldsymbol{U} = \left(\boldsymbol{F}^T \boldsymbol{Q}\boldsymbol{F} + \boldsymbol{R}\right)^{-1} \boldsymbol{F}\boldsymbol{Q}\left(\boldsymbol{Z}_n - \boldsymbol{G}\boldsymbol{y}\right),$$

is evaluated at every interval, and the set of entries in $\boldsymbol{U}$ associated with $\boldsymbol{u}(t)$ is implemented. Notice that this control law imposes no constraint on the values of $\boldsymbol{u}$ at any time.

A.3 Extension to the Hybrid Control System

The finite set control nature of the hybrid system in which $u_i \in \mathbb{U}_i$ motivates a minor modification to this MPC design to realize a hybrid controller. First, consider an alternate formulation of Equation A.7 in which it is assumed that $\boldsymbol{u}(t + j\Delta t) = \boldsymbol{u}(t)$ over the entire prediction interval.

$$\boldsymbol{Z} = \boldsymbol{G}\boldsymbol{y}(t) + \tilde{\boldsymbol{F}}\boldsymbol{u}(t) \tag{A.8}$$

Indeed, this is a common alternative MPC formulation, and an altered cost function can be minimized to directly determine $\boldsymbol{u}(t)$.

$$\mathcal{J} = \frac{1}{2}(\boldsymbol{Z}_n - \boldsymbol{Z})^T \boldsymbol{Q}(\boldsymbol{Z}_n - \boldsymbol{Z}) + \frac{1}{2}\boldsymbol{u}^T \tilde{\boldsymbol{R}}\boldsymbol{u}.$$

Observe that, since $\mathcal{J}$ is minimized at each interval, $\boldsymbol{u}$ may still take on different values at each interval, even though the prediction equation assumes otherwise. Also note, that in the hybrid control scenario, when controls are limited to a finite set of values which necessarily change less regularly than continuous variables, the prediction formulation of Equation A.8 is not unrealistic. More importantly, this formulation removes the control variables of $\boldsymbol{U}$ that are ultimately not implemented.

The model predictive control law for the hybrid system is defined according to

$$\boldsymbol{u} = \arg \min_{\boldsymbol{u} \in \mathbb{U}^{n_u}} \mathcal{J}.$$

Simply stated, the implemented control at the current time is simply the feasible control combination that minimizes the cost function.

Note that if each control variable, u_i may assume m_i possible values, then there are $\bar{m}$ possible control combinations, where

$$\bar{m} = \prod_{i=1}^{n_u} m_i.$$

The implemented control is determined by evaluating $\mathcal{J}$ for each control combination, selecting the minimizing combination. This extension resembles the concept of dynamic programming, and can become computationally intensive for many control variables and/or feasible control values. The control law is most effective for n_u and m_i (and therefore $\bar{m}$) reasonably small to reduce the number of computations per interval.

Appendix B

Background for Libration Point Formation Missions

Libration point formation missions are the subject of Chapter 5 and the primary motivating application for the development of the FSCT method explored in this work. For the interested reader, this appendix serves to provide additional background and context on the unique aspects of these types of missions. This background reveals the motivating force for finding a new collocation method that can treat sensitivities and limitations associated with the dynamic environment and interferometry mission requirements.

The appendix is separated into three major sections. In the first, the present state of the literature addressing libration point formations is surveyed. The methods applied to the problem to date are reviewed. The second section explores the dynamics of the regime in further detail. Although the equations of motion for the Circular Restricted Three Body Problem are presented in Chapter 5, additional context is warranted. Assumptions, advantages and disadvantages, and accuracy are addressed. Finally, a discussion on the challenges and limitations of libration point formation missions provides reasoning for exploring new tool sets for acquiring useful control solutions. The classical method of Lyapunov is briefly introduced to demonstrate how continuous techniques may fail in this dynamic regime. Additionally, considerations into the present state or propulsive technology are addressed.

B.1 Previous Work Towards Libration Point Formations

In the early 1980s, the Space Shuttle first demonstrated the capability of formation flying in conjunction with the use of its remote manipulator arm.[61] In its infancy, formationkeeping was performed by the Shuttle pilot on-board, who implemented velocity corrections manually with a control stick. Since that time, the aerospace community has been motivated to improve satellite capability for autonomous formation flying, in support of both Shuttle operations and unmanned spacecraft formation missions. As the literature has developed to address more and more issues associated with satellite formations, both NASA and the U.S. Air Force have realized the benefits of multiple spacecraft architectures, where the functionality of a single spacecraft is distributed among a number of satellites flying in close proximity.

In many cases, multiple spacecraft missions offer a means for reducing cost and risk while increasing flexibility and responsiveness. If several smaller spacecraft can perform the mission of a larger, monolithic platform, launch and design costs will be lower. In addition, a formation of spacecraft can potentially be reoriented or augmented with other members to modify or increase the architecture's capability after launch. This is rarely possible with a single spacecraft platform.

In some cases, a multiple spacecraft architecture can perform missions that single spacecraft simply cannot. Deep-space imaging is a prime example. Several missions, such as Terrestrial Planet Finder (TPF) and the Micro Arcsecond X-ray Imaging Mission (MAXIM), seek to investigate distant planets, stars, black holes, etc. To focus on deep-space objects so far away, studies[15–17] have indicated that images must be able to resolve angular offsets as small as 0.1 micro-arcsecond. Atmospheric effects significantly reduce the ability of Earth-based telescopes (limiting resolution and blocking particular wavelengths), and a single space platform housing a telescope large enough to accomplish this would be impossible to launch. The clear solution is aperture synthesis, using a collection of telescopes to produce the same angular resolution as a single telescope the size of the collection. With telescopes on board several smaller, launchable spacecraft, a formation properly spaced and oriented will be able to accomplish what nothing else can do.

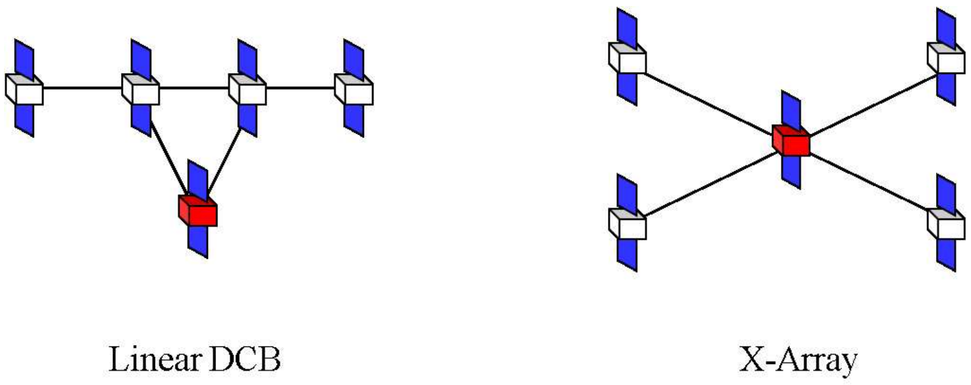


Figure B.1: TPF Interferometer Candidate Formations: Linear DCB and X-Array

NASA's Terrestrial Planet Finder[15,16] missions have the primary objective of seeking out Earth-like planets around nearby stars and characterizing their ability to sustain life. There are two complimentary missions: a coronagraph will be used for imaging in the visible spectrum, a formation flying interferometer for the infrared spectrum. Significant challenges exist in deconvolving planet signatures near their parent stars, to include the small angular offset of approximately 50 micro-arcseconds. Trade studies have determined baseline requirements for precise spacecraft formations near L_2 with between three and five spacecraft, carrying 3-4 m aperture telescopes. Two promising architectures of the TPF interferometer are the Linear Dual Chopped Bracewell (DCB) and the X-Array, both of which consist of four spacecraft carrying 3.8 m telecopes and one serving to combine the signals. Figure B.1 illustrates a rendering of these two likely configurations, ranging in size from 60 m to 240 m.

The Micro Arcsecond X-ray Imaging Mission[17] intends to place possibly 20 or more spacecraft in a formation near the Sun-Earth/Moon L_2 point. It is a concept proposed for the Structure and Evolution of the Universe (SEU) Black Hole Imager mission. In order to provide images that can resolve accretion disks around black holes, spacecraft would be placed as far away as 1 km (the diameter of the formation), with a core spacecraft in the center of the formation. A pathfinder mission with less aggressive formation requirements may lead the way to the ambitious objectives of MAXIM.

These missions, among others, propose spacecraft formations operating near the collinear L_1 and L_2 libration points of the Sun-Earth/Moon system. The nature of interferometry, along with the unique sensitivities of the dynamic regime near the libration points, motivate significant study into the formationkeeping aspects of the missions. While a lot has been accomplished in the realm of formation control to address the dynamical issues of libration point formations, the field is still quite open for determining methods of generating trajectories and control strategies that safely and effectively maintain the formations necessary to perform deep-space imaging. Previous efforts in formation control appear in the literature as early as the mid-1980s, focusing first on Earth orbiting formations. More recently, extensions have been made to libration point formations.

