Parametrized Maneuvers for Autonomous Vehicles

by

Christopher Walden Dever

B.S., Mechanical Engineering, University of Kentucky (1996)
M.S., Mechanical Engineering, M.I.T. (1998)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2004

Author .....
Department of Mechanical Engineering
August 8, 2004

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Marc McConley
Principal Member, Technical Staff, Draper Laboratory
Thesis Supervisor

Certified by .....
Samir Nayfeh
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Parametrized Maneuvers for Autonomous Vehicles

by

Christopher Walden Dever

Submitted to the Department of Mechanical Engineering
on August 8, 2004, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents a method for creating continuously parametrized maneuver classes for autonomous vehicles. These classes provide useful tools for motion planners, bundling sets of related vehicle motions based on a low-dimensional parameter vector that describes the fundamental high-level variations within the trajectory set. The method follows from a relaxation of nonlinear parametric programming necessary conditions that discards the objective function, leaving a simple coordinatized feasible space including all dynamically admissible vehicle motions. A trajectory interpolation algorithm uses projection and integration methods to create the classes, starting from arbitrary user-provided maneuver examples, including those obtained from standard nonlinear optimization or motion capture of human-piloted vehicle flights. The interpolation process, which can be employed for real-time trajectory generation, efficiently creates entire maneuver sets satisfying nonlinear equations of motion and nonlinear state and control constraints without resorting to iterative optimization.

Experimental application to a three degree-of-freedom rotorcraft testbed and the design of a stable feedforward control framework demonstrates the essential features of the method on actual hardware. Integration of the trajectory classes into an existing hybrid system motion planning framework illustrates the use of parametrized maneuvers for solving vehicle guidance problems. The earlier relaxation of strict optimality conditions makes possible the imposition of affine state transformation constraints, allowing maneuver sets to fit easily into a mixed integer-linear programming path planner. The combined scheme generalizes previous planning techniques based on fixed, invariant representations of vehicle equilibrium states and maneuver elements. The method therefore increases the richness of available guidance solutions while maintaining problem tractability associated with hierarchical system models. Application of the framework to one and two-dimensional path planning examples demonstrates its usefulness in practical autonomous vehicle guidance scenarios.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Marc McConley
Title: Principal Member, Technical Staff, Draper Laboratory

Acknowledgments

No thesis can be completed without the input and support of many people. The following individuals, and many more, have greatly enriched my experience in the doctoral program.

This research bears the mark of my committee members in particular. Professor Eric Feron has simultaneously been an enormous free-thinking creative influence and rational scientific evaluator behind the methods of this thesis. His constant inputs, from the original maneuver class formulation to his advocacy of using the method to enhance hybrid motion planning, have greatly improved this work. I wish to thank Marc McConley for his unwavering support throughout my entire program and for his keen insights into many of the finer technical aspects of the thesis. Marc's steady presence throughout my research program has always made me feel secure in my endeavors. Professor Jovan Popović helped me immensely, particularly during the early formulations of the thesis methods. His ideas and insights opened my eyes to new approaches for tackling technical problems, many of which directly influenced the final outcome. Professor Samir Nayfeh served not only as committee chair, counseling me on many programmatic matters, but also taking great interest in the work, quickly diving to the essence of the research and asking critical questions. Finally, my good friend and colleague Bernard Mettler has been an enormous contributor to virtually every aspect of this work. It's impossible to overstate how much I have benefited from his experience, from his high-level conceptual insights to his willingness to discuss and pour over numerous fine technical details. Many features of this thesis bear his mark, in particular the interface between maneuver classes and the MILP planning framework.

Many people at Draper Laboratory provided the professional and personal support that is essential for rewarding thesis work. I wish to express my heartfelt gratitude to Brent Appleby, George Schmidt, Loretta Mitrano, Milt Adams, Neil Adams, and Joe Sarcia for their contributions my program.

From the MIT Mechanical Engineering Department, Leslie Regan, Joan Kravit, and Professor Ain Sonin have been constant resources for both Institute and personal matters. They make the MIT academic world feel like home, an enormous comfort to one who often feels overwhelmed by it all.

My understanding of helicopter maneuvering was greatly enriched by discussions with Vlad Gavrilets, while the time and patience of Tom Schouwenaars, the true Lion of Flanders, allowed me to grasp the details of MILP-based motion planning. In addition, John Plump and Rodin Lyassof helped me understand the Draper and MIT helicopter simulations, respectively.

I wish to express my gratitude for the time and scientific thoughts of Leena Singh, John Hauser, Scott Rasmussen, Bruce Persson, Bill Hall, Sommer Gentry, Zhi-Hong Mao, and Air Force Reserve Captain Tyson Anderson, who all helped in the basic method formulation.

Many thanks go out to Masha Ishutkina and David Robertson for their help with the Quanser experimental apparatus and to Steve Hall and the MIT Aeronautics and Astronautics Department for the design of, and access to the machine as well.

Greg Mark graciously provided technical assistance during filming of the closed-loop helicopter flights.

Many people, including Julie Whatley, Angela Copyak, Lisa Gaumond, Lauren Clark, Yves Etheart, and Bryt Bradley, have helped with countless administrative matters and kept things fun and personal. My sincerest thanks go out to you all.

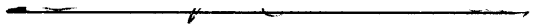
To my mechanical engineering qualifier study friends Cagri Savran, Ryan Jones, Alisa Morss, Ephi Most, Dan Mazzucco, and Dave Dussault, there's nothing I can say to fully express what a great pleasure it was to face the exams by your side. Thanks for the discipline, the motivation, and above all, the humor. You were there at the hardest of times.

Finally, to my family, you have been there through everything. My father Garland, sister Marie, and grandfather George are my true foundation. I look forward every single day to seeing you again. I dedicate this thesis to the memory of my mother, Karen.

8 August, 2004

This thesis was prepared at the Charles Stark Draper Laboratory, Inc., under Independent Research and Development Project Number 13177.

Publication of this thesis does not constitute approval by Draper Laboratory of the findings or conclusions contained therein. It is published for the exchange and stimulation of ideas.



[This page intentionally left blank].

Contents

List of Figures

List of Tables

Chapter 1

Trajectory Design Overview

Autonomous aerial vehicles often possess remarkable dynamic capabilities, with compact size and ample actuation combining to make for extremely agile systems. These vehicles, such as miniature rotorcraft and small-scale fixed-wing airplanes, possess enough control authority and structural rigidity to undergo massive accelerations, rapid direction changes, and even aerobatic flight. Such motions take advantage of physical nonlinearity and, from a modeling viewpoint, drive the system state well outside the range of steady and linear dynamics.

Human pilots operating agile systems at many scales, from military fighter pilots to expert radio-control hobbyists, learn to fly their vehicles through some combination of training, experimentation, and pure insight. They develop the ability to simply visualize and execute motions with their particular aircraft, almost crossing the line between man and machine. As such, experience creates a familiarity with vehicle flight dynamics, so that executing an acceleration or a direction change becomes a routine matter performed to order, not explicitly thought of in terms of control inputs and sensor outputs. The specific amount of acceleration or heading-change, for example, is simply a matter of degree; within a given type of maneuver, the control strategy is often the same, but with the exact state change coming through a subtlety of the control sticks and pedals.

While describing agile maneuvering for human-flown systems may be straightforward, with verbal expressions and body language conveying geometric ideas, mathematical nonlinearity makes the corresponding task for unmanned systems anything but easy. The vehicle becomes a dynamic model, the flight path becomes a trajectory signal, and the control inputs are the results of precisely designed algorithms. Much recent research has expanded the engineer's ability to describe agile maneuvers and other motions that enter the realm of nonlinear dynamics. Specifically, numerical methods now exist to find system trajectories meeting exact boundary conditions for many system types and there are even examples of developing advanced aerobatic controllers by decomposing recorded pilot-machine interactions. These efforts illustrate the enormous promise of directly addressing system nonlinear dynamics and finding powerful control designs.

In this spirit, a needed innovation is the ability to bundle sets of related trajectories into intuitive maneuver classes. Just as a helicopter pilot might think of the set of

quick-stops as a known maneuver having a known execution strategy with the amount of cyclic input and pitch flare dictated by the initial speed, autonomous vehicles can benefit by an analogous engineering understanding of motion control strategies. The ability to cast maneuvers as a continuously-varying family of behaviors can greatly aid many on-board systems. Reference trajectories can then be generated as single instances of a given maneuver type, avoiding the need to "solve" the motion problem from scratch using a large number of trajectory parameters. Similarly, motion planners will be able to make guidance decisions considering the types and ranges of possible maneuvers, as opposed to selecting fixed, static motion elements from a given library.

This thesis addresses the problem of designing parametrized maneuver classes for autonomous vehicles described by nonlinear equations of motion. The flavor of analysis is much as a human pilot might follow: testing out a few maneuvers within a given type and thus learning how to perform similar motions in future settings. The goal is a simple real-time algorithm that allows on-board systems to "think" in terms of general maneuvers and motions, with the execution details coming as a fairly simple by-product.

This chapter introduces this problem in engineering terms, defining what is meant by a maneuver and by trajectory generation. The specific objectives are given and the solution approach outlined. An extensive literature review describes the traditional and state-of-the-art methods for computing useful motions of nonlinear systems. Investigation of related fields, including robotics, computer animation, and biomechanics, reveals that motion description and manipulation are ever-present tasks, with many insights to be learned from ongoing research. A review of hybrid motion planning techniques them makes clear the need for and benefits of maneuver-level trajectory descriptions. The final sections list the specific thesis contributions and outline the overall document structure.

1.1 Flexible Maneuvers for Autonomous Vehicles

Fortunately, there are many existing tools for designing motions of nonlinear systems. The traditional methods of optimal control and numerical optimization provide the right language to cast the flexible maneuvering problem and offer a point-of-departure for the techniques of this thesis. This section describes what is meant by "trajectory generation" and how it interacts with higher-level motion planning and lower-level control. Specific thesis objectives and a sketch of the solution method then follow. A review of relevant source material from aerospace engineering and general motion analysis appears in subsequent sections.

1.1.1 Trajectory Generation

By their very nature, agile motions require a system to undertake rapid and large state changes, possibly using large-magnitude control inputs. As such, simple output error-driven feedback control loops are insufficient to guarantee repeatable and crisp

maneuver execution, allowing too much time delay in the system response. A more effective approach is a feedforward control strategy (sometimes referred to as a "two degree-of-freedom" design [102, 149, 153]), which designs an output-space reference trajectory for tracking and simultaneously applies a *dynamically consistent* control input to drive the system. Feedback control may then be used for stabilization purposes, providing small perturbational control inputs to regulate small output errors while performing the agile motion. Such output-space errors naturally arise from inaccuracies in the nonlinear model and external disturbances pushing the vehicle away from its desired path.

In this thesis, the expression "trajectory generation" refers to the process of finding a consistent set of control, state, and output signals that produce a motion satisfying given boundary conditions. Specifically, the term "maneuver" will denote a trajectory where the vehicle both begins and ends the motion at an equilibrium, or trim, state. For example, it might be desired to perform a rapid direction reversal maneuver, with initial and final velocity vectors given as boundary conditions. The trajectory generator must then provide the appropriate control and state trajectory histories for the maneuver.

As will be discussed in Section 1.2 and Chapter 2, the trajectory generation task for general systems is nontrivial, but there are many existing methods for casting precise problem formulations with known algorithmic solutions. When dealing with general nonlinear systems of the form $\dot{x} = f(x, u)$, (x is the state vector; u is the control input), the mathematics must necessarily be in terms of nonlinear relations, making closed form solutions impossible and thus requiring iterative numerical methods.

Perhaps the most well-established solution method is nonlinear programming (NLP), appearing in virtually every venue of trajectory generation, from traditional aerospace and astrophysics problems to modern agile maneuvering applications. The basic method involves parametrizing a general continuous time system behavior $\mathbf{z}(t)$, (depending on $x(t)$, $u(t)$, and/or some system output $y(t)$) in terms of a finite parameter vector p, thus obtaining a lower-dimensional representation $\mathbf{z}(t; p)$. The vector p is then optimized over some set of feasible candidate motions. Equality and inequality constraint functions, $h(p)$ and $g(p)$, respectively, enforce dynamic consistency and initial and final boundary conditions while giving admissible ranges of state and control variations. An objective function $f(p)$ helps select a particular behavior $\mathbf{z}(t; p)$ out of all candidate feasible motions, typically seeking to minimize some duration, control, or energy metric. The result is a generic nonlinear program of the form

$$\begin{array}{ll} & \min_{p} f(p) \\ \text{subject to} & \begin{array}{l} h(p) = 0 \\ g(p) \leq 0 \end{array} \end{array},$$

whose minimizing feasible point p^* solves the trajectory generation problem.

While nonlinear programming is an extremely effective and general method of finding optimal motions, it has several liabilities when considered for real-time trajectory generation of agile maneuvers. First, it is an iterative numerical method,

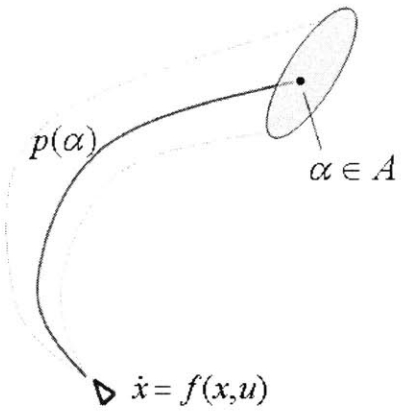


Figure 1-1: An autonomous vehicle with a flexible maneuver family. The low-dimension vector α indicates the particular maneuver instance required while $p(\alpha)$ gives to corresponding open-loop trajectory details.

with no guarantee of convergence or even attainment of a feasible solution in finite time. Second, it is only possible to find locally minimizing solutions, which are therefore extremely sensitive to initial parameter guesses $\hat{p}_0$. The choice of $\hat{p}_0$ in a given problem instance critically influences the final solution, so that poorly considered $\hat{p}_0$ guesses can adversely affect the trajectory generation outcome. Third, the trajectory parametrization p is often far too general when considering vehicle motions at the maneuver level; the large dimension of p adds extra complexity to the iterative solution process and robs the procedure of a pilot-like feel for subtle variations in maneuvering characteristics.

In addition, nonlinear programming is by definition driven by an objective function, seeking optimal solutions while often producing infeasible iterates during the solution process. As such, there is no straightforward way to easily manipulate useful feasible motions in the spirit of a human pilot tweaking and bending maneuvers to achieve some goal. Other than the initial solution estimate $\hat{p}_0$, there is no apparent way to directly introduce human pilot knowledge of vehicle dynamics into the trajectory generation process. Example motions flown on real hardware by expert operators provide enormous insight into useful vehicle operating modes while directly demonstrating the corresponding feasible nonlinear control inputs. Since such example motions typically do not minimize a mathematical function, their inclusion as a simple $\hat{p}_0$ suggestion vector in NLP can seem somewhat inconsistent with a given objective function $f(p)$.

1.1.2 Objectives

Based on the above limitations of nonlinear programming as a real-time trajectory generator, this thesis seeks a method that retains the rigor and basic formulation of NLP while allowing greater flexibility, faster solution, and the ability to incorporate

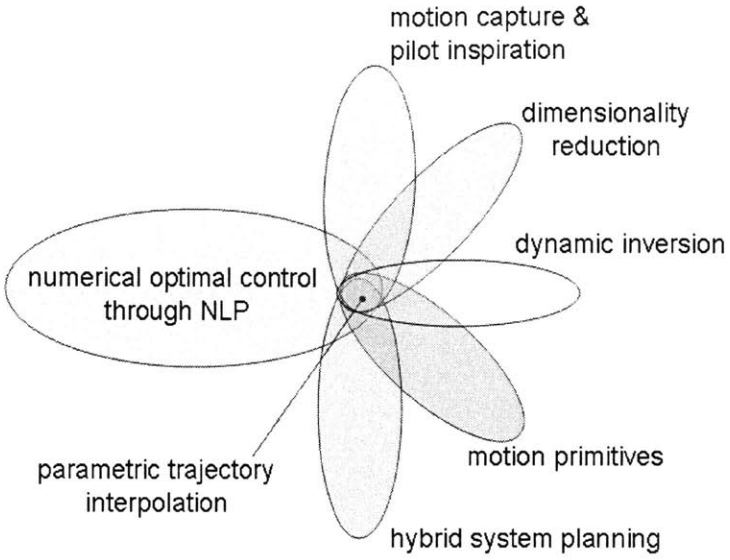


Figure 1-2: The main concepts behind parametrized maneuver classes.

known feasible motions in the design of user-selected maneuver classes. Define the vector α as a reduced-dimension descriptor of a specific maneuver type, listing *only* the fundamental conceptual or boundary condition variations in vehicle motion. For example, if the maneuver is a simultaneous climb and direction change, α would give only the net elevation gain and heading angle modification. If the maneuver is a dash-acceleration from hover, then α is only a scalar final velocity value. In terms of the "trajectory parametrization" p and the so-called "maneuver parametrization" α, the specific thesis objectives are as follows:

- Create parametrized maneuver trajectory generators of the form $p(\alpha)$. Given a specific admissible maneuver instance $\alpha \in A$, return the general trajectory vector p and thus $\mathbf{z}(t;p)$, providing the complete control, state, and output behavior of the vehicle. The set A bounds the allowable variations of α within a particular maneuver class. See Figure 1-1.
- Allow incorporation of any feasible example motions in the creation of the maneuver functions $p(\alpha)$. These example maneuvers must be able to include solutions derived from off-line nonlinear programming as well as observations of pilot-flown trajectories.
- Make trajectory generation a real-time process.
- Demonstrate the use of parametrized maneuvers in an existing hierarchical motion planner.