B.1.1 Earth Orbiting Formations

Vassar and Sherwood[62] presented some of the earliest work in formation control, motivated by attempts to automate formationkeeping operations with the Space Shuttle. Assuming a circular orbit, linearized (Clohessy-Wiltshire) equations of motion simplified the derivation of an optimal discrete-time closed-loop controller. These results were extended by Redding, Adams, and Kubiak[63] and later by Kapila, Sparks, Buffington, and Yan.[64] The former added a discrete-time linear quadratic regulator for disturbance rejection and a traveling elipse approach to stationkeeping. The latter modified the impulsive control solutions of Vassar and Sherwood to pulse-based control laws to account for advances in propulsion technology. Kapila et al. also guaranteed closed-loop stability for the linear controller.

Additional work using a linearization about a circular orbit was accomplished by Ulybyshev and Sabol, Burns, and McLaughlin. Ulybyshev[65] investigated long-term stationkeeping of satellite constellations using unique state variable assignments appropriate for prescribed intersatellite spacing. Sabol et al.[66] propagated linearized equations of motion with perturbations to determine the stability of various formation designs.

The Clohessy-Wiltshire equations effectively describe relative motion only in two-body orbits very near to circular. The full non-linear two-body equations of motion were applied by de Queiroz, Kapila, and Yan,[67] who used Lyapunov-based control techniques to guarantee exponential convergence in position tracking for spacecraft formations. They also considered adaptive control for constant or slowly-varying unknowns in spacecraft masses, disturbance forces, and gravity forces to achieve globally asymptotic convergence on tracking errors. Although the full nonlinear model is applied, an inherent assumption in standard nonlinear control techniques is that continuous control solutions are feasible for implementation.

Schaub and Alfriend[68] used Gauss's variational equations of motion to derive impulsive feedback control solutions intended to correct tracking errors in terms of mean orbital element differences between spacecraft in formation. Recently, Guibout and Scheeres[69] modeled a spacecraft formation as a Hamiltonian dynamic system to generate an accurate polynomial approximation of the system dynamics.

In addition, work has been accomplished to optimize the performance of spacecraft formations. Kong and Miller[70] used a second-order indirect multiple shooting method to determine minimum energy solutions to cluster initialization and reorientation problems. Costate estimates for initial guesses were obtained using the solution to the linearized quadratic tracking problem. Similarly, Scott and Spencer[71,72] investigated optimal low-thrust reconfigurations with a combined linearized dynamics and calculus of variations approach to examine trends of a basic single transfer. Massari, Armellin, and Finzi[73] used a direct optimization algorithm (single shooting) with linear, time-varying differential equations to describe relative motion in a general elliptical reference orbit. Optimal trajectories were generated for low-thrust reconfigurations, and "dynamic refresh" was proposed for real-time control when ideal and real trajectories diverged.

B.1.2 Libration Point Trajectories and Formations

While research in Earth orbiting formations is useful, many additional challenges arise in placing formations in a three-body (in general, n-body) dynamic environment. Interest in libration point formations has motivated significant work[42–57,74–77] to characterize and negotiate this unstable and highly sensitive regime. For interferometry missions, the formation requirements are even more restrictive, and existing literature highlights the difficulties in designing trajectories and control strategies to effectively operate in this environment.

Early investigations focused on designing trajectories for spacecraft near libration points. Howell and Keeter[43] used target point and Floquet approaches to determine impulsive maneuvers at discrete time intervals to maintain trajectories. Although work was based in the Circular Restricted 3-Body Problem (CR3BP), it was extended to more complex models. Barden and Howell[74] applied dynamical systems theory to discover trajectories based on halo orbits and Lissajous trajectories. It was distinguished by a focus on the center manifold, as opposed to simply the stable and unstable manifolds.

Continuous control techniques have also been developed for arbitrary reference trajectories in the CR3BP. Scheeres and Vinh[45] developed a non-traditional continuous controller achieving bounded motion near a halo orbit by exploiting the local eigenstructure of the linear system. Along with Hsiao,[46] they developed a stabilizing control law intended for a low-thrust ion engine. Gurfil, Idan, and Kasdin[47,48] designed a relative position controller using nonlinear adaptive control techniques with a neural network based element compensating for model inversion errors. Gurfil and Kasdin[49] also derived a time-varying continuous linear quadratic control law with linearized dynamics about an arbitrary reference. They recommended plasma eletric propulsion, recognizing the necessity for micro-thrusting capability. Luquette and Sanner[44] designed a Lyapunov-based nonlinear controller based on the relative states between two spacecraft. This extended the work of de Queiroz et al. to the three-body problem.

Continuous control techniques were extended to a more complete dynamic model, as well. Marchand and Howell[54] applied existing linear (LQR) and nonlinear (feedback linearization) methods to spherical and aspherical formulations. Baseline propulsive requirements were established, again highlighting the restrictively low thrusting requirements, bringing into the forefront the limitations of existing propulsion technology. In subsequent work,[42,50–53] they investigated further input and output feedback linearization for non-natural formations, along with insights into naturally existing formations, generalizing to an n-body dynamic model which included solar radiation pressure.

Some efforts have been specifically applied to particular libration point missions. Folta, Hartman, Howell, and Marchand[76] applied continuous nonlinear control to the MAXIM formation requirement specifications. Gomez, Lo, and Masdemont[77] investigated impulsive manuevers to maintain nominal paths for the TPF mission.

Efforts to optimize the trajectories for libration point formations are also observed. Marchand, Howell and Betts[56] used numerical optimization techniques to find optimal discrete (impulsive) control laws. Infeld, Josselyn, Murray, and Ross[57] use their Legendre pseudospectral method on Sun-Earth and Earth-Moon L_2 point trajectories in the CR3BP to minimize fuel consumption.

In summary, the work to date can be characterized by

- the dynamic model used to describe the motion,
- the class of the control scheme, and

- the optimality of the controller's performance.

The dynamic model for libration point orbits is at a minimum the CR3BP, but more complete models incorporate other planetary and solar effects (using ephemeris data for other bodies, solar radiation pressure, etc.) Control schemes can also be characterized by whether a linearization of the dynamics is required to derive it. Control schemes can either be discrete (impulsive) or continuous, where both are potentially impossible to implement. While impulsive control assumes state discontinuities, continuous control solutions may requires prohibitively small control accelerations. Finally, some work has used various optimality schemes in their derivation (LQR, numerical approximations) while others have not.

B.2 Dynamical Description

Libration points are best understood in the context of a 3-body model. Consequently, the Circular Restricted Three Body (CR3B) model is used in this investigation. Therefore, the assumptions of the CR3BP are presented, along with arguments for their relevance. For completeness, the assumptions are relaxed to form a more general n-body model that would be equally effective for additional libration point analysis with the FSCT method.

With each model, a relative set of the dynamics is necessary for effective description of a spacecraft formation. This is primarily due to the scale of the problem. To quantify this, consider a spacecraft formation whose primary gravitational effects are attributed to the Sun and the Earth. An expected distance from the Sun to the formation is $\mathcal{O}(10^8)$ km, while from the Earth to the formation is $\mathcal{O}(10^6)$ km. On the other hand, the relative distance between spacecraft may be less than 1 km. Common reference frames, with heliocentric or geocentric origins, may not capture the fidelity of the relative motion between spacecraft. A simple solution is to adopt a local point as the origin of the reference frame.

Let there be a spacecraft in the formation that is termed the *chief.* For an interferometer formation, for example, the chief may be the spacecraft designated to coordinate the incoming signals from the surrounding telescope-bearing spacecraft. In this case, the chief acts as the combiner. Regardless of its duties, let the chief serve as the reference point for a set of equations of relative motion. That is, every other spacecraft in the formation is termed a *deputy*, and its states are measured *relative to the chief.* To simplify the dynamics significantly, let the following assumption hold for all situations:

Assumption B.2.a. The chief spacecraft shall lie along a natural trajectory.

Note that under this condition, the chief does not require any external control to remain along its course. Note also that because of this assumption, the chief does not need to be an actual satellite. A virtual point along a natural trajectory acting as the reference point for relative motion may also serve the purpose of the chief spacecraft. Whether an actual or virtual spacecraft, in most cases, the chief's path is along a bounded trajectory, such as a halo orbit. Since this is not a necessary restriction in developing the dynamics, however, it is not included in the assumption. Thus, it can be inferred that this is not a restrictive assumption. Any general problem can be stated relative to the moving point identified as the chief.

B.2.1 Circular Restricted 3-Body Problem

Consider a dynamic regime governed by the following assumptions.

Assumption B.2.b. Two primary bodies constitute the gravitional forces acting on a third body. The primaries are considered to be point masses, and the mass of the third body is negligible in comparison to that of the other two.

Assumption B.2.c. The smaller primary moves in circular motion about the larger. This ensures that the distance between the two primaries remains constant, and the vector pointing between the primaries rotates at a constant angular rate.

These are the fundamental assumptions of the CR3BP, useful for describing satellite motion in an Earth-Moon or a Sun-Earth regime, for example. For the purpose of this work, let the first primary be the Sun, and the second a combined Earth/Moon mass acting at the Earth/Moon barycenter.

The equations of motion for the CR3BP, applying these assumptions, are presented in Section 5.2. Absolute equations are presented and modified into a relative set of equations, in which a deputy spacecraft's position and velocity are measured relative to the chief.

B.2.1.1 Dimensionalized vs. Nondimensionalized Equations

In most literature, the CR3BP equations of motion are presented in nondimensionalized units, modifying Equation 5.1 slightly, and introducing a parameter, μ, to describe the ratio between masses (or distances) of the two primaries. It is noted that the equations presented in Section 5.2 are dimensionalized instead. However, for completeness, the nondimensionalized equations are presented.

Let the following characteristic quantities be defined:

$$\begin{aligned} r^* &= r_\odot + r_\oplus \\ m^* &= m_\odot + m_\oplus \\ t^* &= \frac{1}{n} = \frac{1}{\omega}, \end{aligned}$$

where $r_\odot$ and $r_\oplus$ are the distances of the Sun and Earth/Moon, respectively, from the barycenter. For simplicification, introduce the mass ratio μ as

$$\begin{aligned} \mu &= \frac{m_\oplus}{m^*} = \frac{r_\odot}{r^*} \\ 1-\mu &= \frac{m_\odot}{m^*} = \frac{r_\oplus}{r^*}. \end{aligned}$$

Finally, nondimensionlized states and time can be identified through the characteristic quantities.

$$\xi = \frac{x}{r^*}, \;\; \eta = \frac{y}{r^*}, \;\; \zeta = \frac{z}{r^*}, \;\; \rho = \frac{r}{r^*}, \text{ and } \tau = \frac{t}{t^*}$$

Using the derivative abbrevation $(\cdot)' = d(\cdot)/d\tau$, Equation 5.1 in a nondimensionalized form appears as

$$\begin{bmatrix} \xi_c'' \\ \eta_c'' \\ \zeta_c'' \end{bmatrix} = -\frac{1-\mu}{\rho_{\odot c}^3} \begin{bmatrix} \xi_c + \mu \\ \eta_c \\ z_c \end{bmatrix} - \frac{\mu}{\rho_{\oplus c}^3} \begin{bmatrix} \xi_c - 1 + \mu \\ \eta_c \\ \zeta_c \end{bmatrix} + 2 \begin{bmatrix} \eta_c' \\ -\xi_c' \\ 0 \end{bmatrix} + \begin{bmatrix} \xi_c \\ \eta_c \\ 0 \end{bmatrix}. \qquad \text{(B.1)}$$

Notice that the angular rate, ω, disappears due to cancellations with the characteristic time, t^*.

B.2.1.2 Libration Points and Some Natural Trajectories

Equations 5.1 and B.1 have five equilibrium points, known as *libration points.* Three of the points, L_1, L_2, and L_3, lie along the same line as the two primary bodies, and are therefore termed *collinear* libration points. The other two, L_4 and L_5, or *triangular* points, make up equilateral triangles in the orbital plane with the two primaries. Thus, the triangular points are located at positions $[\xi \; \eta \; \zeta] = \left[\left(\frac{1}{2} - \mu\right) \; \left(\pm\frac{\sqrt{3}}{2}\right) \; 0\right]$. The collinear points, with L_1 between the two primaries, L_2 beyond the second primary, and L_3 in front of the first primary, are more difficult to determine. Along with the triangular points, they make up the roots of a quintic polynomial equation which cannot be solved analytically. For a given value of μ, any nonlinear equation solver can be used to determine the collinear points.

Although the collinear equilibria are inherently unstable, near these points exist several families of trajectories that exhibit bounded motion, such as halo orbits or Lissajous trajectories. It is not within the scope of this work to survey all of the types of natural motion existing near libration points. Recall, however, that in Chapter 5, the chief trajectory is a halo orbit. The techniques used to determine the initial conditions for this orbit follow the work of Howell,[58,78,79] and are not repeated here.

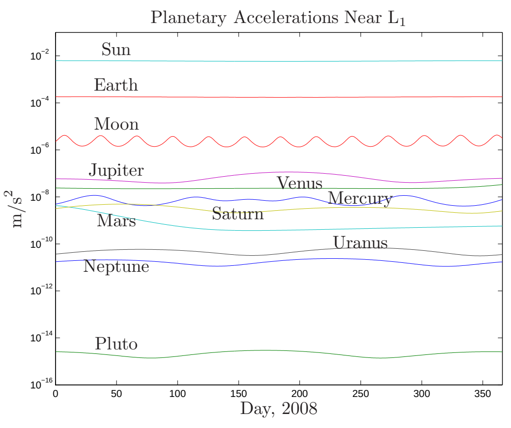
B.2.1.3 Accuracy of the CR3BP Equations of Motion

The CR3BP equations of motion of Equations 5.1 or B.1 are based on Assumptions B.2.b and B.2.c. Thus, it is assumed that in the Sun-Earth/Moon system, the second primary orbits the first with circular motion. In actuality, the Earth's orbit about the Sun has an eccentricity of approximately 0.0167, according to Vallado.[38] In addition, gravitational effects exist from other bodies, most notably the other planets and dwarf planets in the Solar System. The assumptions of the CR3BP clearly degrade the accuracy of the dynamic model, and the question remains as to the extent.

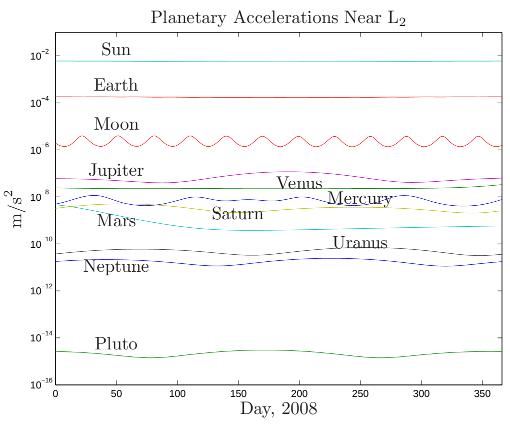
Consider the collinear libration points, L_1, L_2, and L_3, from the Sun-Earth/Moon circular restricted problem. Naturally, with perturbations due to other gravitational effects or solar radiation pressure, a spacecraft located near these equilibrium points will quickly diverge from there. To quantify the extent of these perturbations near the libration points, refer to Figure B.2, which illustrates the magnitude of the major acceleration effects near each of the collinear libration points over the year 2008. Ephemeris data from the Jet Propulsion Laboratory is used to calculate the gravitational accelerations at the three points, indicating the extent of the effects due to each planet in the Solar System, plotted logarithmically. For this analysis, the libration points are calculated as relative distances from the Sun and the Earth/Moon barycenter. This is to say that even though the distance between the Earth and the Sun changes according to the eccentricity of its orbit, the libration points are calculated such that

$$\frac{r_{\mathrm{L}_i}}{r_{\oplus}} = \text{constant}.$$

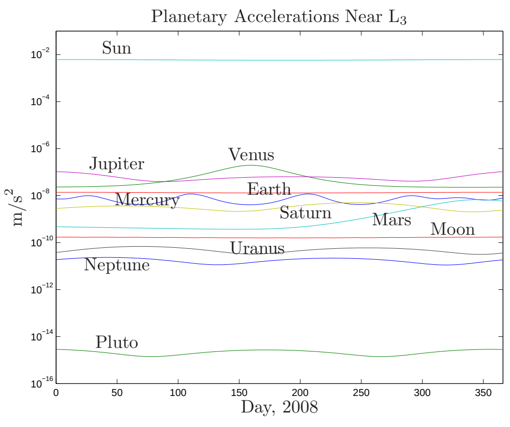
Since from these points, the distance to each body, including the Sun and the Earth, changes with time, the gravitational effects will vary over the course of a year.



(a) L_1



(b) L_2



(c) L_3

Figure B.2: Planetary Accelerations for the Collinear Libration Points for 2008

It is apparent that for spacecraft near L$_1$ or L$_2$, the primary acceleration effects come from the Sun, the Earth, and the Moon. However, Jupiter, Venus, Mercury, Saturn, and Mars still contribute gravitational effects greater than 10^{-10} m/s^2. To put this in context, previous and current work demonstrate control solutions with magnitudes less than 10^{-9} m/s^2 (see Section B.3.1). This indicates that extra planetary effects are significant players in defining the dynamic environment for libration point spacecraft formations. Naturally, the effects of the Earth and Moon drop dramatically at L$_3$. However, the argument remains the same with regard to the significance of extra planetary effects.