1.1.3 Approach

Figure 1-2 illustrates the various mathematical concepts necessary to attain the goals of Section 1.1.2. Since NLP is an established, rigorous method of nonlinear system trajectory generation, the thesis essentially develops several useful adaptations and relaxations to obtain a highly flexible real-time algorithm. The resulting method

allows general manipulation of feasible system trajectories and solves for new motions without any iterative optimization.

To begin, general NLP's are recast as nonlinear *parametric* programs, in which the objective and constraint functions may explicitly include the lower-dimensional vector α, indicating what specific maneuver is desired within a class. Parametric programming methods begin with a single known optimal solution p^* for a given α value and then systematically trace out an optimal solution family $p^*(\alpha)$ using continuation methods and tests for solution type and constraint switching points.

To create a trajectory generator capable of producing *any* general feasible motions, the Karush-Kuhn-Tucker optimality conditions that underlie parametric programming are relaxed, with knowledge of multiple user-selected trajectories replacing the objective function as a means for defining maneuver classes. A dynamic interpolation algorithm, based on continuation methods combined with a feasible projection operation, then allows access to a parametrized family $p(\alpha)$, essentially creating an entire maneuver class from a finite number of example motions.

As chosen in this thesis, the trajectory variable p describes the vehicle output space behavior, with control inputs obtained by dynamic model inversion. If the system is difficult to invert from a small number of outputs, p is expanded to include enough system states necessary to solve for controls, with suitable constraint functions added to maintain dynamic consistency.

Because the trajectory interpolation algorithm works between any suitable feasible points, example motions may come from rigorous off-line nonlinear programming solutions or motion capture of pilot-flown maneuvers. Thus, human knowledge and intuition of vehicle dynamics and useful trajectory shapes may be directly accommodated when defining a maneuver class.

Figure 1-3 illustrates how the parametrized maneuver families and trajectory interpolation can be used in practice. Off-line, the user models the system, chooses a trajectory descriptor p, defines a conceptual maneuver type, and selects a low-dimension maneuver parameter α that describes the desired motion variations. Then, example maneuvers may be generated by nonlinear optimization or piloted-flight motion capture. Another option for creating example maneuvers is to simply "sketch" a motion by manually selecting some $\hat{p}_0$ and then using NLP to find the closest feasible motion to $\hat{p}_0$. With the examples in place, the interpolation algorithm creates the parametrized family $p(\alpha)$. On-line, the process works as a practical trajectory generator, with a higher-level motion planning algorithm requesting a particular maneuver within the class based on overall vehicle guidance needs.

To demonstrate the usefulness of parametrized maneuvers for higher-level hybrid system motion planning, this thesis shows that $p(\alpha)$ sets may be easily integrated as "motion primitives" into a mixed integer-linear programming (MILP) path planner. When combined with the ability of MILP methods to command continuously varying state trajectories for effective linear closed-loop system models, flexible maneuvering creates an extremely general guidance framework. There is no need to restrict the flight envelope to discrete sets of trims and/or maneuvers as required in some existing planners.

Finally, application to a three degree-of-freedom helicopter allows experimental

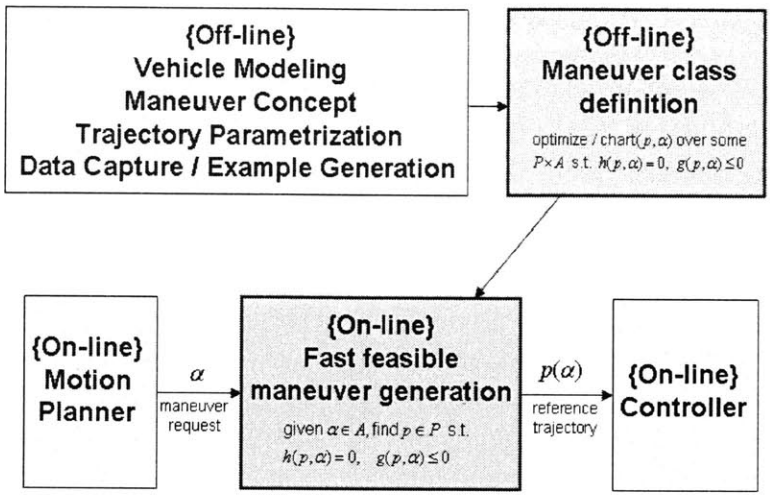


Figure 1-3: Flexible maneuvering classes result from detailed off-line system analysis, example trajectory selection, and family creation. On-line, a motion planner requests a particular maneuver within the family and the appropriate reference trajectory is passed to the control algorithm.

validation and demonstration of all thesis concepts. Emulating the longitudinal dynamics of a full-scale helicopter, the rotorcraft system provides a challenging nonlinear model, while allowing both controller-driven and human-piloted flight with signal data acquisition.

1.2 Existing Methods

This section gives a literature review of techniques useful for trajectory generation, motion design, and control of nonlinear systems. The greatest emphasis is placed on the traditional and emerging methods of aerospace engineering, especially those geared toward real-time trajectory design for autonomous vehicles. Additionally, there are examples of using human-inspired state machines to control aerobatic helicopter flight. Interestingly, other fields such as robotics, computer animation, and biomechanics also deal with complex nonlinear systems and place value in capturing, and then manipulating, demonstrated motions to achieve new effects. Frequently, motion elements are broken into flexible building blocks called "motion primitives". A review of selected works from these disciplines gives insights into ways of manipulating motions of complex systems without redesigning trajectories from scratch. These approaches often achieve a reduced parametrization of the trajectory space; a discussion of two interesting mathematical approaches to dimensionality reduction therefore provides additional insight. Finally, this section gives examples of emerging linear and quadratic *parametric* programming methods in control systems, inspired by goals very much akin to those of this thesis.

1.2.1 Optimal Control in Aerospace

The historic and current practices of aerospace engineering provide most of the guideposts for the methods of this thesis. The flexible maneuvering approach sits as the beneficiary of a variety of tools for dealing with nonlinear systems. Examples include modeling and system identification, constraint specification, separation of the guidance and control functions, discretization of continuous time problems into finite dimensions, and hierarchical systems theory. Combined, these methods coalesce into a clear picture of how to pose the general trajectory generation problem.

The main foundations of trajectory design lie in optimal control theory, where open and closed-loop system behaviors are derived from general nonlinear equations of motion, studied from the viewpoint of the calculus of variations and Pontryagin's principle [25, 26, 116]. A common outcome of an optimal control analysis is an exact statement of the necessary and sufficient conditions for a trajectory and control to optimize an engineering objective function.

In some cases the conditions admit analytic solutions, but for many problems the solution must come from an iterative numerical procedure. A useful approach is to cast the continuous-time optimal control problem in a finite-dimensional nonlinear programming setting [14]. The relationships between optimal control and nonlinear programming (NLP) run very deep, as noted in the close duality between the infinite and finite-dimensional necessary conditions. Fortunately, there are numerous methods for casting an optimal control problem in NLP format, with interesting variations including direct, indirect, transcription (collocation), and shooting methods. References such as [16] and [17] give detailed discussions of these different problem types, with indications of what method to choose based on the engineering objective and available problem information. Additional sources give supporting material on casting continuous time problems into finite-dimensional optimization frameworks [18, 63, 68]. Interestingly, reference [136] discusses the technique of differential inclusion for abstracting system dynamics, so that NLP may be used to design feasible motions without computing detailed state and control histories, a viewpoint similar to many current hierarchical motion planners.

Focusing on autonomous vehicle applications, many trajectory optimizers rely on the availability of control schemes capable of tracking given reference trajectories, in particular those deviating widely from the linear operating range. A common approach is a feedforward tracking design, in which reference control and state histories are computed based on open-loop equations of motion and then fed to a linear or nonlinear tracking controller. These methods are particularly relevant for aggressive or agile maneuvering, where error-driven controllers are likely to perform poorly. Several references discuss in detail the feedforward control problem for interesting aerospace systems. Examples include [153], which designs linear robust controllers for a simulated Westland Lynx combat helicopter. Reference [137] expands on this approach by investigating the merits of several control design methodologies for an autonomous helicopter. These methods range from a linear robust design to a fuzzy logic controller to a nonlinear tracking formulation based on feedback linearization of an approximate system model. This last approach is found to be particularly effective

when linear models simply cannot provide enough predictive fidelity for agile motions. Reference [83] takes a similar approach, designing a bounded-error output-tracking helicopter controller based on a state-input linearizable approximate vehicle model. Other formative examples include [72], which discusses a finite horizon control Lyapunov function method for tracking reference trajectories obtained from a simplified system model and [149], which considers the trajectory generation and control problem for the class of so-called differentially flat systems. These systems are extremely useful, in that motions can be designed in the output-space, with corresponding control inputs obtained by fairly simple analytic model inversion expressions.

The notion of an invertible nonlinear system is not limited to flat models, and there are earlier works considering the approach for control of large-angle aircraft maneuvers. Reference [78] examines airplane maneuvers through a detailed inversion of the nonlinear kinematic and dynamic equations of motion. Reference [87] treats a similar problem from a traditional optimal control approach, deriving control laws based on detailed optimality conditions. The system inversion philosophy is certainly appropriate to rapid trajectory design, when the prospect of forward integration and iterative perturbation of candidate control signals is too daunting for real-time applications.

Returning to flat and near-flat systems, the model inversion technique is particularly relevant for autonomous vehicle trajectory design, especially when given the availability of stable closed-loop tracking methodologies that compensate for modeling errors or approximations. References [48, 105, 106, 149, 150, 151] collectively give a detailed treatment of the theory and practice of differentially flat robotic and aerospace systems. Forming a critical example for the methods of this thesis, these works place considerable emphasis on the numerical trajectory generation problem, often using spline-based output signal parametrizations for real-time and near real-time nonlinear programming. Works giving specific examples of these methods include [113], which considers a multi-state mathematical example system; [30], which looks at a planar, wireless fan-driven air-bed vehicle; [101] and [102], which treats a three degree-of-freedom ducted fan; [42] which designs motions for a V-22 Osprey model; and [152], which treats an AV-8B Harrier aircraft. Note that these spline-based methods form a dynamics-conscious generalization of polynomials as descriptors of helicopter maneuvers [143].

There are other related works, each with a slightly different approach to the trajectory generation process, but still well-suited for feedforward control schemes. Examples include [64], which uses a homotopy-based "morphing" of system models in place of flat characterizations to obtain reference trajectories for complex nonlinear systems. Another is [80], which uses feedback linearization to transform a nonlinear longitudinal helicopter model to linear form, applies the aforementioned differential inclusion-based optimization to generate reference trajectories, and finally transforms them back to the original nonlinear system. Reference [27] is an example of actually parametrizing an input signal for NLP-based optimization, finding useful forward maps between inputs and outputs for a class of polynomial nonlinear systems and then applying the method to a vertical-takeoff-and-landing system model. Collectively, all of these inversion and model-specific optimization methods provide numer-

ous options for trajectory generation, especially when there is an available controller and sufficient time to compute solutions.

However, they do not emphasize using known system behaviors to speed up solution time or directly design useful trajectory characteristics. It is reasonable to expect that having computed one solution or obtained an example feasible trajectory, it would be considerably easier to find related motions without having to resort to new optimizations from scratch. Fortunately, this notion is well-established in certain classical optimal control settings and is beginning to find acceptance for modern vehicle guidance algorithms.

The idea of using known optimal trajectories to find solutions for closely related problems is the driver behind neighboring extremal control [26, 71]. In this setting, first-order optimality conditions are differentially perturbed into second-order conditions and used to find nearby optimal solutions, often employing some sort of feedback on perturbational states. Other examples include [7], which uses known solutions for a given initial condition to speed up dynamic programming for neighboring starting conditions.

Another interesting approach is a homotopy of models, as seen in [65] and [72], where trajectories may be obtained for complex dynamic systems by first designing similar motions for comparatively simpler models and then transforming the feasible behavior back to the original system. A related approach is that of [73], which examines the trade-off between solution time and optimality in a receding horizon control setting. Here, strict optimality is not enforced in the early phase, as an aid to finding a reasonable starting feasible solution. Subsequent iterations can then progressively improve the solution performance, gradually getting closer to optimality.

Finally, forming an important example for this thesis, is the practice of directly taking specific pilot-flown trajectories and studying them for autonomous execution under closed-loop control. References [55] and [114] give a discussion of inferring human control strategies for classes of aerobatic helicopter maneuvers and encoding them as finite state machines. This process avoids the daunting task of devising aggressive trajectory design and control from scratch, and instead observes "real-world" instances to determine the appropriate control strategy. Other references follow up on the promise of this method by rigorously modeling and testing system flight control geared for helicopter aerobatics [57, 58] and then demonstrating the method in actual flight [54, 56].

1.2.2 Robotics

Similar to the aerospace examples just outlined, research into robotic systems often deals with planning and optimizing the motions of highly complex nonlinear systems. There are several works worth mentioning that discuss solutions to problems involving nonlinear dynamics, motion basis sets, and system constraints. Notably, several resort to the methods of model inversion and the idea of perturbing known feasible motions or learning-by-example to obtain new motions.

In some situations, a robot end-effector geometric trajectory has been previously computed and the challenge is to find a suitable path time-parametrization that does

not violate state and control bounds. In [67], the system constraints arise from robot lower and upper joint torque limits. Because of nonlinearity, motions that are either too fast or too slow can violate the input constraints, and the problem cannot be solved by simply tweaking the time scaling in one direction until feasibility occurs. An efficient algorithm results by considering the system inverse dynamics and devising a method for torque recomputation exploiting known candidate path scalings without having to successively re-invert the equations for motion from scratch at each iteration. Reference [12] works in a similar spirit with a goal of finding a minimum-time traversal to a terminal configuration subject to state and input voltage constraints. Designed for on-line implementation, a receding horizon control algorithm performs a scalar optimization over fixed look-ahead intervals. The simplicity of the problem formulation allows numerous solution properties to be considered, including the provable ability to find a safe path parametrization in real time.

In other situations, a path is not given and the task is to plan a motion in the presence of geometric obstacles. Similar to some current aerospace research, reference [38] considers a point-mass system operating with velocity and acceleration bounds and searches for a feasible path around obstacles subject to certain safety criteria. The presented solution exploits a grid-like network description of the motion space, using a directed-graph algorithm that allows an explicit trade-off between planning time and path optimality.

In some situations, robotic systems are sufficiently complex to warrant use of machine learning-type methods to construct new motions. Control of a bipedal humanoid walking robot in [32] is aided by direct use of human motion capture data. The work employs a nonlinear inverse dynamics representation and the notion of a zero moment point to design and control periodic walking patterns of a four degree-of-freedom robot. Reference [129] further explores the merits of robot learning from human example in the context of an anthropomorphic robot performing pole balancing and inverted pendulum swing-up tasks. Noting that humans rarely learn control strategies without some sort of known example, the work uses physical data to "prime" the solutions of linear (Q-learning) and nonlinear (V-learning) reinforcement learning algorithms. Exploring alternatives to basis functions and spline-based signal representations, references [70] and [130] present sets of parametrized nonlinear systems for capturing and modifying demonstrated motions. Inspired by discrete and rhythmic components of human arm movements, the system building blocks perform motion capture with a recursive least squares matching procedure and then allow manipulation of output space behaviors and the corresponding controls.