B.2.2 Ephemeris Model

An alternative to the simplified 3-body model is a general n-body model using Ephemeris data to compute the relative locations of the planets. The relative motion of an n-body system can be described by

$$\ddot{\boldsymbol{r}}_{jc} = -\frac{G(m_j + m_c)}{r_{jc}^3}\boldsymbol{r}_{jc} + G\sum_{\substack{i=1 \\ i\neq j,c}}^{n} m_i\left(\frac{\boldsymbol{r}_{ij}}{r_{ij}^3} - \frac{\boldsymbol{r}_{ic}}{r_{ic}^3}\right). \tag{B.2}$$

This computes the acceleration of the c^{th} body relative to the j^{th} body, where $\boldsymbol{r}_{jc} = \boldsymbol{r}_c - \boldsymbol{r}_j$ and $r = |\boldsymbol{r}|$. In the case of the Sun-Earth/Moon libration formation, let the cth body be the chief satellite, whose mass, m_c, is negligible. The j^{th} body, from which the chief dynamics are measured, can either be the Sun or the Earth. As seen from a non-accelerating point, such as the Solar System barycenter, the acceleration of the chief can be modeled as

$$\ddot{\boldsymbol{r}}_c = -\sum_{i=1}^{n}\frac{Gm_i}{r_{ic}^3}\boldsymbol{r}_{ic}. \tag{B.3}$$

Again, the dynamics desired describe a deputy spacecraft relative to the chief spacecraft. Using either Equation B.2 or B.3, by assuming negligible mass for all satellites, the relative dynamics between chief and the l^{th} deputy is

$$\ddot{\boldsymbol{r}}_{cd_l} = -\sum_{i=1}^{n} Gm_i\left(\frac{\boldsymbol{r}_{id_l}}{r_{id_l}^3} - \frac{\boldsymbol{r}_{ic}}{r_{ic}^3}\right).$$

Once this is placed in first-order form, including a term for external control accelerations, the dynamics can be expressed as

$$\dot{\boldsymbol{y}}_l = \boldsymbol{f}(t, \boldsymbol{y}_l, \boldsymbol{u}_l),$$

just as with the CR3BP. In this case, however, the time element is not related to an epoch, t_0, but rather represents a Julian date indicating the locations of the planetary bodies.

B.2.3 Pros and Cons of the Two Models

It has already been demonstrated that the CR3BP has limitations in accurately modeling the dynamics near libration points. Since potential control accelerations are on the same order of magnitude as some of the perturbing bodies (beyond the two primaries considered in the 3-body model), it is crucial to consider those bodies when constructing a dynamic model.

The numerical methods presented in this work are equally effective in any dynamic regime. That is, the underlying methods do not need to be altered to account for a different set of dynamics. Therefore, in demonstrating the solution methods, it is most beneficial to use a simpler dynamic model to validate the technique and to gain insight into the solutions. The CR3BP model, then, offers an excellent platform off of which to develop solution methods.

B.3 Challenges and Sensitivities for Libration Point Formations

Some of the sensitivity of the dynamic regime near libration points has already been implied. It has been stated that extra planetary gravitational accelerations have an impact on the dynamics on the same order of magnitude as potential control accelerations. This is demonstrated presently, highlighting the importance of a high fidelity model in accurately describing satellite motion. In addition, consideration is given to even more sources of sensitivity that contribute to the challenge of the problem. They can be demonstrated in either dynamic environment, so for simplicity the CR3BP model is used.

It is conceivable that operating requirements on an interferometer formation may involve tight constraints. In that light, the following assumptions are introduced.

Assumption B.3.a (Formation Size Constraints). The relative distances between spacecraft (deputy-chief and deputy-deputy) are constrained.

Assumption B.3.b (Formation Orientation Constraints). The orientation of the formation plane is constrained in inertial space or relative to a particular body.

Assumption B.3.c (Formation Rotation Constraints). The formation must rotate within the formation plane at a constrained rotation rate.

Assumption B.3.d (Spacecraft Orientation Constraints). The attitude of each spacecraft is constrained relative to a point in inertial space, a particular body, and/or other spacecraft in the formation.

At this point, Assumptions B.3.a-d are left in a general form. That is, tolerances are not assigned to quantify the constraints imposed. It should be understood, however, that the nature of interferometry implies that these constraints are significantly restrictive. At a minimum, it is safe to say that size, orientation, and rotation requirements should keep spacecraft in formation *as close as possible* to a set of nominal values.

B.3.1 Tracking a Nominal Trajectory with an Unconstrained Control Scheme

One way of addressing Assumptions B.3.a-c is to identify trajectories for each spacecraft that satisfy the nominal values of the formation size, orientation, and rotation constraints. A spacecraft can be assigned to track a nominal trajectory, and any number of continuous or discrete control laws may be used to ensure stable tracking.

As an example, consider a deputy spacecraft in formation around a chief in the L_1 halo orbit used in Chapter 5. A nominal trajectory is assigned to the deputy to ensure a constant distance between spacecraft and a constant rotation rate. In this case, let the nominal be a circular trajectory about the chief in the $\hat{y}_{\mathfrak{R}}$-$\hat{z}_{\mathfrak{R}}$ plane. For the sake of this example, the nominal formation parameters are arbitrarily chosen. The nominal trajectory can be expressed as a function of time as

$$\boldsymbol{y}_n(t) = \begin{bmatrix} \boldsymbol{r}_n(t) \\ \boldsymbol{v}_n(t) \end{bmatrix}$$

where

$$\boldsymbol{r}_n(t) = \begin{bmatrix} 0 \\ \cos\left(\frac{\pi}{4} - \frac{2\pi(t-t_0)}{t_{\text{Per}}}\right) \\ \sin\left(\frac{\pi}{4} - \frac{2\pi(t-t_0)}{t_{\text{Per}}}\right) \end{bmatrix} \text{ km}$$

and $\boldsymbol{v}_n = \dot{\boldsymbol{r}}_n$ to satisfy a traditional matching constraint. The constant, t_{Per}, represents the period of the chief's halo orbit. At $t = t_0$, the deputy spacecraft's actual states coincide with the nominal trajectory, that is, $\boldsymbol{y}(t_0) = \boldsymbol{y}_n(t_0)$, so that control is only required to maintain the formation. Observe the tracking control law that result using a traditional continuous control scheme.

B.3.1.1 Lyapunov-Based Controller

A continuous control law can easily be derived through a standard nonlinear Lyapunov analysis. From Equation 1.1, the form can easily be modified to separate the control terms, which appear linearly.

$$\begin{aligned} \dot{\boldsymbol{y}} &= \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{u}) \\ &= \begin{bmatrix} \dot{\boldsymbol{r}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \overline{\boldsymbol{f}}(t, \boldsymbol{r}, \boldsymbol{v}) + \boldsymbol{u} \end{bmatrix} \end{aligned}$$

Error dynamics can be defined to relate the actual trajectory to the nominal. Let the error in the position states be $\boldsymbol{e}_r = \boldsymbol{r} - \boldsymbol{r}_n$, and for the velocities, $\boldsymbol{e}_v = \boldsymbol{v} - \boldsymbol{v}_n$. Thus,

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{e}}_r \\ \dot{\boldsymbol{e}}_v \end{bmatrix} &= \begin{bmatrix} \dot{\boldsymbol{r}} - \dot{\boldsymbol{r}}_n \\ \dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} - \boldsymbol{v}_n \\ \overline{\boldsymbol{f}}(t, \boldsymbol{r}, \boldsymbol{v}) - \dot{\boldsymbol{v}}_n + \boldsymbol{u} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{e}_v \\ \Delta\overline{\boldsymbol{f}} + \boldsymbol{u} \end{bmatrix}. \end{aligned}$$

Defining the control law,

$$\boldsymbol{u} = -\Delta\overline{\boldsymbol{f}} - \boldsymbol{K}_r\boldsymbol{e}_r - \boldsymbol{K}_v\boldsymbol{e}_v,$$

where $\boldsymbol{K}_r = \boldsymbol{K}_r^T > \boldsymbol{0}$ and $\boldsymbol{K}_v = \boldsymbol{K}_v^T > \boldsymbol{0}$, the velocity error dynamics reduce to

$$\dot{\boldsymbol{e}}_v = -\boldsymbol{K}_r\boldsymbol{e}_r - \boldsymbol{K}_v\boldsymbol{e}_v.$$

With the decrescent, positive definite Lyapunov function

$$V = \frac{1}{2}\boldsymbol{e}_r^T\boldsymbol{K}_r\boldsymbol{e}_r + \frac{1}{2}\boldsymbol{e}_v^T\boldsymbol{e}_v,$$

analysis can show that

$$\dot{V} = -\boldsymbol{e}_v^T\boldsymbol{K}_v\boldsymbol{e}_v \leq 0,$$

and

$$\lim_{t\to\infty}\boldsymbol{e}_v = \lim_{t\to\infty}\boldsymbol{e}_r = \boldsymbol{0}.$$

This control law offers asymptotically stabilizing control to track to an arbitrary trajectory from any epoch position and velocity. Of course, when the epoch values are along the trajectory as in this example, the error dynamics will remain at zero while the control cancels out the natural dynamics. This, of course, is in the presence of no perturbations. Figure B.3 displays the resulting control history from a simulation of this scenario over a time period of 60 days (approximately a third of the halo orbital period).