1.2.3 Computer Animation

Curiously, the field of computer animation shares many commonalities with mainstream aerospace and robotics practice. Historically, animation required the rather laborious task of keyframing, where snapshots of motion were rendered individually and then brought into continuous time through operations such as spline interpolation. Not only did this process often result in physically unsatisfying motions, but it required manual edits of many keyframes to make even simple alterations to the

overall movement sequence.

While developing more sophisticated techniques to automate this tedious process and free up the animator for more creative endeavors, computer science researchers typically find themselves working with many of the staples of traditional engineering analysis. These include signal parametrization, nonlinear system kinematics and dynamics, and constrained optimization. Differences occur, however, in the required level of modeling detail, as animated motions need only be perceptually plausible, not necessarily exactly feasible dynamically.

This subsection surveys relevant animation research in three categories. The first is perception-based motion editing, where equations of motion are not considered and the focus is on manipulating purely kinematic variables. Here the emphasis is on working with existing motions, especially those obtained from motion capture. The second set of methods are physics-based animations, where equations of motion are explicitly included, but where movements are designed from scratch, without knowledge of existing trajectories. Finally, physics-based motion capture methods combine the ability to edit demonstrated motions while employing the rigors of physical modeling and dynamic feasibility. These methods are particularly relevant as examples for designing flexible maneuvering schemes for autonomous vehicles.

Perception-based motion editing

Not surprisingly, the animation methods that tend to bypass nonlinear equations of motion and work directly with output space kinematics often treat an exceedingly complex dynamic system: the human body. Still, working with torso and limb motion geometry often entails complicated nonlinear relations and the need for motion capture to provide reasonable starting points is clear. In an interesting parallel to motion planning as discussed in Section 1.3, these methods often treat the movement space hierarchically, with flexible motion elements providing the basic tools for dynamic or combinatorial planners.

A typical example of such methods is reference [6], which takes existing human motion sequences, determines points at which they may be spliced together, and then develops a directed-graph representation based on these switching points. To generate new animations, a combinatorial method using randomized searches recombines portions of the original motion sequences and applies a smoothing operation to reduce any perceived discontinuities. Reference [84] presents a similar method for animation synthesis based on graph representations and a thresholding mechanism for determining likely switching points between existing data sequences. The user can enrich the motion database by providing examples with different character displacements and activities.

Other works include [24], which applies multiresolution filtering and other signal processing tools to blend styles of existing motions using mathematical operations such as time warping and frequency domain filtering; references [59] and [155] take data from various human motions (jumps, walks, slides, etc.) and then allow the user to specify desired keyframe edits to the figures, forming a nonlinear problem whose solution bends to original sequences to satisfy the edits.

A different approach is taken in [89], where motion "textons", linear statistical models of individual human dance segments, form the basic elements for a state transition matrix representation of motion planning. The matrix elements correspond to likelihoods of switchings between the individual textons, allowing motion synthesis from highly varying elements. Finally, reference [86] uses a hierarchical B-spline representation coupled with inverse kinematics to synthesize new motions based on user-imposed constraints, providing tools for editing an existing data set. Applications include selection of new geometric paths, reassignment of sampled motions to new characters, character geometric distortion, and movement style transitions, similar to those seen above in [84].

Physics-based motion generation

In some settings, it is possible to use physics-based modeling to improve the perceived motion realism and allow the animator to design sequences using more traditional constraints and objective functions. A representative example of this method is reference [154], which models the dynamics of a joint torque-driven animated lamp character and applies nonlinear optimization to find the minimum power input signal satisfying a number of user-chosen equality constraints. A related method appears in [23], where physics-based methods provide a dynamically feasible keyframe interpolator. This technique allows the animator to optimize a simulated linear state-space model in a series of terminal constraint problems (thus avoiding a difficult multipoint boundary value problem), with a goal of minimizing control effort and a perceptual "nonsmoothness" index.

Reference [85] allows a direct user handle of animations through proportional-derivative controlled physical models of articulated lamps, cats, and humanoid characters. State machines then provide a method to combine these user-produced motion primitives into longer compound or repetitive motions. Finally, reference [81] uses an inverse dynamic model of a 97 degree-of-freedom humanoid figure to provide direct balance and comfort control while creating dynamically sound walking motions with variations in gait, motion path, and figure size.

Physics-based motion editing

There are also methods which combine physics-based modeling with motion capture, providing a way for users to directly influence nonlinear equations of motion and infuse trajectory manipulation schemes with quantitative knowledge of the animation objectives. These approaches most closely resemble the methods of this thesis, where demonstrated vehicle trajectories provide guideposts for parametric descriptions of feasible maneuvers.

Reference [121] reexamines the problem of humanoid animation, taking motion capture data parametrized to a highly detailed system model and then mapping these samples to a feasible point of a simplified lower-dimensional model. Application of spacetime constraints then allows the animator to edit and distort motion to produce new effects. Finally, a constrained optimization problem re-maps the edited

motion back to the full-order system, using knowledge of muscle force-articulated body dynamics to create realistic animations of human athletic activities.

In a similar spirit, reference [120] gives a method for manipulating physical simulations of moving and colliding objects. A forward simulation is parametrized in terms of initial conditions, elasticity coefficients, and surface normals. The animator may then directly edit the simulation output behavior, with a nonlinear optimization routine computing the corresponding perturbations to the simulation physical parameters. The method is useful for creating animations with seemingly unlikely effects, such as objects colliding in mid-air and then falling into perfectly placed containers. This approach continues in [119] where initial trajectory guesses may come from human-demonstrated examples. A multiple shooting methodology allows interactive editing of motion capture examples for tumbling and colliding objects. The demonstrations provide the spirit of the motion while the interactive manipulation can be used to enforce the exact animation effects, such as objects falling into perfect arrangement after collisions or objects tumbling in seemingly impossible ways.

Finally, there are several examples of physical animation that closely resemble the aerobatic maneuvering control methodologies discussed at the end of section 1.2.1. Large change of pose animations of humanoid characters in reference [43] follow from stable feedback control algorithms with state-machine governed switching points, similar to the pilot-inspired logic of [54, 55, 56, 57, 114].

Similar efforts appear in reference [66], where velocity error feedback drives human characters during athletic activities using rules derived from real-world strategies for multiphase or periodic motions, including gymnastics, running, and cycling. The rule-based control schemes apply to individual actors as well as larger collision-free group behaviors (for example a cycling peloton navigating a curve or road obstacle) and could find application to guidance of swarms of autonomous vehicles. In a highly complex demonstration of such methods, reference [157] applies feedback logic to muscular models of physics-based bird flight animations that include body articulations and detailed flapping wing aerodynamic models. The animator may specify a particular flight path and flight style with the animation algorithm determining the numerous wing-beat parameters necessary to produce the motion.

1.2.4 Biological Motion Analysis

The field of biomechanics is another area where analysis of collected motion data is important, with emphasis placed on concise ways of describing and explaining motor control. Here the term "motion primitive" is often used to describe a flexible, fundamental building block from which general movements, particularly those of human beings, are constructed. Although there is no universally agreed upon definition of a motion primitive, a common thread is that such basic elements are not just convenient ways of describing motion, but actually capture fundamental physiological mechanisms. The references in this section confirm that motion analysis is a common problem across many branches of applied science and that the hierarchical systems used to command and actuate biological systems share much in common with the man-made path planning systems to be discussed in Section 1.3.1. From this view-

point, the flexible maneuvering elements that are the subject of this thesis are similar to the flexible motion primitive methods discussed here: each attempts to compactly represent dynamic system behavior for use by a higher-level planner.

Examples of work in biological motion analysis include [4], [5], and [74], each of which treats anthropomorphic arm and joint movements, performing dimensionality reduction on motion capture data and decomposing observed behaviors into multi-layer primitives. The lower levels describe pure physical action while the higher levels attempt to capture the behavioral algorithms for organizing the lower levels into useful tasks. Each of the references takes a slightly different approach, with the first emphasizing self-ordered maps to discover the relationship between the lower and higher levels, the second applying probabilistic hidden Markov models, and the third using spatio-temporal decomposition. Each of these works extends the motion analysis problem into a control problem by synthesizing new motions from the derived primitives and demonstrating their techniques on human mock-up systems.

A closely related method is that of [49], which uses a classic dimensionality reduction technique, principal components analysis, to form linear basis sets describing arm motions. The linear decomposition is useful for classifying new data sets based on sample clustering methods.

Working with human-produced two-dimensional reaching and drawing motions, reference [36] defines "movemes" as a motion primitive analogy to phonemes in spoken language. Mathematically, a moveme is the output of a parameterized linear system designed and trained to produce a certain type of motion, such as reaching or pointing. These systems are shown to possess independence and segmentability properties, making them useful for both movement analysis and synthesis problems.

Paralleling the notions of invertible and flat systems as discussed in Section 1.2.1, reference [140] applies inverse kinematics to test two interesting hypotheses concerning human arm control during drawing tasks. A counterexample to a control segmentability theory is presented using an anthropomorphic robot arm acting under continuous control. In the end, oscillatory pattern generators similar to those of reference [70] in Section 1.2.2 are suggested as a natural way of producing and controlling rhythmic motions.

Lastly, [107] applies the inverse dynamics approach as a tool for uncovering basic neurological motor commands and experimentally studies frog and rat limb movements. It is shown that central nervous system interactions with muscle actuators can be conveniently defined in terms of force-field motion primitives. Interestingly, the reference discusses several of the leading theories on muscular limb control, in particular the distinction between feedforward and feedback control, especially when visual feedback is present. Just as with aerospace control of agile vehicle maneuvers, a combination of feedforward and feedback seems an essential way of specifying and controlling motion in the real-world.

1.2.5 Dimensionality Reduction

The motion primitive and motion building block methods seen above in biological and animated motion analysis perform a useful function in reducing planning complexity.

The problem of synthesizing new movements is made vastly easier when the general motion space is reducible to simpler, flexible elements. The concept of dimensionality reduction is useful as a general mathematical tool and there are two specific techniques which contribute to this thesis as examples.

The first is principal components analysis (PCA), which uses an eigenvalue decomposition of a covariance matrix to decompose a large sample population into a much smaller number of dominant variations [76]. The level of dimensionality reduction depends on the number of terms retained in a linear expansion of the so-called principal components. Reference [49] above demonstrates the PCA technique for analyzing and classifying human arm movement data while reference [37] applies the method for characterization of helicopter aerobatic flight data. While principal components are easy to use, their linear series format makes them difficult to apply to trajectory generation for nonlinear systems.

A method that attempts to work directly with nonlinearity is the locally linear embedding (LLE) approach [127, 142]. Here, an analysis algorithm processes a general nonlinear data set into its dominant variations, first locally and then globally. The outcome is a something of a coordinate chart representation of the population, whose coordinate axes describe continuous modulations in the data. The references use the technique to concisely characterize large sets of hand-drawn characters and images of facial expressions. Interestingly, LLE provides a very similar function to parametrized maneuvering (as presented in this thesis), where a small number of variables can be used to chart and manipulate larger-dimensional vectors on continuous, nonlinear surfaces.

1.2.6 Parametric Programming in Control Systems

Applying the notion of dimensionality reduction to the trajectory generation problem, this thesis uses a relaxed version of nonlinear parametric programming to map out the feasible maneuver space of autonomous vehicles. The discipline of parametric programming is gradually finding a place in modern control practice; this section examines several recent works that have used this optimization subspecialty for system feedback control.

References [9] and [117] consider the model predictive control/receding horizon control (MPC/RHC) problem for linear systems, noting that control synthesis typically entails on-line solution of a quadratic program. In many situations, an initial vehicle state forms a parametric description of the quadratic program. Therefore, a parametric map of the corresponding solution provides a computationally simpler alternative to iterative optimization. These works develop a multiparametric quadratic programming (mp-QP) formulation, where optimal solutions can be computed off-line as piecewise-affine representations, reducing the on-line control procedure to an explicit function evaluation. The solution space breaks into partitions since the variations in the problem initial conditions force changes in the optimal solution active constraint set.

Reference [147] studies and improves the algorithms for solving the off-line mp-QP problem, while [21] presents algorithms for more efficient on-line function evaluation.

Noting that large numbers of constraints can fracture the solution space into an exceedingly complex array of affine function partitions, [10] simplifies the solution process by relaxing the Karush-Kuhn-Tucker optimality conditions and provides a simple scalar tuning variable to govern the solution complexity. In one extreme, this variable fully relaxes the quadratic programming problem, resulting in a static linear control law; in the other extreme the full mp-QP solution is applied, resulting in maximum segmentation of the solution space.

Reference [20] fully discusses the corresponding theory for various control scenarios, including MPC and RHC, constrained robust control, and hybrid plants posed as mixed logic-linear dynamical systems. Finally, [75] applies the method to more general systems by approximating nonlinear MPC as a series of local linear problems allowing application of the mp-QP approach.

1.3 Hierarchical Motion Planning

As is apparent in the above literature review of trajectory and motion processing methods, a multilayer approach to planning is common to aerospace, robotic, animated, and biological systems and is an extremely active research topic in all these fields. Indeed, as discussed in [1], hierarchical decomposition is logical, if not critical, when both natural and man-made systems interact with their surrounding worlds. The clear benefit is a decoupling of information and computing challenges into efficient and tractable subproblems.

For autonomous vehicles, it is reasonable to expect that a high-level view of guidance should not be concerned with the detailed "inner-loop" interactions of dynamics and control, but instead use a more abstracted approach of simply considering what motions *types* are available and then using them as building blocks to solve a given problem. In this context, continuously parametrized maneuver classes provide the desired level of abstraction for motion planners, with underlying detailed algorithms computing the dynamics-based system control and reference trajectories.

This section briefly reviews several current methods for autonomous vehicle motion planning, emphasizing those techniques that employ hybrid system representations. It is then discussed how flexible maneuvering is a natural contribution to the decoupled planning problem.

1.3.1 Emerging Methods

Loosely speaking, a hybrid system model combines continuous-valued state equations (in either continuous or discrete *time*) with the ability to switch among a purely discrete collection of operating modes. For example, one might think of an aerospace system switching between different tasks, such as take-off, cruise flight, maneuvering modes, and landing, each requiring a specialized model and control design. As shown in the following references, hybrid systems are extremely useful in planning and controlling dynamic systems, and often take advantage of "motion primitive" abstractions of low-level dynamics.

The notions of hybrid and hierarchical systems are closely related, with the latter being a logical way of organizing information, typically employing the former to represent specific system dynamics. In the case of motion control, reference [22] mathematically outlines various structures of hybrid systems, pointing out that motion control is often a feedback system based on trajectory geometry and physical force-displacement behaviors. Similar to [1] above, the work points out hybrid system commonalities between man-made and biological systems.

As a useful model for various engineering devices, reference [11] considers mixed logical-dynamical (MDL) systems, useful for describing combinations of linear dynamics with integer decision variables, all subject to linear constraints. Specific topics include stability criteria and tracking of reference trajectories, with mixed integer-quadratic programming (MIQP) providing the main optimization workhorse for receding horizon control of MDLs.

Looking specifically at robotic and aerospace motion planning applications, there are several recent works advocating multiscale and hybrid approaches. An important example is reference [2], which demonstrates the benefits of planning at multiple levels, so that higher layers can focus on the strategies of motion while lower levels handle the specific execution details from a system dynamics input-output viewpoint. The reference illustrates guidance of a nonholonomic ground robot through a known maze-like obstacle field, gradually refining the results of a randomized, exhaustive search algorithm.

Hierarchical principles also appear in reference [29], which considers particular classes of second-order underactuated mechanical systems, defining notions of system controllability and illustrating a two-level kinematic planning and path-scaling approach for finding collision-free trajectories. This method places the path time-parametrization algorithms seen earlier for robotic applications ([12, 38, 67]) in a larger motion planning context.

Employing an explicit notion of motion primitives for mechanical dynamic systems, [28] considers the class of underactuated mechanical systems evolving on Lie groups, defining two specific control-based primitives for maintaining and changing velocity. Using these basic building blocks, motion planning algorithms then determine how to reposition and reorient example systems. A mathematical definition of motion primitives also appears in [51] and [53], which focus on autonomous vehicle models similar to that seen in [28]. Here an abstracted system configuration vector is given and controlled by executing combinations of motion primitives, defined as a fixed set of trim states (i.e. equilibria) and static maneuvers. A hybrid system representation, termed a "maneuver automaton", then captures the configuration and motion primitives in a single framework, allowing path planning via dynamic programming and randomized search algorithms. Demonstrated applications of the method include navigation through a dynamically changing obstacle field and guidance of an agile rotorcraft, seen in [100]. Reference [52] shows how the maneuver automaton extends into a formal language, useful for obtaining explicit solutions to optimal planning problems with on-line solution sets based on linear programming methods.

Another example of motion primitive-based guidance of rotorcraft is given by ref-

erence [41], where the primitives correspond to intuitive flight modes, such as take-off, steady cruise, and turning. Motion planning involves casting the vehicle dynamics as linear system representations of primitives, which are themselves controlled by nonlinear methods such as the differentially flat model inversion approach, as referenced in section 1.2.1. To navigate through a set of waypoints, the planner selects a path made up of sequences of primitives, minimizing the overall vehicle state error relative to the given waypoints.

The final hybrid system examples cast path planning as a mixed integer-linear programming (MILP) problem, akin to the MDL framework seen above in [11]. Similar to the maneuver automaton notion, the general flight envelope is represented by sets of linearly-controlled modes and fixed maneuvering elements. The main difference lies in the automaton's requirement that non-maneuvering modes be fixed equilibrium conditions (of variable time duration) while the MILP framework allows the system to operate in continuously evolving linear system modes, often approximating command-following closed-loop systems. However, the maneuver automaton admits off-line solution by dynamic programming over a wide ranging set of initial conditions. Storage of the corresponding cost, or value, function makes for extremely efficient in-flight implementation while MILP requires on-line optimization.

Extensive discussion and examples of the MILP method appear in [133, 134], which consider rotorcraft guidance scenarios subject to acceleration and velocity-dependent turn rate constraints while including the use of aggressive maneuvers, such as those developed in [54] and [56]. Reference [125] illustrates MILP for finding minimum-time aircraft paths through waypoints and obstacle fields and also multiple aircraft flying in proximity subject to collision avoidance constraints. Other problem variations exist, as shown in [8], where MILP plans now include a terminal cost map, obtained by building up visibility-to-goal-state cost functions with a Dijkstra's algorithm-based search procedure. This added feature aids airplane navigation through obstacle fields in receding horizon mode. Reference [132] augments the receding horizon implementation to guarantee safe trajectories, avoiding a catastrophic collision with an obstacle previously outside the finite planning horizon. Finally, [95] integrates MILP into a higher-level dynamic programming approach designed to navigate clusters of autonomous agents through partially-known urban environments. Guidance is cast as a graph traversal problem with a lower-level MILP function performing shortest path waypoint navigation subject to kinematic and dynamic constraints.

1.3.2 Benefits of Flexible Maneuvering

The above discussion of hierarchical and hybrid motion planning methods demonstrates the advantages of breaking a general dynamics and control problem into multiple layers. Overall strategies to reach a set of waypoints or navigate an obstacle field can be handled at a high level by considering the basic vehicle capabilities. Intermediate levels can produce the required reference trajectories, using knowledge of system dynamics, while lower levels can process sensor information and close control loops. For best efficiency, it is important that each level considers only the information necessary to perform its specific function, without having to "overmodel" the

immediate problem and include details that are better handled at other levels.

For those methods that employ notions of motion primitives, the representation and richness of the primitives is crucial to creating effective planning schemes. Mathematical representation involves choosing a concise set of variables that represent the essence of the primitive and allow easy manipulation in other frameworks, such as optimization settings. The benefits of richness are intuitive: the more flexible the primitives, the greater the number of possible motions a planner can choose from and the lower the cost of the overall guidance objective.

Chapter 5 demonstrates the application of parametrized maneuvers as motion primitives in the existing mixed integer-linear programming planner. As continuous mappings $p(\alpha)$, parametrized maneuvers add tremendous richness to motion planners by capturing an infinite number of maneuvers within a single class. In previous versions of the MILP framework, individual fixed maneuvers were each assigned a binary decision variable that indicates when a given maneuver is to be executed. Therefore every motion in a set of fixed maneuvers would require its own binary variable, its own characteristic state transformation matrix, as well as its own set of required initial conditions, such as a vehicle speed or elevation angle. Given the branch-and-bound algorithms typically used to solve MILP optimization problems, introduction of large numbers of maneuvers can greatly hinder the ability to quickly solve planning problems, potentially negating the potential benefits of including them.

As will be shown in Chapter 5, parametrized maneuver classes $p(\alpha)$ allow an infinite number of maneuvers with a continuous range of boundary conditions to be represented by a *single* binary decision variable and transition matrix. Thus problem dimensionality is kept in check while simultaneously enriching the scope of possible motions and representing similar maneuvers as a unified class. Naturally, since MILP has a linear programming structure, maneuver families have to obey a single linear transformation relation. However, as will be shown, the linearity requirements appear simply as additional components in a parametrized equality constraint vector $h(p, \alpha)$, thus restricting motions to specific submanifolds within the overall feasible maneuver space. For motion planners with requirements different than MILP, the appropriate elements can be added to $h(p, \alpha)$. In the formulation of Chapter 5, the vehicle state at the instant of a decision to maneuver determines the specific value of α and thus, through $p(\alpha)$, the entire maneuver state and control history.

Although MILP is not the only optimization framework that can be adapted to handle maneuvers, it is chosen in this thesis as an example motion planner for four reasons. First, the required modifications to the existing path planning formulations are essentially trivial. Flexible maneuvers now have the same representation as previously static motions, so richness is added with virtually no extra complexity in problem formulation. This feature also makes it easier to compare performance with and without parametrized maneuver families.

Second, for the planar three degree-of-freedom helicopter application of Chapters 4 and 5, the closed-loop system behavior is easily captured as a command following linear time-invariant modes. Changes in speed and elevation are straightforward system inputs in the MILP formulation. Note that for general helicopters in three-dimensional space and with six degrees-of-freedom, it would be necessary to approx-

imate nonlinear body-to-inertial frame transformations in a manner amenable to linear programming. Because of the restriction to longitudinal dynamics of the example system, there is no need for such approximations in the given MILP formulation.

Third, the mixed integer-linear programming framework allows a number of interesting problem formulations. Features include minimum-time and minimum state-error objective functions, inclusion of polyhedral obstacles, specification of a wide variety of maneuver initiation requirements, and the ability to impose numerous state and control vector constraints.

Finally, introduction of parametrized maneuvers into the MILP framework creates a fully flexible motion primitive-based planning method, with no restriction to either discrete trim state or discrete maneuver objects. The MILP analogy of the equilibrium state is a linearly controlled system that approximates the closed-loop operation of the system in command-following mode. This approximation needs to include only those states necessary for planning purposes and relegates detailed system modeling to lower-level control algorithms.

1.4 Thesis Contributions

Overall, this thesis presents an application-oriented method of continuously parametrizing vehicle maneuvers. The technique is of interest both as a general simplification of nonlinear dynamics into characteristic behavior sets and as a useful aid to existing motion planning algorithms. The point-of-departure is the traditional nonlinear programming approach to numerical optimal control and the ultimate payoff is an easily formulated method for real-time flexible trajectory generation. The following is a listing of the primary contributions.

• The thesis develops a trajectory interpolation framework, in which example vehicle motions of any origin define continuously parametrized maneuver families. These families reduce the real-time trajectory generation problem for nonlinear systems to a much simpler numerical integration procedure. The interpolation method directly addresses three classic challenges of nonlinear programming:

(1) *extreme sensitivity of local minima to initial trajectory guesses*: Instead of using initial guesses to indirectly influence the outcome of a single laborious nonlinear programming solution, example trajectories now allow the user to directly shape an entire maneuver class.

(2) *nonguaranteed convergence*: Removal of iterative optimization from the on-line algorithm results in a fast trajectory generator.

(3) *unnecessary generality*: Rather than have a nonlinear programming solver operate on a large number of free trajectory variables (that describe virtually the entire flight envelope), the trajectory interpolator uses only as many free quantities as there are intuitive variations within a maneuver class, achieving a large dimensionality reduction.

- Elimination of an explicit objective function from the interpolation framework allows parametrization both optimal and suboptimal trajectories, giving the user control over many useful vehicle motions.
- Relaxation of the strict optimality necessary conditions reduces the problem dimension by roughly half in comparison to full nonlinear parametric programming, the finite-dimensional analog of neighboring extremal control.
- The ability to use human-piloted flights as example motions provides a way of incorporating expert operator knowledge of system dynamics directly into the maneuver definition process.
- Despite elimination of the objective function from the trajectory generation procedure, the user can retain control of the cost performance of a given maneuver class by imposing suitable equality constraints.
- The continuously parametrized maneuvers capture the full state and input behavior of a given trajectory, thus providing a direct interface with feedforward nonlinear control methods.
- A complexity analysis gives lower and upper bounds on the computing load of the trajectory generation algorithm.
- The parametrized maneuvers are demonstrated in an existing mixed integer-linear programming hybrid motion planning scheme, capturing an entire maneuver class with a binary decision variable and affine state transformation model.
- Parametrized trajectory families eliminate the need to individually encode discrete maneuvers in most planning schemes, thus preventing a build-up in problem dimensionality while simultaneously improving the overall planning performance.
- Experimental demonstration on a three degree-of-freedom helicopter validates the essential features of flexible maneuvering and hybrid motion planning methods. In particular, parametrized trajectories, piloted-flight motion capture, and feedforward control design are demonstrated on a nontrivial nonlinear system.

1.5 Thesis Outline

The dissertation progresses from traditional trajectory optimization methods, such as optimal control and nonlinear programming, to a set of problem relaxations allowing manipulation of any feasible trajectories, to experimental demonstration of flexible maneuvers in an existing hybrid motion planner.

Chapter 2 establishes the method for creating flexible and feasible maneuvers. Beginning with the nonlinear programming necessary conditions, the chapter examines the motivations behind parametric programming and summarizes the most useful mathematical tools behind it: parametrized solution sets and continuation methods. Relaxations of the optimality conditions then make it possible to easily create parametrized maneuver families, with knowledge of multiple known system behaviors replacing optimality as the continuation process driver. It is shown how to create families of feasible system trajectories with a motion interpolation process, using a small set of variables to describe intuitive maneuvering effects. The chapter then outlines a method for capturing observed real-world system motions as feasible points,

in particular those demonstrated by human pilots.

Chapter 3 highlights various properties of the interpolation method. Of greatest interest is the consequence of discarding optimality as the foundation of parametric programming. Through a series of examples, the discussion shows that, in some cases, optimality is recovered even though not explicitly enforced. In other situations, the user can exert direct control over the shape of the maneuver families, helping to close the optimality gap. In addition, a numerical analysis considers the number of elementary operations required for trajectory interpolation, taking into account an active set switching mechanism. The result is a lower and upper bound on the expected computing load for trajectory generation. The last two sections discuss the continuity properties of parametrized motion families and review some of the interesting applicable nonlinear system types.

Chapter 4 gives an in-depth experimental application to a three degree-of-freedom tabletop-mounted helicopter, emulating the challenges of working with nonlinear rotary-wing systems. Casting the vehicle trajectory in terms of a spline basis and developing a model inversion relationship results in a parametrized feasible space with a large number of nonlinear equality and inequality constraints. The chapter then discusses example trajectory generation for typical helicopter motions, using both off-line nonlinear programming and motion capture to define flexible maneuver families.

Chapter 5 gives an application of flexible maneuvering to hybrid motion planning. While not the only candidate path planner, mixed integer-linear programming (MILP) provides a near-ideal setting for integrating flexible maneuvers with general unsteady flight of a stability-augmented system. The chapter shows how to adapt flexible trajectories to meet the specific requirements of MILP. One and two-dimensional scenarios then illustrate the combined path planning and trajectory generation scheme in interesting scenarios.

Chapter 6 concludes the thesis by discussing the method attributes and reviewing the remaining technical questions of interest. A number of future applications and expansions are proposed, including demonstration on other vehicles, use of parametrized maneuver families as a method for human-machine learning, and integration into other guidance schemes.

Two appendices provide necessary supporting material. Appendix A gives a detailed discussion of the linear and nonlinear helicopter modeling and system identification. Appendix B first discusses the construction and elementary properties of B-spline basis functions and then derives several differentiation rules useful for the methods of Chapters 2, 4, and 5.

Chapter 2

Parametrized Maneuvers

The preceding chapter laid out the fundamental motivations for creating parametrized maneuver classes. Representation of complex vehicle motions by low-dimensional sets is a logical way to organize the general vehicle flight envelope and serves as a valuable tool for motion planning. In addition, generation of system trajectories from scratch is a nontrivial problem in general, often involving considerable computing resources and making clear the need for faster methods for obtaining feasible motions. But the matter would not be solved even if trajectory optimization algorithms ran arbitrarily fast. Many vehicle motions, such as aggressive or aerobatic maneuvers, may not minimize an obvious objective function and existing solution methods do not readily exploit known, reproducible system behaviors. There is a need for methods that combine the rigor of traditional trajectory optimization with a greater flexibility to work with any feasible system motions, including those observed in motion capture data. A clear merit of such methods would be the ability to take existing system trajectories, of any type, and alter them to meet new needs, all while satisfying underlying equations of motion.

This chapter presents such a method, beginning with the well-established problem formulations of continuous-time optimal control and finite-dimensional nonlinear programming. Examination of the optimality conditions from a perturbational viewpoint indicates a clear method for creating classes of parametrized extremal motions. A relaxation of the first-order necessary criteria to include all trajectories, not just optimal ones, then makes apparent a tool to feasibly manipulate maneuvers with great flexibility. The resulting framework starts with multiple user-selected example motions and produces low-dimensional feasible maneuver families, retaining the attributes of the examples.

The chapter begins with two background sections that provide a foundation for creating maneuver families. Section 2.1 reviews trajectory optimization by nonlinear programming while Section 2.2 introduces nonlinear *parametric* programming and numerical continuation methods as techniques for finding families of optimal trajectories. The remaining sections then use these mathematical programming formulations to develop the new methods of this thesis. Section 2.3 develops the parametrized maneuver framework, beginning with a relaxation of optimal parametric methods that allows access to the entire vehicle flight envelope. Section 2.4 presents a motion cap-

ture technique for mapping observed real-world vehicle motions to the feasible space of a nonlinear model; Section 2.5 discusses interesting alternative problem formulations; and Section 2.6 relates the specific techniques of this thesis to existing tools of nonlinear optimization.

2.1 Trajectory Design via Nonlinear Optimization

Feasible trajectory generation for nonlinear systems naturally requires nonlinear methods, typically in the form of iterative optimization algorithms. The intimately related fields of continuous time optimal control and finite-dimensional nonlinear programming (NLP) provide the tools for casting such problems, obtaining rigorous optimality criteria, and deriving solution methods. This section briefly reviews the trajectory optimization problem statement, emphasizing the NLP framework, thereby laying out the specific techniques needed for trajectory manipulation.

2.1.1 Problem Formulation

When modeling a nonlinear aerospace or mechanical system, it is necessary to derive physics-based equations of motion, identify initial (and perhaps terminal) system states, and describe operational constraint limits. The resulting mathematical expressions typically appear as continuous time equations and inequalities, and are therefore well-suited for standard optimal control methods. The field of continuous time optimal control is certainly immense and includes many possible problem formulations [16, 26]. To set the properly stage for the methods of this thesis, it is necessary to start with a general problem statement.

Traditional Optimal Control

In what follows, assume the general system equations of motion are smooth, deterministic, and of the form $\dot{x} = f(x, u)$, where x is the system state and u is the control input. A general descriptive system behavior $z(t)$ may be defined as some general function of state and/or input:

$$z(t) = z(x(t), u(t)). \tag{2.1}$$

When finding a useful system behavior, one typically desires an optimal motion that obeys the dynamics and limits of the system. This task is cast as the general mathematical program

$$\begin{aligned} &\min_{z(t)} M(z(t)) \\ &\text{subject to} \quad z(t) \in \mathbf{Z}, \end{aligned} \tag{2.2}$$

where $\mathbf{Z}$ represents the set of dynamically feasible motions that obey the system constraints.