Notice that the unconstrained law derived using nonlinear control theory requires a magnitude of the control acceleration never exceeding 2.5×10^{-10} m/s^2 over the duration

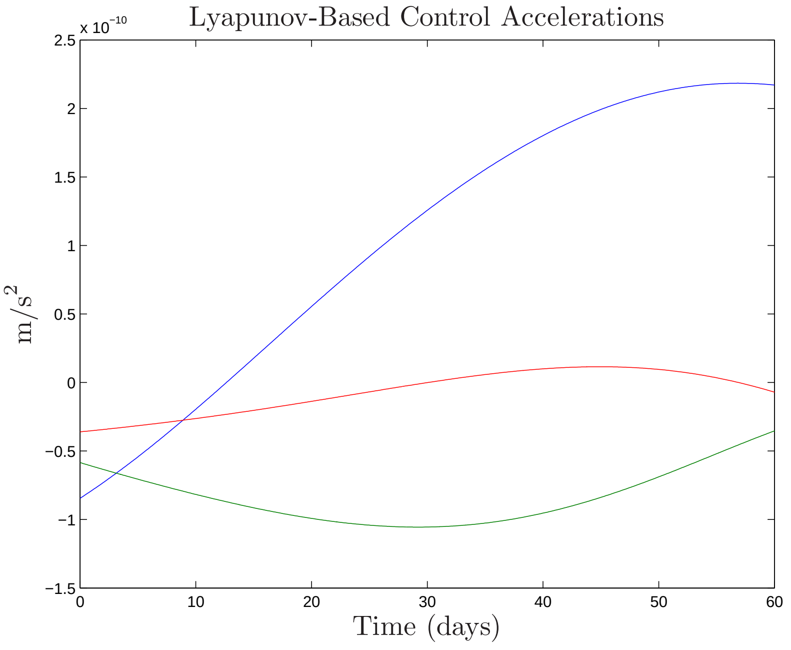


Figure B.3: Lyapunov-Based Controller for an Arbitrary Libration Point Trajectory

of the simulation. It is an underestimate to say that these magnitudes are extremely small. These control results are consistent with previous efforts in this realm by other authors, bringing to the forefront several potential issues. First, these small control acceleration requirements support the claim that the perturbations due to extra planetary accelerations cannot be ignored. Because planets like Jupiter and Venus perturb the motion of a spacecraft near a libration point *even more* than the control accelerations would as compared in the CR3BP, required control accelerations will be distinctly affected by extra bodies beyond the Sun and Earth/Moon.

A more important concern regarding these control magnitudes constitutes the main motivation for this research effort. Such small magnitudes brings into question the capabilities of existing propulsive technology. Is it possible for a spacecraft to deliver the required actuation to control (maintain and maneuver) spacecraft in formation near libration points? The remaining discussion seeks to answer this question while framing the motivation for this investigation.

B.3.2 Current Capability of Propulsive Technology

With recent trends in spacecraft design towards micro- and nanosatellite platforms, there has been considerable effort to reduce the lower bounds on thrust on secondary propulsion systems for orbit maintenance and precision pointing/flying. This is timely, as libration point formation control presents one of the more restrictive sets of propulsive requirements, as exhibited by the commanded control accelerations required in Figure B.3.

In current operations, the smallest thrust magnitudes are produced by pulsed plasma thrusters like those used on EO-1, which operate with thrust levels as low as 90 to 1000 μN.[42] However, some exciting new technologies using on-board lasers can potentially produce even smaller thrust levels.[19–22] For example, Gonzales and Baker have investigated laser ablation of aluminum to see a thrust range between 0.5 nN and 5 μN. Similarly, a laser plasma thruster (LPT) developed by Phipps and Luke is characterized by a 1 nN/s impulse for as small as 100 μs pulses.

A high-level description of the niches for various thruster technologies is presented by Gonzales and Baker,[20] and is reproduced in Figure B.4. On-board Laser Ablation Thrusters (OLAT) are compared with Field Emission Electric Propulsion (FEEP), pulsed plasma thrusters (PPT), micro-PPT, and colloid thrusters in terms of thrust magnitudes and specific impulse.

With the sample control requirements developed in Section B.3.1 along with some basic (order of magnitude) estimates on potential spacecraft sizes, one can gain a sense of how well these potential propulsive capabilities can meet the challenge. Control requirements obviously vary according to the specific conditions of a formation, and the values for

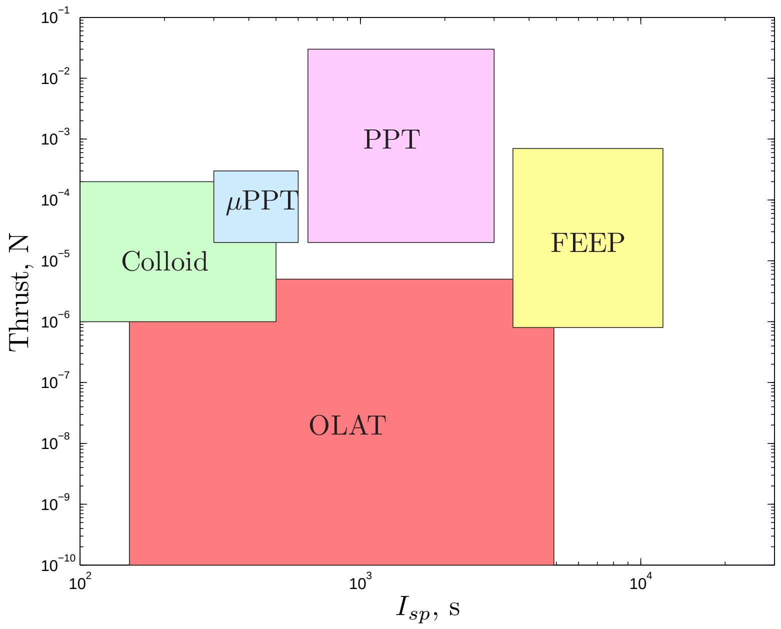


Figure B.4: Niches for Current Thruster Technology [20]

propulsive capabilities presented are only approximations. Accordingly, no precise conclusions can be drawn from the data. However, even a 'back of the envelope' calculation can serve to highlight a potential discrepancy between requirement and capability.

In Table B.1, several propulsion technologies are compared in terms of their *minimum thrust* and *minimum specific impulse*. These two characteristics give the best sense of how precise and how efficient control actuation can be. A baseline technology is also listed as a representative value used in analysis of Chapter 5, in the theoretical range of currently flight-ready technologies. The associated minimum thrust acceleration is shown as if a single thruster is placed on either a 500 kg or 1000 kg satellite. In addition, an estimate is provided of the total mass consumption over a year of *continuous thrusting* using the ideal

Table B.1: Thruster Technologies Against Spacecraft Size Estimates

			$m_0 = 500$ kg			$m_0 = 1000$ kg		
				365 days cont. thrusting			365 days cont. thrusting	
Technology	min. Thrust (N)	min. I_{sp} (s)	Thrust Acc. (m/s^2)	Δm (kg)	$\Delta m/m_0$	Thrust Acc. (m/s^2)	Δm (kg)	$\Delta m/m_0$
PPT	9.0E-5	800	1.8E-7	3.62E-1	0.07233%	9.0E-8	3.62E-1	0.03617%
µPPT	5.0E-5	400	1.0E-7	4.02E-1	0.08036%	5.0E-8	4.02E-1	0.04019%
Colloid	1.0E-6	100	2.0E-9	3.22E-2	0.00643%	1.0E-9	3.22E-2	0.00322%
FEEP	9.0E-7	4000	1.8E-9	7.24E-4	0.00014%	9.0E-10	7.24E-4	0.00007%
OLAT	1.0E-9	200	2.0E-12	1.61E-5	0.00000%	1.0E-12	1.61E-5	0.00000%
LPT	1.0E-13	200	2.0e-16	1.61e-9	0.00000%	1.0E-16	1.61E-9	0.00000%
Baseline	1.0E-6	1000	2.0E-9	3.22E-3	0.00064%	1.0E-9	3.22E-3	0.00032%

thrust equation,[39]

$$\Delta v = I_{sp} g_0 \ln\left(\frac{m_f}{m_0}\right),$$

where the accumulated Δv is the constant thrust acceleration integrated over one year, $g_0 = 9.807\text{m/s}^2$, and $\Delta m = m_0 - m_f$.

The thrust accelerations of Table B.1 should be compared to the control profile in Figure B.3. Clearly, on-board laser technology offers a window for potentially actuating the precise control requirements derived from libration point formation analysis. However, it should be apparent that with the potential exception of the OLAT or LPT technologies, the minimum thrust accelerations from existing propulsive technology may not be able to produce thrust magnitudes as small as an unconstrained control law would require.