References [16], [26], and [116] provide tools to more exactly frame this problem. Typically, one seeks a behavior $z(t)$ that minimizes some objective metric of the motion quality over a time interval $[0, t_f]$, often measured by a cost function J based on a terminal cost expression ϕ_f and a running path penalty function l:

$$J = \phi_f(x(t_f), t_f) + \int_0^{t_f} l(x(t), u(t), t)dt. \tag{2.3}$$

The general behavior set $\mathbf{Z}$ includes the requirement of satisfying the equations of motion

$$\dot{x}(t) = f(x(t), u(t)), \tag{2.4}$$

which are assumed here to be autonomous, that is, time invariant. Additional elements of $\mathbf{Z}$ may dictate the initial and final boundary conditions of the motion in terms of some vector functions ψ_0 and ψ_f, obeying some upper and lower bounds:

$$\begin{aligned} \psi_{0,lb} \leq \psi_0(x(0), u(0)) \leq \psi_{0,ub} \\ \psi_{f,lb} \leq \psi_f(x(t_f), u(t_f)) \leq \psi_{f,ub}. \end{aligned} \tag{2.5}$$

Setting corresponding components of $\psi_{0,lb}$ and $\psi_{0,ub}$ to the same value results in a strict equality constraint, and similarly for $\psi_{f,lb}$ and $\psi_{f,ub}$. In addition to the boundary constraints, there may exist running constraints during some portion of, or along the entire trajectory. Such constraints typically involve transforming the system behavior of Equation (2.1) and its time derivatives according to some time-varying nonlinear vector function ψ with associated lower ψ_{lb} and upper ψ_{ub} bounds:

$$\psi_{lb}(t) \leq \psi(x, \dot{x}, \ldots, x^{(l)}, u, \dot{u}, \ldots, u^{(k)}, t) \leq \psi_{ub}(t) \qquad \forall\ t \in [0, t_f]. \tag{2.6}$$

The function ψ might include general holonomic and nonholonomic path constraints as well as the frequently employed time-invariant state and control bounds, written explicitly here as:

$$\begin{aligned} x_{lb} \leq x(t) \leq x_{ub} \qquad \forall\ t \in [0, t_f] \\ u_{lb} \leq u(t) \leq u_{ub} \qquad \forall\ t \in [0, t_f]. \end{aligned} \tag{2.7}$$

Unbounded components of x occur when the corresponding elements of x_{lb} and x_{ub} are negative and positive infinity, respectively. Similar considerations apply to u_{lb} and u_{ub} as well.

The generic problem statement of Equations (2.2) through (2.7) includes many classic trajectory optimization problems and is addressed by the methods of variational calculus [31, 116], useful for obtaining the so-called Euler-Lagrange necessary conditions for optimality. References [25], [26], and [116] discuss the variational problem in detail and give procedures for deriving necessary conditions, as well as sufficiency conditions and even analytic solutions in some special cases. Since the present goal is a real-time trajectory generator, it is appropriate to now discuss a corresponding finite-dimensional framework.

Finite Parametrization

In most practical cases, a trajectory parameter vector p reduces the above continuous time optimization problem to one in finite dimensions. That is, the general system behavior $z(t)$ is expressed as $z(t;p)$, where p represents z in terms of some polynomial, spline, exponential, wavelet, or other finite basis. The clear benefit is a reduction of the infinite-dimensional optimization problem (2.2) to one of optimizing over p in a finite space, a setting better suited for nonlinear programming methods and modern computing resources.

References [16], [17], [18], [63], [68], and [102] describe many techniques for transforming optimal control problems to parameter optimization problems. The transformations often take either a direct form, where the goal is to minimize J without regard to the continuous time necessary conditions, or an indirect form, where the emphasis is on satisfying the Euler-Lagrange relations. Another distinction comes between transcription methods (also known as collocation) and shooting methods. The former involve using p to parametrize both $x(t)$ and $u(t)$ simultaneously and enforcing dynamic consistency between the two through an approximation of the equations of motion. The latter methods typically involve parametrizing only $u(t)$ in terms of p, so that the equations of motion (2.4) must be integrated to find $x(t)$.

For now, it is assumed that some general $p \in R^n$ parametrizes the system behavior. The NLP must contain a scalar objective function $f : R^n \to R$ that provides some measure of trajectory quality. Examples include functions measuring trajectory time, energy, control effort, position error, and threat exposure.

In addition to the objective f, there are typically m scalar equality constraint functions $h_i : R^n \to R$, each of the form $h_i(p) = 0$. These are frequently combined into a single vector function $h : R^n \to R^m$ defined by $h(p) = [h_1(p), \cdots, h_m(p)]^T$ and expressed collectively as $h(p) = 0$. Of course, there must be available degrees-of-freedom in p, i.e. $m < n$, to have a well-posed optimization problem.

The equality constraint components typically have one of two roles, the first being to maintain dynamic feasibility. Sometimes referred to as zero-defect, collocation, or consistency constraints, these components of h make sure some finite approximation of the equations of motion are satisfied. This class of constraints typically includes any holonomic and/or nonholonomic equality constraints present in the system [31]. The other primary of role of $h(p)$ is to enforce boundary condition constraints, such as the initial and final state and control setpoints.

In addition to h, there may be r scalar inequality constraint functions $g_j : R^n \to R$ corresponding to single scalar bounds $g_j(p) \leq 0$. Similar to h, these are concatenated into a single vector inequality $g(p) \leq 0$ with $g : R^n \to R^r$. Such constraints typically arise from discretization of Inequalities (2.5) through (2.7).

With the objective and constraint functions in hand, the general nonlinear programming optimization problem is

$$\begin{aligned} & \min_p f(p) \\ \text{subject to} \quad & h(p) = 0 \\ & g(p) \leq 0 \end{aligned} . \tag{2.8}$$

The eventual method for creating parametrized maneuvers will be in terms of operations on the finite vector p and is based on the optimality conditions of Program (2.8). References [14], [18], [47], and [141] discuss how to solve the above nonlinear programming problem.

2.1.2 Nonlinear Programming Optimality Conditions

First-order optimality conditions indicate when a given vector p is an extremum of Program (2.8) while second-order conditions help classify the candidate point p by type, that is, whether it is a local minimizer, maximizer, or other critical point.

This subsection will not derive the general NLP optimality conditions but instead briefly review them to provide motivational background material. References on the theory and practice of NLP optimality conditions, especially those involving Lagrange multiplier theory, include [14], [46], [47], and [141].

Several definitions and notations will be useful throughout this chapter. The first are those of first and second-order derivatives. For a vector $x \in R^n$ and a function $F : R^n \rightarrow R^a$, the derivative, or Jacobian, of F with respect to x is a mapping $D_xF : R^n \rightarrow R^{a \times n}$ and is given by the array of first partials $D_xF = [D_{x_1}F, \ldots, D_{x_n}F]$. The gradient of F with respect to x is the transpose mapping $\nabla_x F : R^n \rightarrow R^{n \times a}$ given as the array $\nabla_x F = (D_xF)^T$. For a scalar function $G : R^n \rightarrow R$ the Hessian is a mapping $\nabla_{xx}G : R^n \rightarrow R^{n \times n}$ and is the matrix of second partials $\nabla^2_{xx}G = D_x(\nabla_x G)$. The standard Lagrangian function $L : R^n \times R^m \times R^r \rightarrow R$ has the definition

$$L(p, \lambda, \mu) = f(p) + \lambda^T h(p) + \mu^T g(p), \tag{2.9}$$

where $\lambda \in R^m$ and $\mu \in R^r$ are vectors of equality and inequality constraint Lagrange multipliers, respectively. In addition, for a feasible point p of the nonlinear program (2.8) the active set at p is the set J_0 of component indices for those inequalities *not* strictly satisfied at p:

$$J_0 = \{j | g_j(p) = 0\}. \tag{2.10}$$

Let $g_{J_0}(p)$ denote the corresponding active set vector, which is simply the ordered components of g that are individually active at p. Let r_0 denote the dimensionality of J_0, that is, the *number* of active components in g. Note that in general, $0 \leq r_0 \leq n - m$.

A feasible point p of (2.8) is called a *regular point* if the equality constraint gradients, $\nabla_p h_i(p)$ with $i \in \{1, \ldots, m\}$, and active inequality constraint gradients, $\nabla_p g_{J_0}(p)$, are all linearly independent. Finally, it is assumed that h and g are sufficiently differentiable to compute the needed quantities in the remainder of the chapter.

The Karush-Kuhn-Tucker (KKT) first-order necessary conditions for optimality state that if p^* is both a local extremum of the nonlinear program (2.8) and a regular point, then there exist Lagrange multiplier vectors $\lambda^* = [\lambda_1^*, \ldots, \lambda_m^*]^T$ and $\mu^* = [\mu_1^*, \ldots, \mu_r^*]^T$ such that

$$\nabla_p L(p^*, \lambda^*, \mu^*) = 0 \tag{2.11}$$

with

$$\begin{aligned} \mu_j^* &\geq 0 \quad \forall\ j \\ \mu_j^* &= 0 \quad \forall\ j \notin J_0. \end{aligned} \tag{2.12}$$

Further, if p^* is a local minimizer of (2.8), then the second-order Hessian relationship

$$y^T \cdot \nabla_{pp}^2 L(p^*, \lambda^*, \mu^*) \cdot y \geq 0 \tag{2.13}$$

must hold for any nonzero $y \in R^p$ that is a first-order feasible direction at p^*, that is, a direction y such that

$$\begin{aligned} D_p h_i(p^*) \cdot y &= 0 \quad \forall\ i = 1, \ldots, m \\ D_p g_j(p^*) \cdot y &= 0 \quad \forall\ j \in J_0. \end{aligned} \tag{2.14}$$

These conditions can be derived in a number of ways, virtually all of which follow from casting the constrained optimization problem in an unconstrained form and then examining the corresponding unconstrained optimality conditions [14]. The Lagrange multipliers typically have the interpretation of dual variables to Program (2.8) and are useful in examining the sensitivity of the cost function to changes in constraint problem data. Specifically, if Program (2.8) is considered as a member of the class of general problems

$$\text{subject to} \quad \begin{array}{l} \min\limits_p f(p) \\ h(p) = u \\ g(p) \leq v \end{array}, \tag{2.15}$$

one can consider an optimal solution family $p^*(u, v)$ for small variations in vectors $u \in R^m$ and $v \in R^n$. It can be shown that the optimal multipliers to the original problem satisfy

$$\begin{aligned} \lambda^* &= -\nabla_u f(p^*(u, v))|_{(u=0, v=0)} \\ \mu^* &= -\nabla_v f(p^*(u, v))|_{(u=0, v=0)}, \end{aligned} \tag{2.16}$$

giving them the interpretation of cost sensitivities with respect to changes in the problem data. For the equality constraints, a positive multiplier $\lambda_i^* > 0$ implies that increasing the right hand side of $h_i(p) = u_i$ (where $u_i = 0$ in the nominal problem case) takes the solution point p^* closer to an unconstrained local minimum of $f(p)$, thus decreasing the cost $f(p^*)$. A negative multiplier $\lambda_i^* < 0$ has the opposite interpretation: increasing u_i greater than 0 moves the solution point p^* farther from a local minimum of f, thus increasing the cost $f(p^*)$. A zero multiplier $\lambda_i^* = 0$ implies that the equality constraint is locally redundant, that is, that the solution to Program (2.8) would not change if $h_i(p) = 0$ were removed from the constraint set.

Considerations for the inequality constraints follow similarly based on Equations (2.16), so that $\mu_j^* \geq 0$ for all j since increasing the right hand side of any $g_j(p) \leq v_j$

constraint can only relax the constraint set and lower the objective function value. Note that $\mu_j = 0$ must hold for all $j \notin J_0$ since these constraints are not active and thus play no role in determining the solution point p^* and therefore no role in the local cost sensitivity. For the remaining active inequality constraints, it is typical that $\mu_j^* > 0$, but $\mu_j^* = 0$ may occur for those $j \in J_0$ that happen to be redundant to the problem statement, just as with the equality constraints.

For completeness, the second-order *sufficient* conditions for a local minimum of Program (2.8) state that if p^* is a feasible point satisfying

$$\nabla_p L(p^*, \lambda^*, \mu^*) = 0 \tag{2.17}$$

with

$$\begin{aligned} \mu_j^* &> 0 \quad \forall\ j \in J_0 \\ \mu_j^* &= 0 \quad \forall\ j \notin J_0 \end{aligned} \tag{2.18}$$

and

$$y^T \cdot \nabla_{xx}^2 L(p^*, \lambda^*, \mu^*) \cdot y > 0 \tag{2.19}$$

for any $y \neq 0$ such that

$$\begin{aligned} D_p h_i(p^*) \cdot y &= 0 \quad \forall\ i = 1, \ldots, m \\ D_p g_j(p^*) \cdot y &= 0 \quad \forall\ j \in J_0 \end{aligned} \tag{2.20}$$

then p^* is a *strict* local minimizer of Program (2.8). This criterion fulfills the usual interpretation of a local minimizer as a zero derivative point with strictly positive second derivative, although here the curvature is restricted to only those directions that are locally feasible to first-order. Note that, in contrast, the necessary conditions only require a nonnegative second derivative along feasible directions.

Often, a more compact version of the first-order necessary conditions of Equations (2.11) and (2.12) is given by the system of $n + m + r$ equations

$$\begin{bmatrix} \nabla_p L(p^*, \lambda^*, \mu^*) \\ h(p^*) \\ M(\mu^*) g(p^*) \end{bmatrix} = 0, \tag{2.21}$$

where $M(\mu) \equiv \text{diag}(\mu_1, \ldots, \mu_r)$ is a diagonal matrix of inequality Lagrange multipliers. The last r components of this vector equation enforce the so-called complementary slackness condition. That is, for every $j \in \{1, \ldots, r\}$ it must hold that $\mu_j^* g_j(p^*) = 0$, as follows from the definition of the active set J_0 and Equations (2.12).

These collected necessary conditions, along with the sufficiency condition of Equations (2.17) through (2.20), enumerate the criteria nonlinear programming algorithms attempt to satisfy. They form the point-of-departure for nonlinear *parametric* programming, the method which provides the tools for creating parametrized maneuver families. Note that the final algorithms will not follow parametric programming in its strict form, but instead relax the optimality conditions to provide access to all

system trajectories, not just those that are mathematically optimal.

2.1.3 Nonlinear Programming Considerations

The previously cited references on NLP solution methods give many different techniques for solving the generic optimization problem. Although the procedures come in many different varieties (sequential quadratic programming methods, penalty and barrier function methods, primal-dual interior point methods, etc.), they have several aspects in common. Specifically, they start with an initial guess of the solution (or solve a subproblem to determine one) and then proceed iteratively to find a point satisfying the necessary and sufficiency conditions. From a trajectory generation point of view, the general NLP approach is very attractive for its extreme generality and applicability to countless classes of systems.

However, there are several key limitations to nonlinear programming for real-time trajectory solution. First, the necessary and sufficiency conditions only guarantee locally optimal solutions and cannot make any assertions about the global standing of a solution point p^*. As such, and due to the nature of iterative algorithms, a solution point p^* is *extremely* sensitive to its initial guess $\hat{p}_0$. Thus, there is a tremendous burden placed on the engineer to find good initial estimates of trajectory profiles.

Secondly, NLP solver convergence is not guaranteed in general. Based on the quality of the initial guess and the algorithm properties, the method may not converge to a satisfactory solution or even find a candidate feasible point.

Finally, the parametrization p of the trajectory may be too general and not specific enough for the desired qualitative solution type. That is, p may have too many degrees-of-freedom for an NLP solver to sort through quickly, thus wasting valuable time that could go towards finding useful solutions.

As discussed in Chapter 1, these three considerations, among others, are some of the prime motivators for developing parametrized solution families $p(\alpha)$, where here "parameter" refers not to p, but to an α vector that indicates exactly what motion is sought within a maneuver class. The necessary conditions covered so far will now help define a method of parametrically describing optimal solution families $p^*(\alpha)$ once one single optimizer is known for some value of α. A relaxation of these techniques will then allow creation of more general families $p(\alpha)$, where all feasible points p will be allowed.

2.2 Nonlinear Parametric Programming

Now comes a transition from the general nonlinear program (2.8) to the general nonlinear *parametric* program. From this point forward, the vector p, earlier termed the finite trajectory "parametrization", is simply referred to as a "trajectory", "maneuver", or "solution point". The new variable $\alpha \in R^q$ gets the name "parameter" and acts (when $q \ll n$) as a lower-dimension descriptor of a class of feasible trajectories $p(\alpha)$, the central object of this chapter. One might call $p(\alpha)$ a "parametrization of a parametrization", which would be accurate, but the main emphasis is on developing

a concise way of representing feasible maneuvers in terms of a small number of variables. The payoff will be a concise representation of motion classes that easily makes possible parametrized maneuver sets, starting from a few known feasible instances. All on-line computation can then be spent evaluating $p(\alpha)$ for the desired feasible motion, and not resorting to on-line nonlinear programming, which has no convergence guarantees and is therefore hard to certify for real-time use. Instead, nonlinear programming can be used *off-line* to generate high-quality example motions. In addition, motions exhibited by the vehicle in testing and pilot evaluation can be cast as feasible points and used to design the parametrized families $p(\alpha)$.

This section begins the transformation from nonlinear programming to what will be termed trajectory, or solution, "interpolation", the main algorithm behind creating the $p(\alpha)$ maneuver classes. The key intermediate step is nonlinear parametric programming (NLPP), in which a vector α appears explicitly in the optimization problem statement, and captures a set of *problem data* which might vary over a fixed set. As α varies, one would expect the solution p^* to vary in some smooth fashion, with λ^* and μ^* varying as well. The parametric programming field, documented extensively in [44], [45], [61], and [92], gives precise algorithms for tracing functions of the form $p^*(\alpha)$, $\lambda^*(\alpha)$, and $\mu^*(\alpha)$, starting from a known optimizing solution set $p^*(\alpha_0)$,$\lambda^*(\alpha_0)$,$\mu^*(\alpha_0)$, for some $\alpha = \alpha_0$. When α is viewed as a varying boundary condition for the trajectory generation problem, the NLPP appears as the finite-dimensional analog of neighboring extremal control [26, 71], generating a set of optimal maneuvers over a range of initial or final conditions.

This section outlines the main points of parametric programming, leaving the (considerable) algorithmic details to the aforementioned references. Instead, once the main mathematical tools of parametrized solutions and solution pathfollowing methods are introduced, the parametric program for optimality will be significantly relaxed by discarding the objective function and Lagrange multipliers. This simplification drastically reduces the problem dimension and makes accessible the general optimal *and* suboptimal trajectory space, all while retaining the ability to functionally describe solution sets.

2.2.1 Problem Formulation

To define the general parametric programming problem, introduce a vector of variable parameters $\alpha \in R^q$ to each of the nonlinear functions in the original Program (2.8). Specifically, redefine the objective as the map $f : R^n \times R^q \to R$, the equality constraints as the map $h : R^n \times R^q \to R^m$, and the inequality constraints as the map $g : R^n \times R^q \to R^r$. The optimization problem then assumes the parametric formulation

$$\text{subject to} \quad \begin{array}{l} \min\limits_{p} f(p, \alpha) \\ h(p, \alpha) = 0 \\ g(p, \alpha) \leq 0 \end{array} . \tag{2.22}$$

Note that parametric programs are useful as trajectory generators, but also find application in chemical equilibrium problems [19], chemical process dynamics [135], structures and loading problems [92], inventory management [44], and as feasibility analysis tools for on-line convex optimization algorithms [98].

2.2.2 Revised Optimality Conditions

Parametric programming algorithms work by tracing solutions of the first-order optimality conditions, so now reconsider the relations of Section 2.1.2. Since the objective and constraint sets contain an α argument, the Lagrangian L must as well, so define the parametrized Lagrangian $L : R^n \times R^m \times R^r \times R^q \rightarrow R$ as

$$L(p, \lambda, \mu, \alpha) = f(p, \alpha) + \lambda^T h(p, \alpha) + \mu^T g(p, \alpha). \tag{2.23}$$

The Karush-Kuhn-Tucker first-order necessary conditions for p^* to be a local extremum are still the $n + m + r$ equations

$$\begin{bmatrix} \nabla_p L(p^*, \lambda^*, \mu^*, \alpha) \\ h(p^*, \alpha) \\ M(\mu^*) g(p^*, \alpha) \end{bmatrix} = 0, \tag{2.24}$$

but now containing the α vector. As before, the inequality Lagrange multipliers must obey

$$\begin{aligned} \mu_j^* &\geq 0 \quad \forall\ j \\ \mu_j^* &= 0 \quad \forall\ j \notin J_0. \end{aligned} \tag{2.25}$$

If p^* is in fact a local minimum, then

$$y^T \cdot \nabla_{xx}^2 L(p^*, \lambda^*, \mu^*, \alpha) \cdot y \geq 0 \tag{2.26}$$

for any nonzero $y \in R^n$ such that

$$\begin{aligned} D_p h_i(p^*, \alpha) \cdot y &= 0 \quad \forall\ i = 1, \ldots, m \\ D_p g_j(p^*, \alpha) \cdot y &= 0 \quad \forall\ j \in J_0, \end{aligned} \tag{2.27}$$

as before.

2.2.3 Implicitly-Defined Solutions

Equation (2.24) is a set of $n + m + r$ scalar equations in $n + m + r + q$ unknowns. It is reasonable to expect that given the q components of α it is possible to solve for the remaining $n + m + r$ variables, accounting for Equations (2.25). Indeed, this expectation is the basis behind most NLP solvers in the previous, nonparametric case.

Even further, under certain conditions it is possible to exactly trace solution sets $p^*(\alpha)$, $\lambda^*(\alpha)$, and $\mu^*(\alpha)$, starting from a single known solution and using α as a

coordinate variable. Functionally solving for p^*, λ^*, and μ^* in terms of α achieves a (large) dimensionality reduction and avoids the need to resolve Program (2.22) for new instances of α. The mathematical basis for this approach is the *implicit function theorem*. Consider a function $F : R^{N_w} \times R^{N_\alpha} \to R^{N_w}$ satisfying $F(w, \alpha) = 0$. The theorem statement is:

> **Implicit Function Theorem ([104, 139])** Let A be an open set in $R^{N_w} \times R^{N_\alpha}$ and let $F : A \to R^{N_w}$ be of class C^l. Write F in the form $F(w, \alpha)$ with $w \in R^{N_w}$ and $\alpha \in R^{N_\alpha}$. Suppose that (w_0, α_0) is a point in A such that $F(w_0, \alpha_0) = 0$ and $\det D_w(w_0, \alpha_0) \neq 0$. There exists a neighborhood B of α_0 in R^{N_α} and a unique function $g : B \to R^{N_w}$ of class C^l such that $g(\alpha_0) = w_0$ and $F(g(\alpha), \alpha) = 0$ for all $\alpha \in B$.

The theorem provides a very useful tool for dealing with systems of nonlinear equations as well as being the basis for implicit differentiation, as seen in the relation $D_\alpha w = -(D_w F)^{-1} D_\alpha F$.

Now view Equation (2.24) through the lens of the theorem, defining the composite vector $w \equiv [p^T \ \lambda^T \ \mu^T]^T$ with $N_w = n + m + r$, and letting $N_\alpha = q$ with the result:

$$F(\underbrace{p^*, \lambda^*, \mu^*}_{w^*}, \alpha) = \begin{bmatrix} \nabla_p L(p^*, \lambda^*, \mu^*, \alpha) \\ h(p^*, \alpha) \\ M(\mu^*) g(p^*, \alpha) \end{bmatrix} = 0 \tag{2.28}$$

Under the conditions of the theorem, if $\det D_w F(w, \alpha) \neq 0$, there is a implicit function $w^*(\alpha)$ defining a parametrized family of optimal trajectories.

Before outlining the specific algorithm used to trace $w^*(\alpha)$. It is helpful to modify Equation (2.24) and the function F to ensure numeric stability, following the practices of reference [92]. First, recall that there are only r_0 active inequality constraints, where r_0 depends on p at any given instant. It is therefore possible to economize the first-order conditions of Equation (2.28). By dropping the inactive inequality constraints from $g(p^*, \alpha)$, the vector equation is reducible to $n + m + r_0$ components. Note that the inactive constraint multipliers are zero from Equations (2.25), so corresponding elements of Equation (2.24) were already trivially satisfied. For the active inequalities, retain the corresponding rows of $g(p^*, \alpha)$ but with the μ_j^* multipliers divided out. These active constraint multipliers will still appear in the Lagrangian and contribute to the first-order optimality condition.

The second modification is to introduce a new multiplier $\mu_0^* > 0$ in the objective function, leading to a modified Lagrangian:

$$\overline{L}(p, \lambda, \mu, \alpha) = \mu_0 f(p, \alpha) + \sum_{i=1}^{m} \lambda_i h_i(p, \alpha) + \sum_{j=1}^{r} \mu_j g_j(p, \alpha), \tag{2.29}$$

where the vector μ is now of the form $\mu = [\mu_0, \ldots, \mu_r]^T \in R^{r_0+1}$ and the active set J_0 has the expanded definition $J_0 = \{0, J_{0,1}, \ldots, J_{0,r_0}\}$. The addition of μ_0 formally transforms the KKT system to a so-called Fritz John optimality condition [14], which among other subtleties, removes the need for p^* to be a regular point of the nonlinear

program at every instant. This consideration is important for the technical practice of detailed parametric programming algorithms.

The introduction of μ_0^* disturbs the balance between the number of equations and number of unknowns, so an additional normalizing constraint of the form $\lambda^T\lambda+\mu^T\mu-\beta^2=0$ restores the balance and keeps the multipliers real and bounded during path following; the constant β is a user-chosen sufficiently large, fixed positive number. Note that this constraint (as well as the Lagrange multipliers) will not figure into the maneuvering framework of the next section but is included here for completeness and to motivate the eventual algorithm simplifications.

With these two alterations, the parametrized necessary conditions take the form of a nonlinear equation set $\overline{F}: R^{n+m+r_0+1}\times R^q \rightarrow R^{n+m+r_0+1}$ given by

$$\overline{F}(\underbrace{p^*,\lambda^*,\mu^*}_{w^*},\alpha)=\begin{bmatrix}\nabla_p\overline{L}(p^*,\lambda^*,\mu^*,\alpha)\\ h(p^*,\alpha)\\ g_{J_0}(p^*,\alpha)\\ \lambda^{*T}\lambda^*+\mu^{*T}\mu^*-\beta^2\end{bmatrix}=0. \tag{2.30}$$

When coupled with the multiplier conditions

$$\begin{aligned}\mu_j^* &\geq 0 \quad \forall\ j\\ \mu_j^* &= 0 \quad \forall\ j\notin J_0,\end{aligned} \tag{2.31}$$

Equation (2.30) satisfies the hypotheses of the implicit function theorem when $D_w\overline{F}$ has rank $n+m+r_0+1$.

2.2.4 Continuation Methods

Given a parametrized nonlinear equation set such as that in Equation (2.30), the continuation and solution path-following methods provide specific algorithms for tracing curves of the form $w^*(\alpha)$ (recall that $p^*(\alpha)$, the set of extremal trajectories, is a component of w^*). The general continuation problem is treated rigorously in references [3], [91], [92], and [124], so only the basic ideas necessary for trajectory parametrization are mentioned here.

The main continuation approaches generally fall into one of three categories: integration methods, predictor-corrector methods, and piecewise linear methods. See Figure 2-1 for a simple illustration of the integration and predictor-corrector variants for a function of the form $F(w,\alpha)=0$. All three methods start from a known solution $F(w_0,\alpha_0)=0$ but differ in how to proceed in finding neighboring solutions.

Integration methods trace curves $w(\alpha)$, accumulating differential changes in w by essentially numerically integrating both sides of the equation $D_\alpha w=-(D_wF)^{-1}D_\alpha F$. The key benefit is knowledge of the entire curve $w(\alpha)$ but with a cost of possibly slow progress along the solution arc.

Predictor-corrector (PC) methods exploit the contractive properties of solution neighborhoods for the equation $F(w,\alpha)=0$, making larger jumps along the tangent to the solution surface and then correcting back with iterative Newton methods. This

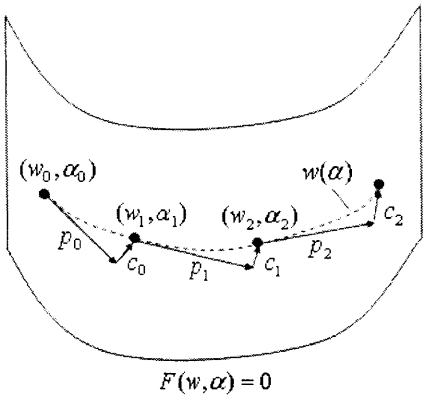


Figure 2-1: Integration and predictor-corrector methods for the curve $F(w, \alpha) = 0$. Integral methods track the entire parametrized curve to obtain $w(\alpha)$. Predictor-corrector algorithms make jumps from one solution point (w_k, α_k) to the next (w_{k+1}, α_{k+1}) with a tangent prediction step p_k and a (typically orthogonal) correction step c_k.

approach quickly moves to distant solution points, determining only a few intermediate feasible solutions. Many of the details of PC methods revolve around computing the tangent direction and making the appropriate prediction step-size. This method is best suited when large variations in α are required and when there is little chance of encountering singularities in $D_w F$.

Finally, piecewise-linear methods take the approach of not actually working with exact solutions, but instead riding along triangularized approximations of the solution manifold. Corrections back to the surface are only made near the final goal point.

In the context of parametric programming, PC methods are quite common and the following subsection outlines at a very high level how to apply them to Equations (2.30) and (2.31). Only details relevant for either appreciating the subtleties of full-scale parametric programming or for deriving a method for more general maneuver parametrization are presented.

2.2.5 Parametric Programming Outline

This subsection gives a rather coarse outline of the full parametric programming algorithm incorporating the predictor-corrector path-following technique. Although not fully apparent here, the details can become quite involved, as the constraint set can change, the solution path may bifurcate into multiple solutions of differing types (or collapse into single solutions), and/or the solution type might change from minimum to maximum without bifurcation. In contrast, the approach taken in Section 2.3 will by-pass these nuances by dropping the objective function explicitly, and using two known solutions, instead of one, to define a unique path-following direction. The benefits include: an elimination of the need to trace the Lagrange multipliers (a reduction in problem dimension by roughly half); the ability to work with suboptimal trajectories if desired, including those obtained from motion capture; and a signifi-

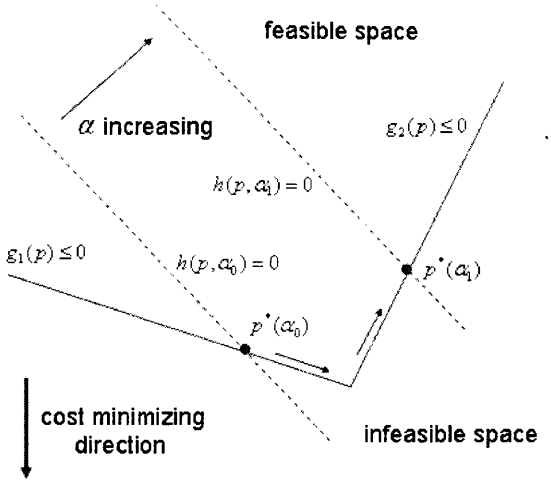


Figure 2-2: Constraint switching during parametric solution of a linear programming problem. As α increases from α_0 to α_1, the solution point must switch active constraints from g_1 to g_2. The parametrized equality constraint is $h(p, \alpha) = 0$.

cantly simpler algorithm, since there is no need to check for solution-type changes and bifurcations.

In the present section, the optimality-based parametric program method is highlighted for two reasons. First, it introduces several of the methods required in Section 2.3 and obviates the simplifications made there. Second, NLPP is a completely viable method of parametrizing *extremal* trajectories and provides a finite-dimensional alternative to neighboring extremal control (NEC).

References discussing theory and specific, implementable algorithms include [44], [45], [61], [92], and [123]. Analytical discussions concerning the detection, nature, and classification of solution bifurcation points appear in references [92], [118], and [144]. More theoretical discussions of the nature of deformable nonlinear programs and their general structure are found in [77], [82], and [126].

Sensitivity and Stability

The terms sensitivity and stability refer, respectively, to the problems of tracing the local behavior of the solution curve $w(\alpha)$, and of determining if the constraint set and/or qualitative solution type (minimum, maximum, saddle) is changing. This distinction can be seen by considering the standard linear programming polytope, as shown in Figure 2-2. Here, α parametrizes an equality constraint $h(p, \alpha) = 0$. As α increases marginally from α_0, the solution point p^* moves along the currently active constraint boundaries (sensitivity). Once α increases substantially, up to and above α_1, then the solution must switch hyperplanes from inequality constraint $g_1(p) \le 0$ to $g_2(p) \le 0$ and now proceed in new direction (stability). Recall that such constraint switchings are integral to the simplex method [141], although the matter can become significantly more complicated in the general nonlinear case.

Predictor-Corrector Iterates

The basic steps of the PC method are as follows. Define the *combined* solution-parameter vector $v \in R^{n+m+r_0+1+q}$ according to

$$v = \begin{bmatrix} w \\ \alpha \end{bmatrix} \tag{2.32}$$

and let v_k denote a feasible v at an iteration k. The basic predictor step is to find a direction T_k approximating the solution surface to first-order, that is, a vector satisfying

$$D_v \overline{F}(v_k) \cdot T_k = 0. \tag{2.33}$$

Note that the null space of D_v will have dimension q but that exact direction and orientation of T_k in R^q is uniquely determined by the desired vector change in α, also in R^q. The vector T_k then forms the basis of a first-order Euler prediction step of the form

$$v_k^p = v_k + \Delta s_k T_k, \tag{2.34}$$

where Δs_k is a step-size variable, typically chosen by comparing with the desired variation in α and the local behavior of the Equations (2.30).