Concluding that the thrust capability of a technology is on the same order of magnitude as the desired control does not sufficiently reveal whether it can effectively actuate the types of control histories of Figure B.3. The control histories require varying control accelerations over time, and the required precision of the control acceleration is most likely several orders of magnitude *smaller* than the total magnitudes displayed. Combined with the fact that some of these technologies are relatively new and untested, there is some uncertainty in how well a thruster can perform.

This uncertainty motivates the consideration of how *existing* propulsive technology could be used to control libration point formations. Without relying on a capability that may not be available on the same timeline as a libration point mission, a reasonable approach is to assume a baseline propulsive capability, and then determine how it can be used to meet theoretical or actual mission requirements. The baseline entry in Table B.1 serves this purpose.

B.3.3 Addressing the Discrepancy Between Desired (Unconstrained) and Feasible (Constrained) Controls

This investigation seeks an alternative control scheme that does not place the onus on propulsive technology for delivering maintenance and maneuver control to a spacecraft formation, specifically near libration points. The problem can be generalized as answering the following question:

> In what ways can a spacecraft be controlled when the available thrust acceleration level is larger than the desired (unconstrained) control solution?

In other words, there is an active *lower* bound on the thrust magnitude. In this investigation, considerations are limited to cases where the desired control is always smaller than the available control. Thus, there is no need to address an upper bound on thrust.

Referring again to Table B.1 it is apparent that the fuel consumption is very small for the types of thrusters considered. Recall that the percentages listed are for a thruster activated continuously for an entire year, and that these percentages would be even smaller if considering specific impulses greater than their minima. A simple conclusion from this result is that maintenance-type maneuvers on a spacecraft formation will contribute negligible mass change on the spacecraft. These conditions motivate another key assumption for this effort.

Assumption B.3.e (Finite Control Acceleration). The control available to a spacecraft is limited to constant acceleration in the direction of thrust.

Because the mass of the spacecraft remains constant, control can be represented as an acceleration as opposed to a thrust. Because the desired acceleration magnitude is always less than the deliverable acceleration magnitude, a thruster only operates at its minimum value. Thus, thruster control is always at a constant value of the acceleration. Either the thruster produces the acceleration associated with its minimum thrust (when turned on), or it produces zero acceleration (when turned off).

The coupling of Assumptions B.3.d and B.3.e cannot be ignored. When a control acceleration can only have two values (on and off), the direction of the thrust vector becomes crucial. However, the spacecraft orientation is constrained in some way in order to perform the mission. Thus without even considering the attitude dynamics problem, the spacecraft orientation has a significant impact on the translational dynamics problem!

Further assumptions are required regarding the orientation constraints on a spacecraft, but for this discussion, it suffices to consider some possible configurations of the spacecraft structure. For this hypothetical discussion, consider a two-dimensional spacecraft depicted with two different thruster configurations in Figure B.5. For both, the primary structure of the spacecraft is assumed to be a rigid box. In the first configuration, a thruster is located on each of the four spacecraft faces to deliver translational acceleration to the vehicle. If each thruster operates at an acceleration level of T^* when activated, then the following control accelerations are possible:

$$u_x, u_y \in \mathbb{U} = \{-T^*, 0, T^*\} . \tag{B.4}$$

Note that a net acceleration of zero results when either corresponding direction thrusters are both on or both off. In the second configuration, thrusters are located on the corners of the spacecraft surface, as well. Depending upon the angle at which the thrusters are

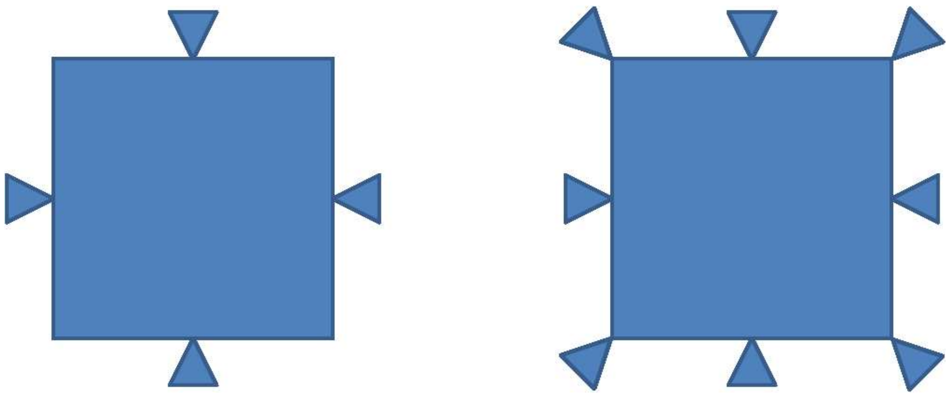


Figure B.5: Two Thruster Configurations

placed, significantly more acceleration magnitudes—both smaller and larger than T^*—can result by turning on specific combinations of thrusters.

B.3.3.1 Variable Thrust Through Gimballing

An extention of the hypothetical structure configuration considers the gimballing of thruster outlets to vary the thrust vector. It is observed that gimballed thrust offers a solution to the constrained control problem. Consider, for example, the configuration displayed in Figure B.6, representing thrust acceleration contributions in the $\hat{\boldsymbol{x}}$ axis only. Notice for this example, that one thruster contributes acceleration purely in the $\hat{\boldsymbol{x}}$ direction at magnitude T^*, while the other two present acceleration along $-\hat{\boldsymbol{x}}$ according to $T^* \cos\theta$. Let all three thrusters fire simultaneously, where θ is allowed to vary between 0 and 90 degrees. Since the gimballed thrusters move together, it is observed that translational accelerations in $\hat{\boldsymbol{y}}$ and rotational acceleration terms cancel, leaving the thruster system to only operate in the $\hat{\boldsymbol{x}}$-direction. The system produces control accelerations according to

$$u_x = T^* \left(1 - 2\cos\theta\right).$$

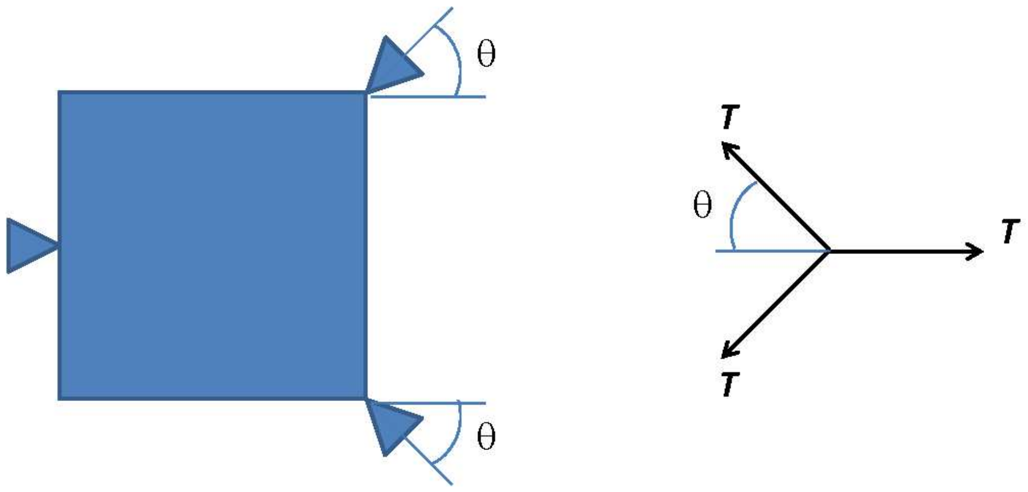


Figure B.6: Thruster Gimballing

The control acceleration, u_x, varies continuously with θ from $-T^*$ to T^* without relying on any variation in the control acceleration delivered from each thruster.

Naturally, this control actuation system can be extended with six more thrusters to provide the same performance in the $\hat{\boldsymbol{y}}$- and $\hat{\boldsymbol{z}}$-directions for a three-dimensional structure. Using gimballing in each axis, the control acceleration histories of Figure B.3 can easily be realized regardless of what the acceleration magnitude, T^*, is. With a secondary propulsion suite like this, a spacecraft attitude can be in any orientation and still implement a stabilizing control law.

Unfortunately, it may not be feasible to implement such a thruster configuration. Continuous control laws require continuous activation of the entire control suite, and fuel resources may deplete too rapidly. In addition, although this is a simple theoretical solution, the implementation of this may be extremely difficult. Slight misalignments would cause coupling with attitude dynamics, potentially introducing unwanted complications in the problem. Note that although T^* can be arbitrarily large, the larger T^* becomes, the more fuel is consumed, and the more precise θ must be to generate a particular control acceleration. Also, spacecraft design may have limitations on the number and placement of thrusters. The surfaces of the spacecraft represent prime real estate for payloads equally as necessary for mission operations.

It is potentially interesting to investigate this scheme further for implementation issues in libration point formations. However, due to the potential feasibility considerations listed above, the concept is abandoned in this investigation in pursuit of dynamic solutions that do not require the precision in control accelerations of an unconstrained control law.