The correction step back to the feasible surface solves the equation system

$$G_k(v) = \begin{bmatrix} \overline{F}(v) \\ N_k(v) \end{bmatrix} = 0 \tag{2.35}$$

using an iterative Newton root-finding technique. Here, $N_k(v)$ is an added constraint enforcing orthogonality between the prediction and correction steps and typically takes the form

$$N_k(v) = (v - v_k^p)^T T_k. \tag{2.36}$$

The Newton correction proceeds by the standard iterations

$$v_k^{c,i+1} = v_k^{c,i} - (D_v G(v_k^{c,i}))^{-1} G(v_k^{c,i}) \tag{2.37}$$

beginning with $v_k^{c,1} = v_k^p$.

Singularity Detection

Iteration of the predictor-corrector steps as presented is a fairly simple matter. However, recall that the implicit function theorem requires that $D_w \overline{F}$ be nonsingular, a condition frequently violated in nonlinear parametric programs. Any point w^* where $D_w \overline{F}$ loses rank will be called a *singular point.* These points occur for three primary reasons, which each have geometric interpretations from an optimization viewpoint. First, from Equation (2.30), the Jacobian of $\overline{F}$ with respect to w is

$$D_w\overline{F} = \begin{bmatrix} \nabla^2_{pp}\bar{L} & \nabla_p h_1 & \cdots & \nabla_p h_m & \nabla_p f & \nabla_p g_{J_{0,1}} & \cdots & \nabla_p g_{J_{0,r_0}} \\ D_p h_1 & & 0 & & 0 & & 0 & \\ \vdots & & \vdots & & \vdots & & \vdots & \\ D_p h_m & & 0 & & 0 & & 0 & \\ D_p g_{J_{0,1}} & & 0 & & 0 & & 0 & \\ \vdots & & \vdots & & \vdots & & \vdots & \\ D_p g_{J_{0,r_0}} & & 0 & & 0 & & 0 & \\ 0 & 2\lambda_1 & \cdots & 2\lambda_m & 2\mu_0 & 2\mu_{J_{0,1}} & \cdots & 2\mu_{J_{0,r_0}} \end{bmatrix}, \quad (2.38)$$

and helps to illustrate the three conditions for $D_w\overline{F}$ to lose rank, as proved in [92].

The first cause of singularity is a loss of strict complementarity, in that either a previously inactive constraint $g_j(p,\alpha)$, where $j \notin J_0$, becomes active, *or* a multiplier for a currently active constraint $g_j(p,\alpha)$, where $j \in J_0$, vanishes. In these cases, the matrix in Equation (2.38) loses rank and the continuation algorithm must modify the active constraint set J_0.

The second possibility is a loss of the so-called linear independence constraint qualification (LICQ), meaning that the active constraint derivative vectors $\nabla_p h_1, \cdots, \nabla_p h_m, \nabla_p g_{J_{0,1}}, \cdots, \nabla_p g_{J_{0,r_0}}$ have become linearly dependent and/or the objective multiplier μ_0 has vanished. In such settings, a solution bifurcation occurs and the new candidate path-following directions must be classified by type and a particular direction selected. Note that a practical discussion of constraint qualifications can be found [14] while [112] gives an extensive treatment of the subject.

Finally, the third cause of singularity is a change in the inertia of the Hessian matrix $\nabla^2_{pp}\bar{L}$ along the feasible tangent directions. The matrix inertia for a symmetric matrix is defined as the number of positive, negative, and zero eigenvalues [60]. As with the second condition, an inertia change implies a bifurcation *or* a change in solution type (transition from a local valley to a local peak or local saddle point).

Detailed parametric programming algorithms must search for these various singularities while performing the predictor-corrector steps of Equations (2.33) through (2.37). Although it is possible to define iterative algorithms that perform these tasks, the mathematics can become quite cumbersome and the resulting parametrized solution arcs must remain extremal by construction.

For the remainder of this thesis, the strict optimality conditions are dropped for simplicity, greater flexibility, and real-time implementation. This relaxation allows the design of very simple algorithms for finding parametrized maneuver sets. The removal of the optimality conditions provides extra design freedoms to the user, and as will be shown in Chapter 3, there do exist cases where optimality is retained, even if not explicitly required.

2.3 Interpolation of Feasible Trajectories

Fortunately, it is possible to create continuously parametrized trajectory families of the type $p(\alpha)$ without resorting to the full rigor of nonlinear parametric programming. A drastic simplification arises by dropping the first-order optimality conditions of Equations (2.30) and (2.31) and replacing them with a simple set of parametrized feasibility conditions, namely $h(p, \alpha) = 0$ and $g(p) \leq 0$. The benefits are a reduction in the problem size by roughly half (since there is no need to trace the paths $\lambda(\alpha)$ and $\mu(\alpha)$), a drastic algorithmic simplification since there is no need to test for bifurcation and solution type changes, and the freedom to work with *any* general feasible p, including those arising from maneuver motion capture. These simplifications enable useful on-line trajectory generation algorithms and give the engineer greater flexibility in designing motion characteristics, since up to $n - m$ extra trajectory degrees-of-freedom arise from the relaxation of strict optimality.

Whereas the specific path $p^*(\alpha)$ originating from a known solution was uniquely determined by the optimality conditions in parametric programming, now there is no unique direction. Instead, the method here is based on knowledge of *two* known solutions that together define the endpoints of a path $p(\alpha)$. Tremendous freedom is granted to the engineer in the selection in the bounding "example" trajectories. If a set of agile motions is desired, then the user should provide fast vehicle example trajectories. If the goal is to observe and generalize maneuvers observed in human-piloted demonstration, then two (or more) sets of motion capture data can be processed into trajectory feasible points and used as known solutions (see Figure 2-3). Since the feasible path connecting the two examples will result from a simple projection operation seeking the most direct arc between the known solutions, the resulting family $p(\alpha)$ tends to retain the characteristics of the examples. The strengths of nonlinear programming may now be dedicated to creating the example trajectories, while a much simpler interpolation method generates the continuous maneuver family $p(\alpha)$.

Before proceeding to the interpolation algorithm, so-called because it finds dynamically feasible trajectories that are in some sense "between" the endpoint examples, it is worthwhile to restate the feasibility conditions. In most cases of interest, the variable maneuver descriptors α specify the initial and/or final boundary states of a vehicle, and thus occur in the equality constraints [16, 26, 68, 102]. Therefore, consider the feasible space to be defined by the maps $h : R^n \times R^{N_\alpha} \rightarrow R^m$ and $g : R^n \rightarrow R^r$, giving the basic feasibility relations:

$$\begin{aligned} h(p, \alpha) &= 0 \\ g(p) &\leq 0. \end{aligned} \tag{2.39}$$

2.3.1 Coordinate Chart for Maneuvers

One may think of α as giving the intuitive "coordinates" of a maneuver, such as initial and final velocities, altitude gain, heading-change, etc., while $p(\alpha)$ captures the entire trajectory. Recall that p is a finite-space descriptor of the continuous time system behavior $z(t; p)$.

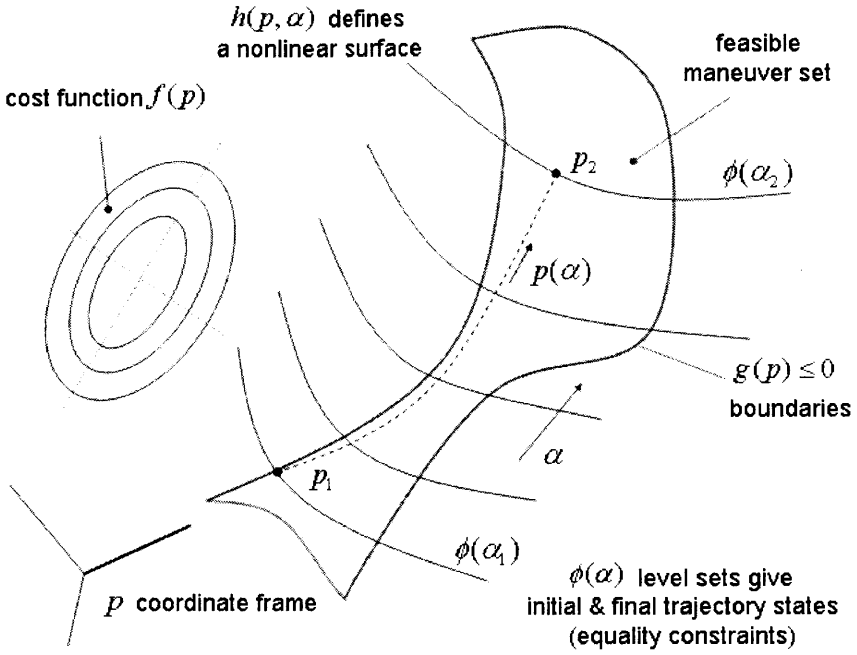


Figure 2-3: The general maneuver space. Feasible maneuvers p_1 and p_2 can come from any source, including nonlinear programming or motion capture. The parametrized family of feasible maneuvers based on these examples is given by the curve $p(\alpha)$.

In general, α may contain N_i initial coordinates, N_f final coordinates, and possibly even a coordinate dictating the time duration of the maneuver, so that $N_\alpha \equiv q = N_i + N_f + 1$. This general α takes the form

$$\alpha = [\alpha_i^1 \ \cdots \ \alpha_i^{N_i} \ \alpha_f^1 \ \cdots \ \alpha_f^{N_f} \ \alpha_{t_f}]^T, \tag{2.40}$$

Note however, that one should only include as many components in α as there are desired degrees of variability in the maneuver family: the fewer components of α, the greater the dimensionality reduction.

When it is necessary to work with multi-component α vectors, it is always possible to reduce α again to a scalar variable using some map of the form $\gamma : R \to R^{N_\alpha}$ where $\alpha = \gamma(\sigma)$ and γ is an appropriate, user-selected smooth function. Use of such dimensionality reducing γ functions is common practice in multiparametric nonlinear continuation [92].

2.3.2 Known Feasible Motions

As stated previously, the idea of trajectory interpolation is to replace parametric optimality conditions with knowledge of multiple feasible maneuvers. These motions may be optimal, suboptimal, derived from motion capture, or come from other sources. However, in the notation of the feasible model of Equations (2.39), all feasible ma-

neuvers take the form of a vector $v \in R^{n+N_\alpha}$ defined simply as

$$v = \begin{bmatrix} p \\ \alpha \end{bmatrix}. \tag{2.41}$$

When creating a parametrized maneuver family $p(\alpha)$, the known feasibles are motions of the *same type* but satisfying differing boundary conditions α. Therefore, a set of example motions for a given maneuver type exist as discrete vectors

$$v_k = \begin{bmatrix} p_k \\ \alpha_k \end{bmatrix} \qquad k = 1, 2, \ldots \tag{2.42}$$

where it is understood that the p_k describe similar trajectories but with variations due to the different boundary conditions α_k. For example, one might think of a set of aggressive heading-change maneuvers for a fixed-wing aircraft, but where $\alpha = \Delta\phi$ is used to indicate the specific heading change value.

2.3.3 Parametrized Maneuver Formulation

Given the feasible space of Equation (2.39), the equality constraint vector $h(p, \alpha)$ generally takes the form

$$h(p, \alpha) = \begin{bmatrix} h_e(p) \\ h_{bc}(p, \alpha) \end{bmatrix}. \tag{2.43}$$

The partition $h_e : R^n \to R^{N_{h_e}}$ gives equality constraints not involved in specifying boundary conditions. Instead, these components ensure that the system nonlinear equations of motion $\dot{x} = f(x, u)$ are satisfied and/or enforce holonomic and nonholonomic path constraints. Note that these constraints often arise in transcription (collocation) formulations when it is not possible to invert the system dynamics or easily forward integrate them. In such cases, p describes both state and input components of $z(t; p)$, with

$$p = \begin{bmatrix} p_x \\ p_u \end{bmatrix}, \tag{2.44}$$

requiring that the free state and input parameters be made consistent through discrete approximations of the equations of motion [16, 68]. In other cases, such as that seen in Chapter 4, the vector p describes more system states than the number of inputs. Then $h_e(p)$ must enforce consistency between the state signals via a discrete sampling of a modified differential equations of motion. Finally, $h_e(p)$ might also enforce holonomic constraints, such as fixed state arcs, or nonholonomic path equality constraints, such as zero sideslip conditions for aircraft and rolling wheel kinematics for ground robots.

The second component in Equation (2.43) is the boundary condition vector $h_{bc} : R^n \times R^{N_\alpha} \to R^{N_{h_{bc}}}$, dictating the maneuver initial and final conditions. Although the number of possible forms of h_{bc} is too large to enumerate here, it is useful to consider an example case with $\alpha \in R^2$ and of the form $\alpha = [\alpha_i \;\; \alpha_f]^T$. The h_{bc} partition then

appears as

$$h_{bc}(p,\alpha) = \begin{bmatrix} h_{bc}^0(p) \\ h_i(p,\alpha_i) \\ h_f(p,\alpha_f) \end{bmatrix}. \tag{2.45}$$

The function $h_{bc}^0 : R^n \to R^{N_{h_{bc}^0}}$ enforces boundary conditions that do not depend on the particular values of α_i and α_f. For example, these components might enforce altitude conditions when the α's are used to designate speed and/or direction.

The remaining components take the form $h_i(p,\alpha_i) = \psi_i(p) - \phi_i(\alpha_i)$ and $h_f(p,\alpha_f) = \psi_f(p) - \phi_f(\alpha_f)$, where all vectors have the consistent numbers of components. For all but the simplest systems, the functions h_i, ψ_i, and ϕ_i are vector-valued, even if $\alpha_i \in R$ (and similarly for the terminal functions h_f, ψ_f, and ϕ_f). For example, if α_i is an initial helicopter trim velocity, it is necessary to enforce steady values of speed, vehicle pitch, rotor pitch, and possibly all four of the control inputs, all depending on the scalar α_i. The mapping $\psi_i : R^n \to R^{m_i}$ extracts these quantities from a candidate p. The function $\phi_i : R \to R^{m_i}$ maps the specific initial coordinate α_i to the required physical settings, often using trim relations derived from a steady-state analysis of $\dot{x} = f(x,u)$. Note that both ϕ_i and ψ_i are nonlinear in general. Chapter 4 gives a detailed example of these relations in practice for a three degree-of-freedom helicopter.

As mentioned previously in Section 2.3.1, another possible option, is to control the time-duration of a maneuver. One then sets $\alpha = \alpha_{t_f} \in R$ so that $h(p,\alpha)$ takes the form

$$h(p,\alpha) = \begin{bmatrix} h_e(p) \\ h_{t_f}(p,\alpha_{t_f}) \end{bmatrix}, \tag{2.46}$$

with $h_{t_f}(p,\alpha_{t_f}) = \psi_{t_f}(p) - \phi_{t_f}(\alpha_{t_f})$. The ψ and ϕ functions play similar roles as in the initial and final condition case, with ψ extracting the maneuver duration from p and ϕ converting α_{t_f} into a trajectory duration. These operations allow a maneuver to be flown at variable speed, depending on scheduling or tactical requirements.

For now, the inequality constraints are assumed to be α-invariant and thus simply of the form $g(p)$. As discussed in Section 2.1, these inequalities typically enforce path, state, and control limits, such as those in Inequalities (2.6) and (2.7).

2.3.4 Interpolation with Equality Constraints Only

This section describes the specific trajectory interpolation method, first for the case of equality constraints only; Section 2.3.5 then completes the story by adding inequalities. It is useful to begin with the equality constraint-only case since it illustrates the simple projection scheme for defining feasible directions and is itself a useful algorithm in practice.

The maneuver model reduces to the simple expression $h(p,\alpha) = 0$ with $h \in R^m$ and $p \in R^n$. For now, assume that $\alpha \in R$ (or equivalently that there is a map $\alpha = \gamma(\sigma)$ that replaces α by a scalar σ). In general, this maneuver space has $n - m$ available degrees-of-freedom and it is not possible to apply the implicit function theorem to determine a unique family $p(\alpha)$.