B.3.3.2 Finite Burn Control Solutions

In the absence of thrust vector gimballing, the solution space is limited to control solutions that exhibit finite, constant burns in each thrust direction. When on, the thruster produces acceleration of magnitude T^*; when off, zero thrust. In addition, the vector of thrust (pointing direction) is necessarily constant in a spacecraft body-fixed reference frame, and the attitude of the spacecraft is a crucial element to any control solution.

An appealing advantage of finite burn control is that it is easily implemented on a real spacecraft. Regardless of how the solution is derived, a control history simply communicates *when* to fire each thruster, and the thruster only operates at its minimum thrust setting when firing.

A general trend in the literature leaves a separation between control theory and actuation implementation. This is to say that many control solutions presented in the literature, although quite elegant, can only be approximated with actual hardware. Hardware limitations always exist: thrust levels have minimum and maximum values, measurements can only be taken at finite intervals, impulses are impossible. In many cases, the implementation approximations are sufficient. However, the application driving this research is not considered to be one of them.

This effort presents a novel approach to optimal control, where implementation issues are considered as part of the solution process. Thus, the solutions presented in this work, although perhaps not as elegant, present the *optimal implementable* control solutions given an existing set of hardware.

Another aspect of these finite burn solutions to consider is that the solutions naturally do not exhibit the same quality of performance as theoretical solutions. For example, the Lyapunov-based controller of Section B.3.1 guaranteed asymptotic tracking of a reference trajectory. That is, the limits in tracking errors in both positions and velocities are zero. In contrast, a finite burn solution can make no such claim. Certainly, the control is constrained to such a great extent that it would be impossible to guarantee perfect tracking. From a theoretical perspective, this is a great disadvantage. However, from a mission perspective, finite burn solutions represent the *best possible performance with actual hardware*. Mission designers can use the solutions and solution methods presented in this work to determine performance capabilities and expectations for future missions.

Bibliography

[1] M.S. Branicky, V.S. Borkar, and S.K. Mitter, "A unified framework for hybrid control: Model and optimal control theory", *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, Jan 1998.

[2] Philip E. Gill, Walter Murray, and Michael A. Saunders, *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, Feb 12, 2006.

[3] H. S. Witsenhausen, "A class of hybrid-state continuous-time dynamic systems", *IEEE Transactions on Automatic Control*, vol. 11, pp. 161–167, Feb 1966.

[4] Hideyuki Takagi, "Introduction to fuzzy systems, neural networks, and genetic algorithms", *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*, pp. 405–468, 1991.

[5] Kai Chen, Ian C. Parmee, and Chris R. Gane, "A genetic algorithm for mixed-integer optimisation in power and water system design and control", *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*, pp. 311–330, 1997.

[6] M.S. Branicky, "Stability of switched and hybrid systems", in *Proceedings of the 33rd Conference on Decision and Control*, Lake Buena Vista, FL, Dec 1994, pp. 3498–3503.

[7] Stefan Pettersson and Bengt Lennartson, "Controller design of hybrid systems", *Hybrid and Real-Time Systems*, vol. 1201, pp. 240–254, Mar 1997.

[8] M.S. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems", *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, Apr 1998.

[9] E. R. Vrscay, “Iterated function systems: Theory, applications, and the inverse problem”, *Fractal Geometry and Analysis*, vol. 11, pp. 161–167, Feb 1966.

[10] F. Curti, L. Ascani, and M. Parisse, “Adaptive pulse width modulation”, *Advances in the Astronautical Sciences*, vol. 119, no. 1, pp. 775–788, 2005.

[11] Paolo Bolzern, Patrizio Colaneri, and José Claudio Geromel, “Optimal switching of 1-dof oscillating systems”, *Hybrid Systems: Computation and Control*, vol. 4416, pp. 118–130, Apr 2007.

[12] J.A. Ball, J. Chudoung, and M.V. Day, “Robust optimal switching control for nonlinear systems”, *SIAM Journal on Control and Optimization*, vol. 41, no. 3, pp. 900–931, 2003.

[13] Kagan Gokbayrak and Christos G. Cassandras, “Hybrid controllers for hierarchically decomposed systems”, *Hybrid Systems: Computation and Control*, vol. 1790, pp. 117–129, Mar 2000.

[14] Vadim Azhmyakov, Sid Ahmed Attia, Dmitry Gromov, and Jörg Raisch, “Necessary optimality conditions for a class of hybrid optimal control problems”, *Hybrid Systems: Computation and Control*, vol. 4416, pp. 637–640, Apr 2007.

[15] D. Coulter, “Nasa’s terrestrial planet finder missions”, in *Proceedings of SPIE*, Bellingham, WA, 2004, vol. 5487, pp. 1207–1215.

[16] O.P. et al. Lay, “Architecture trade study for the terrestrial planet finder interferometer”, in *Proceedings of SPIE*, Bellingham, WA, 2005, vol. 5905.

[17] W. Cash and K. Gendreau, “Maxim science and technology”, in *Proceedings of SPIE*, Bellingham, WA, 2004, vol. 5491, pp. 199–211.

[18] J. Mueller, "Thruster options for microspacecraft: a review and evaluation of existing hardware and emerging technologies", in *33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Seattle, WA, Jul 6-9, 1997, AIAA Paper 97-3058.

[19] D.A. Gonzales and R.P. Baker, "Microchip laser propulsion for small satellites", in *37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Salt Lake City, UT, Jul 8-11, 2001, AIAA Paper 2001-3789.

[20] D.A. Gonzales and R.P. Baker, "Micropropulsion using a nd:yag microchip laser", in *Proceedings of SPIE - The International Society for Optical Engineering*, 2002, vol. 4760, pp. 752–765.

[21] C. Phipps and J. Luke, "Diode laser-driven microthrusters–a new departure for micropropulsion", *AIAA Journal*, vol. 40, no. 2, pp. 310–318, 2002.

[22] C. Phipps, J. Luke, and W. Helgeson, "Laser space propulsion overview", in *Proceedings of SPIE - The International Society for Optical Engineering*, 2002, vol. 6606, p. 660602.

[23] David G. Hull, *Optimal Control Theory for Applications*, Springer-Verlag, New York, 2003.

[24] Arthur E. Bryson and Yu-Chi Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Taylor & Francis, New York, 1975.

[25] Donald E. Kirk, *Optimal Control Theory: An Introduction*, Prentice Hall, Englewood Cliffs, NJ, 1970.

[26] R.L. Haupt and S.E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons, Inc., New York, 1998.

[27] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, John Wiley & Sons, Inc., New York, 1998.

[28] Jon Lee, *A First Course in Combinatorial Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[29] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[30] L.C.W. Dixon, *Nonlinear Optimization*, The English Universities Press Ltd., London, 1972.

[31] A. Pourshaghaghy, F. Kowsary, and A. Behbahaninia, "Comparison of four different versions of the variable metric method for solving inverse heat conduction problems", *Heat and Mass Transfer*, vol. 43, no. 3, pp. 285–294, Jan 2007.

[32] Ladislav Lukšan and Emilio. Spedicato, "Variable metric methods for unconstrained optimization and nonlinear least squares", *Journal of Computational and Applied Mathematics*, vol. 124, pp. 61–95, 2000.

[33] David G. Hull, "Conversion of optimal control problems into parameter optimization problems", *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 57–60, Jan-Feb 1997.

[34] F. Fahroo and I.M. Ross, "A spectral patching method for direct trajectory optimization", *The Journal of the Astronautical Sciences*, vol. 48, no. 2-3, pp. 269–286, Apr-Sep 2000.

[35] F. Fahroo and I.M. Ross, "Costate estimation by a legendre pseudospectral method", *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 270–277, Mar-Apr 2001.

[36] J.R. Rea, "A legendre pseudospectral method for rapid optimization of launch vehicle trajectories", Master's thesis, Massachusetts Institute of Technology, 2001.

[37] S.A. Stanton, “Optimal orbital transfer using a legendre pseudospectral method”, Master’s thesis, Massachusetts Institute of Technology, 2003.

[38] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, McGraw-Hill, New York, 1997.

[39] R.W. Humble, G.N. Henry, and W.J. Larson, Eds., *Space Propulsion Analysis and Design*, McGraw-Hill, New York, 1995.

[40] S. Adler, A. Warshavsky, and A. Peretz, “Low-cost cold-gas reaction control system for sloshsat flevo small satellite”, *Journal of Spacecraft and Rockets*, vol. 42, no. 2, pp. 345–351, Mar-Apr 2005.

[41] W.C. Stone, “Fast variable-amplitude gold gas thruster”, *Journal of Spacecraft and Rockets*, vol. 32, no. 2, pp. 335–343, Mar-Apr 1995.

[42] K.C. Howell and B.G. Marchand, “Natural and non-natural spacecraft formations near the L_1 and L_2 libration points in the sun-earth/moon ephemeris system”, *Dynamical Systems: An International Journal, Special Issue: Dynamical Systems in Dynamical Astronomy and Space Mission Design*, vol. 20, no. 1, pp. 149–173, Mar 2005.

[43] K.C. Howell and T.M. Keeter, “Station-keeping strategies for libration point orbits: Target point and floquet mode approaches”, *Advances in the Astronautical Sciences*, vol. 89, no. 2, pp. 1377–1396, 1995.

[44] R.J. Luquette and R.M. Sanner, “A non-linear approach to spacecraft formation control in the vicinity of a collinear libration point”, in *AAS/AIAA Astrodynamics Conference*, Quebec, Canada, Jul 30 - Aug 2, 2001, AAS Paper 01-330.

[45] D.J. Scheeres and N.X. Vinh, “Dynamics and control of relative motion in an unstable orbit”, Aug 2000, AIAA Paper 2000-4235.

[46] D.J. Scheeres, F.-Y. Hsiao, and N.X. Vinh, "Stabilizing motion relative to an unstable orbit: Applications to spacecraft formation flight", *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 1, pp. 62–73, Jan-Feb 2003.

[47] P. Gurfil, M. Idan, and N.J. Kasdin, "Adaptive neural control of deep-space formation flying", in *American Control Conference*, Anchorage, AK, May 8-10, 2002, pp. 2842–2847.

[48] P. Gurfil, M. Idan, and N.J. Kasdin, "Adaptive neural control of deep-space formation flying", *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 3, pp. 491–501, May-Jun 2003.

[49] P. Gurfil and N.J. Kasdin, "Stability and control of spacecraft formation flying in trajectories of the restricted three-body problem", *Acta Astronautica*, vol. 54, pp. 433–453, 2004.

[50] B.G. Marchand and K.C. Howell, "Formation flight near L_1 and L_2 in the sun-earth/moon ephemeris system including solar radiation pressure", in *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Big Sky, MT, Aug 2003, AAS Paper 03-596.

[51] K.C. Howell and B.G. Marchand, "Design and control of formations near the libration points of the sun-earth/moon ephemeris system", in *Proceedings of the Space Flight Mechanics Symposium - Goddard Space Flight Center*, Greenbelt, MD, Oct 2003.

[52] B.G. Marchand and K.C. Howell, "Aspherical formations near the libration points in the sun-earth/moon ephemeris system", in *AAS/AIAA Space Flight Mechanics Meeting*, Maui, HI, Feb 7-12, 2004, AAS Paper 04-157.

[53] K.C. Howell and B.G. Marchand, "Formations near the libration points: Design strategies using natural and non-natural arcs", in *Proceedings of GSFC* 2^{nd} *International Symposium on Formation Flying Missions and Technologies*, Greenbelt, MD, Sep 2004.

[54] B.G. Marchand and K.C. Howell, "Control strategies for formation flight in the vicinity of the libration points", *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 6, pp. 1210–1219, Nov-Dec 2005.

[55] B.G. Marchand, *Spacecraft Formation Keeping Near the Libration Points of the Sun-Earth/Moon System*, PhD thesis, Purdue University, Aug 2004.

[56] B.G. Marchand, K.C. Howell, and J.T. Betts, "Discrete nonlinear optimal control of s/c formations near the L_1 and L_2 points of the sun-earth/moon system", in *AAS/AIAA Astrodynamics Specialists Conference*, Lake Tahoe, CA, Aug, 2005.

[57] S.I. Infeld, S.B. Josselyn, W. Murray, and I.M. Ross, "Design and control of libration point spacecraft formations", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 899–909, Jul-Aug 2007.

[58] K.C. Howell, "Three-dimensional, periodic, 'halo' orbits", *Celestial Mechanics*, vol. 32, pp. 53–71, 1984.

[59] Jaime Peraire, Anthony T. Patera, Jacob White, and Boo Cheong Khoo, "Numerical methods for partial differential equations", Spring 2002, Massachusetts Institute of Technology Course 16.920J/2.097J Notes.

[60] Huang Sunan, Tan Kok Kiong, and Lee Tong Heng, *Applied Predictive Control*, Springer-Verlag, London, 2002.

[61] "Space shuttle mission archives", http://www.nasa.gov/mission_pages/shuttle/ shuttlemissions/list_main.html.

[62] R.H. Vassar and R.B. Sherwood, "Formationkeeping for a pair of satellites in a circular orbit", *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 2, pp. 235–242, Mar-Apr 1985.

[63] D.C. Redding, N.J. Adams, and E.T. Kubiak, "Linear-quadratic stationkeeping for the sts orbiter", *Journal of Guidance, Control, and Dynamics*, vol. 12, no. 2, pp. 248–255, Mar-Apr 1989.

[64] V. Kapila, A.G. Sparks, J.M. Buffington, and Q. Yan, "Spacecraft formation flying: Dynamics and control", in *American Control Conference*, San Diego, CA, Jun 1999, pp. 4137–4141.

[65] Y. Ulybyshev, "Long-term formation keeping of satellite constellation using linear-quadratic controller", *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 1, pp. 109–115, Jan-Feb 1998.

[66] C. Sabol, R. Burns, and C.A. McLaughlin, "Satellite formation flying design and evolution", *Journal of Spacecraft and Rockets*, vol. 38, no. 2, pp. 270–278, Mar-Apr 2001.

[67] M.S. de Queiroz, V. Kapila, and Q. Yan, "Adaptive nonlinear control of multiple spacecraft formation flying", *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 3, pp. 385–390, May-Jun 2000.

[68] H. Schaub and K.T. Alfriend, "Impulsive feedback control to establish specific mean orbit elements of spacecraft formations", *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 739–745, Jul-Aug 2001.

[69] V.M. Guibout and D.J. Scheeres, "Spacecraft formation dynamics and design", *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 1, pp. 121–133, Jan-Feb 2006.

[70] E.M.C. Kong and D.W. Miller, "Optimal spacecraft reorientation for earth orbiting clusters: applications to techsat 21", *Acta Astronautica*, vol. 53, pp. 863–877, 2003.

[71] C.J. Scott and D.B. Spencer, "Optimal low-thrust reconfiguration for satellites in formation", *Advances in the Astronautical Sciences*, vol. 120, no. 1, pp. 3–21, 2005.

[72] C.J. Scott and D.B. Spencer, "Optimal reconfiguration of satellites in formation", *Journal of Spacecraft and Rockets*, vol. 44, no. 1, pp. 230–239, Jan-Feb 2007.

[73] M. Massari, R. Armellin, and A.E. Finzi, "Optimal trajectory generation and control for reconfiguration maneuvers of formation flying using low-thrust propulsion", *Advances in the Astronautical Sciences*, vol. 119, no. 3, pp. 2461–2474, 2004.

[74] B.T. Barden and K.C. Howell, "Formation flying in the vicinity of libration point orbits", *Advances in the Astronautical Sciences*, vol. 99, no. 2, pp. 969–988, 1998.

[75] R.S. Wilson and K.C. Howell, "Trajectory design in the sun-earth-moon system using lunar gravity assists", *Journal of Spacecraft and Rockets*, vol. 35, no. 2, pp. 191–198, Mar-Apr 1998.

[76] D. Folta, K. Hartman, K. Howell, and B. Marchand, "Formation control of the maxim L_2 libration orbit mission", in *AIAA/AAS Astrodynamics Specialist Conference*, Providence, RI, Aug, 2004.

[77] G. Gomez, M.W. Lo, and J.J. Masdemont, "Study on the station keeping maintenance for the tpf mission", *Advances in the Astronautical Sciences*, vol. 123, no. 3, pp. 2737–2750, 2005, AAS 05-42.

[78] K.C. Howell and H.J. Pernicka, "Numerical determination of lissajous trajectories in the restricted three-body problem", *Celestial Mechanics*, vol. 41, no. 1-4, pp. 107–124, 1988.

[79] Richardson D.L., "Analytic construction of periodic orbits about the collinear points", *Celestial Mechanics*, vol. 22, no. 3, pp. 241–253, Oct 1980.

Vita

Stuart A. Stanton was born in 1979 in Alamagordo, NM. Having spent some of his childhood in New Mexico, Massachusetts, and Indiana, he considers Colorado Springs, CO 'home.' Stuart attended the U. S. Air Force Academy, graduating in 2001 with a Bachelor of Science degree in Astronautical Engineering. Immediately upon commissioning, he was assigned to Cambridge, MA to pursue his Master of Science in Aeronautics and Astronautics from the Massachusetts Institute of Technology. Conducting research at the Charles Stark Draper Laboratory, he earned his degree in 2003. From 2003 to 2006, Stuart was assigned to the Space Superiority Materiel Wing at the Space and Missile Systems Center, Los Angeles Air Force Base, CA.

Stuart was selected by the Department of Astronautics at the Air Force Academy to pursue a Doctoral degree at the University of Texas at Austin, beginning studies in Guidance and Control in the Fall of 2006. As part of the Faculty Pipeline program, Stuart looks forward to joining the Academy faculty later in his Air Force career.

Stuart and his wife begin the next chapter of their lives in the Summer of 2009 with a new Air Force assignment located in the National Capital Region.

Permanent address: stuart.stanton@alum.mit.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.