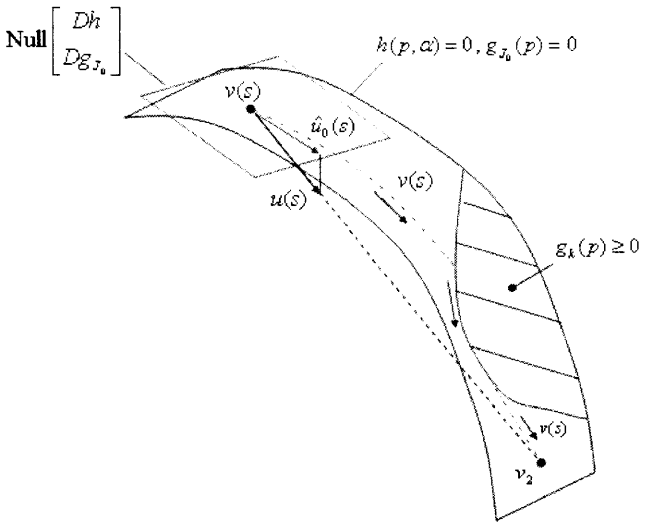


Figure 2-4: Projection method for choosing a unique feasible direction. The difference vector u is projected onto the local tangent hyperplane to obtain $\hat{u}$, a tangent vector for the feasible path $p(\alpha)$. The path must consider any inequality constraints $g_k(p)$ occurring between v_1 and v_2.

Using the notation of Section 2.3.2, let v_1 denote a known maneuver satisfying $h(p, \alpha) = 0$. Given another feasible v_2, it is possible to define a unique feasible direction "towards" v_2 from v_1. To do so, simply take $u = v_2 - v_1$ and project it onto the constraint tangent plane at v_1.

Specifically, $D_v h(p_1, \alpha_1) \in R^{m \times (n+1)}$ must have a null space of at least dimension $n + 1 - m$. Let $Z = [z_1, \ldots, z_{n+1-m}]$ with $z_i \in R^{n+1}$ denote a basis for $\text{Null}(D_v h)$ and take the projection of u onto Z, expressed as $\hat{u} = \text{Proj}_Z u$. Then $\hat{u}$ is a first-order feasible direction from v_1 towards v_2, and by extension from trajectory p_1 (with boundary condition α_1) towards trajectory p_2 (with boundary condition α_2). See Figure 2-4, assuming for the time being that the indicated inequalities do not yet exist. All that remains to create a maneuver family $p(\alpha)$ is to trace this first-order direction towards v_2, differentially updating the projected direction while moving along the path.

Letting s denote the arc length in R^{n+1} space along this feasible path, the resulting path has the form $v(s) = [p(s); \alpha(s)]$. Generally, the example trajectories v_1 and v_2 are taken to be of the same type, so that the projection operation is well-posed and the arc $v(s)$ actually reaches v_2 without getting stuck at intermediate feasible points. Finding a general set of conditions under which the trajectory set is in fact "interpolable" without problems is a hard proposition. However, assuming in engineering practice that v_1 and v_2 are chosen so that $v(s)$ can reach v_2, then $\alpha(s)$ is a monotonic function, implying that the inverse map $s(\alpha)$ exists, and thus giving $p(s) = p(s(\alpha)) = p(\alpha)$, as desired. Noting that after projection, $\hat{u}_{n+1} = d\alpha/ds$, and thus a multiplication of $\hat{u}$ by $\hat{u}_{n+1}^{-1}$ will ensure that $s = \alpha$ identically and therefore $p(s) = p(\alpha)$ follows

automatically.

A slightly different viewpoint is to write $h(v(s)) = h(p(s), \alpha(s)) = 0$ and note that

$$\begin{aligned} \frac{dh}{ds} &= \frac{\partial h}{\partial p}\frac{dp}{ds} + \frac{\partial h}{\partial \alpha}\frac{d\alpha}{ds} \\ &= \begin{bmatrix} \frac{\partial h}{\partial p} & \frac{\partial h}{\partial \alpha} \end{bmatrix} \begin{bmatrix} \frac{dp}{ds} \\ \frac{d\alpha}{ds} \end{bmatrix} \\ &= D_v h(v)\frac{dv}{ds} \\ &= 0. \end{aligned} \tag{2.47}$$

Therefore any vector in the null space of $D_v h$ is a well-defined first-order feasible direction; the obvious choice is simply to take $dv/ds = \hat{u} = \text{Proj}_Z u$. If the projection method is the standard least squares minimization of $||\hat{u} - u||_2$, then

$$\hat{u} = Z^+ u = Z(Z^T Z)^{-1} Z(v_2 - v_1). \tag{2.48}$$

Expressing these operations as an *integration* continuation method gives the following trajectory interpolation algorithm, posed as starting from $v_1 = [p_1; \alpha_1]$ and seeking a maneuver $p(\alpha_{goal})$ with $\alpha_1 \leq \alpha_{goal} \leq \alpha_2$.

Pseudo-code for Algorithm 1

∘ Obtain feasible maneuvers $v_1 = [p_1; \alpha_1]$ and $v_2 = [p_2; \alpha_2]$ of the same type but satisfying numerically different boundary conditions.

∘ Integrate $\dot{v}(s) = \hat{u}(s, v)$ with independent variable s along $[0, \alpha_{goal} - \alpha_1]$ with initial condition $v(0) = v_1$ and $\hat{u}(s, v)$ at any point s given by the following logic:

1: $u(s) = v_2 - v(s)$	direct difference
2: evaluate $D_v h(v(s))$	local equality derivatives
3: $Z =$ basis of Null$(D_v h)$	tangent space basis
4: $\hat{u} = \text{Proj}_Z u$	project difference vector
5: $\hat{u} \leftarrow \hat{u} \cdot (d\alpha/ds)^{-1}$	normalize arc length

---

Typically, the most costly operation in Algorithm 1 is the evaluation of $D_v h(v(s))$. It is preferable to compute these partial derivatives analytically instead of numerically when possible, saving effort and giving more accurate quantities. Appendix B derives tools for computing these derivatives for B-splines and general nonlinear constraint functions h and g, and will prove useful for the specific application in Chapter 4.

Fortunately, it is reasonable to only update this matrix periodically, say, at some fixed interval $\Delta\alpha$, and not at every forward integration step. The suitable interval $\Delta\alpha$ can easily be found empirically, since solution trajectories $p(\alpha)$ can be tested against the equations of motion $\dot{x} = f(x, u)$ to verify fidelity. In addition, reference [17] discusses various rank-one Broyden updating rules to further economize the computation of $D_v h$.

Finally, recalling from Section 2.2.5 that continuation methods can be accelerated by using a predictor-corrector format, the following Algorithm 2 gives a PC version of Algorithm 1. The notable difference is the inclusion of the function $G : R^{n+1} \rightarrow R^{m+1}$ given by

$$G_k(v) = G_k(p, \alpha) = \begin{bmatrix} h(p, \alpha) \\ N_k(v) \end{bmatrix} \tag{2.49}$$

with $N_k(v) = (v - v_k^p)^T \hat{u}_k$, similar to Equations (2.35) and (2.36). Now it is unnecessary to normalize the prediction vector (first-order feasible direction), since the α-value at the next iteration will follow from the Newton root-finding procedure, which again utilizes a pseudo-inverse $(D_v G)^+$ since $m < n$. Choice of step size δ_k is left to the user, although references [3], [14], [91], and [124] give numerous criteria for step size selection. The quantity ϵ is taken to be suitable small number for terminating the iterative corrector process. The predictor-corrector algorithm is as follows.

Pseudo-code for Algorithm 2

∘ Obtain feasible maneuvers $v_1 = [p_1; \alpha_1]$ and $v_2 = [p_2; \alpha_2]$ of the same type but satisfying numerically different boundary conditions.

∘ Obtain a feasible sequence $v_k = [p_k, \alpha_k]$ with $k = 1, 2, \ldots$ by iterative repetition of the following predictor-corrector steps:

1: $u_k = v_2 - v_k$	direct difference
2: evaluate $D_v h(v_k)$	local equality derivatives
3: $Z =$ basis of Null$(D_v h)$	tangent space basis
4: $\hat{u}_k = \text{Proj}_Z u_k$	project difference
5: $v_k^p = v_k + \delta s_k \cdot \hat{u}_k$	predictor point
6: set $v_k^{c,1} = v_k^p$	first corrector point
7: **while** $\|\|h(p_k^{c,i}, \alpha_k^{c,i})\|\| \geq \epsilon$	
8: $\quad v_k^{c,i+1} = v_k^{c,i} - (D_v G_k(v_k^{c,i}))^+ G_k(v_k^{c,i})$	Newton correction steps
9: **end**	
10: $v_{k+1} = v_k^{c,i+1}$	next feasible point

Now what is needed is a simple and reliable way of including inequality constraints.

2.3.5 Equality and Inequality Constraints

The addition of inequalities back into the problem returns the parametrized feasible space to the form of Equations (2.39). Now, logic must be added to the trajectory interpolation algorithm to detect inequalities that are becoming active and drop those going inactive. These steps can be challenging to add to predictor-corrector methods, since the prediction step may jump over a point where the active set changes and the active set can potentially change during each of the Newton corrector iterations.

Therefore, in the context of this thesis, it is preferred to modify the integration-based method of Algorithm 1.

For continuation purposes, define the *active set feasible space* at a point $v = [p; \alpha]$ as

$$F_0(p, \alpha) = \begin{bmatrix} h(p, \alpha) \\ g_{J_0}(p) \end{bmatrix} = 0, \qquad g(p) \leq 0 \tag{2.50}$$

where J_0 depends explicitly on the current trajectory point p. A parametrized curve $p(\alpha)$ then must obey

$$F_0(p(\alpha), \alpha) = 0, \qquad g(p) \leq 0 \qquad \forall\ \alpha_1 \leq \alpha \leq \alpha_2. \tag{2.51}$$

Now, when computing the tangent plane for projection of u, it is necessary to include the components of g_{J_0}. Components are added to g_{J_0} when a constraint $g_j(p)$ becomes active along $p(\alpha)$. An active constraint $g_j(p)$ is dropped when $\hat{u} = \text{Proj}_Z u$ no longer violates it to first-order, that is, when the $D_v g_j \cdot \hat{u} < 0$. One may interpret this condition as the feasible "line of sight" vector $\hat{u}$ to v_2 no longer crossing the tangent to g_j (see Figure 2-4). The following algorithm performs integration-based trajectory interpolation with both equality and inequality constraints.

Pseudo-code for Algorithm 3

∘ Obtain feasible maneuvers $v_1 = [p_1; \alpha_1]$ and $v_2 = [p_2; \alpha_2]$ of the same type but satisfying numerically different boundary conditions.

∘ Integrate $\dot{v}(s) = \hat{u}(s, v)$ with independent variable s along $[0, \alpha_{goal} - \alpha_1]$ with initial condition $v(0) = v_1$ and $\hat{u}(s, v)$ at any point s given by the following logic:

1:	$u(s) = v_2(s) - v(s)$	direct difference
2:	evaluate $D_v h(v(s))$	local equality derivatives
3:	$Z =$ basis of Null$(D_v h)$	tangent space basis
4:	$\hat{u}_0 = \text{Proj}_Z u$	project difference
5:	$\hat{u}_0 \leftarrow \hat{u}_0 \cdot (d\alpha/ds)^{-1}$	normalize arc length
6:	$J_1 = \{i \mid g_i(p) \geq 0\}$	test for active inequalities
7:	**if** $J_1 = \emptyset$	none active
8:	$\quad \hat{u} = \hat{u}_0$	
9:	**else**	
10:	$\quad J_2 = \{j \in J_1 \mid D_v g_j \cdot \hat{u}_0 \geq 0\}$	test 1st-order inequality behavior
11:	$\quad$ **if** $J_2 = \emptyset$	none active to 1st-order
12:	$\quad\quad \hat{u} = \hat{u}_0$	
13:	$\quad$ **else**	
14:	$\quad\quad Z' =$ basis of Null $([D_v h; D_v g_{J_2}])$	follow 1st-order active inequalities
15:	$\quad\quad \hat{u} = \text{Proj}_{Z'} u$	re-project difference
16:	$\quad\quad \hat{u} \leftarrow \hat{u} \cdot (d\alpha/ds)^{-1}$	normalize arc length
17:	$\quad$ **end**	
18:	**end**	

As before, assume that $v_1 = [p_1; \alpha_1]$ and $v_2 = [p_2; \alpha_2]$ are well-chosen maneuvers of the *same type*, but with differing boundary conditions, so the algorithm can run to completion.

Note that in Algorithm 3, the tangent plane and its basis Z now have dimension $n+1-m-r_0$, so that active inequalities reduce the degrees-of-freedom of the maneuver during projection. This fact is consistent with intuition, since each active inequality acts essentially as an equality, reducing the dimension of the local feasible space by 1. A fully active set of inequality constraints occurs when $r_0 = n - m$ and the tangent direction has dimension 1. In such cases, the projection operation $\hat{u} = \text{Proj}_Z u$ reduces to simply choosing the correct orientation on a one-dimensional curve. Any additional constraints that become active (typically for numerical reasons) are disregarded, since $D_p F_0$ is then square and the extra derivatives cannot contribute any additional rank to the projection problem.

2.3.6 The Multivariable Case

Algorithms 1, 2, and 3 as stated assume that α is of dimension 1. However Section 2.3.1 discussed the more general, and desirable, notion of using vector-valued α, allowing greater freedom over maneuver design and making the family $p(\alpha)$ more general. This subsection briefly discusses three mathematical techniques for handling a multivariable coordinate chart.

The first method is to use the algorithms exactly as given, modifying only the interpretation of the arc length variable since α now has several components. One can either choose to scale $\hat{u}$ relative to a particular component of α or simply normalize $\hat{u}$ according to $\hat{u} \leftarrow \hat{u}/||\hat{u}||$. These operations take place in line 5 of Algorithm 1 and lines 5 and 16 of Algorithm 3. With this approach, the constraint formulations are allowed to contain multiple chart components, as in Equation (2.45). Note, however, that this method will not allow the user to explicitly control the variations *between* the components of α. Since the interpolation algorithm seeks the closest projection in the feasible space of dimension $n+N_\alpha$, it does not enforce specific differential relationships between the last q components of v, making the exact chart path between v_1 and v_2 hard to predict.

A second method addresses this concern and has been mentioned in previous subsections. Reduce the vector α, with the general components of Equation (2.40), to a single parametrization by choosing a differentiable mapping, or coordinate path, of the form $\gamma : R \to R^{N_\alpha}$, with $\alpha = \gamma(\sigma)$ and $\sigma \in R$. Note that the feasible space technically is then of the form $h(p, \sigma) = 0, g(p) \leq 0$ and the symbol σ replaces α in Algorithms 1 through 3. This approach is common in the multiparametric programming literature [92] and is fairly intuitive. The interpolation can still take place in one process and use two known solutions v_1 and v_2 as guides.

The third method is to perform multiple scalar interpolations, one component of α at a time. The first interpolations would be performed to obtain the desired value of α_1, the second set to obtain the desired value of α_2, and so on. This approach therefore requires 2^{N_α} example trajectories and $2^{N_\alpha} - 1$ scalar interpolations. Although not realistic for maneuvers of high-dimension, it can be effective when N_α is 2 or 3.

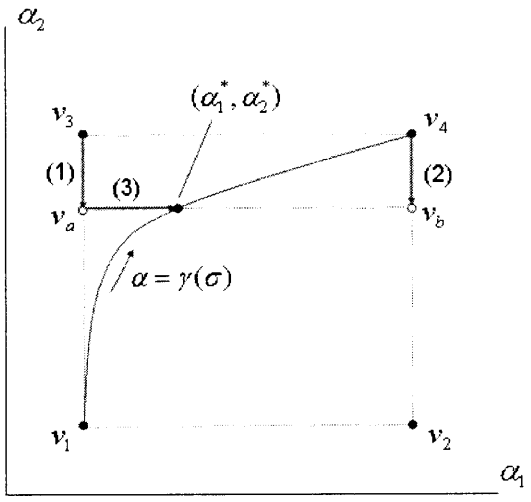


Figure 2-5: The second the third methods for handling multivariable α. The goal maneuver is at (α_1^*, α_2^*) and known maneuvers are at v_1, v_2, v_3, and v_4. The second method, uses a nonlinear mapping $\alpha = \gamma(\sigma)$ between v_1 and v_4. The third method uses three single-variable interpolations, with (1) and (2) producing intermediate maneuvers v_a and v_b and (3) then achieving (α_1^*, α_2^*).

See Figure 2-5 for an illustration of the second and third methods for treating a case with $\alpha \in R^2$. For a general multidimensional maneuver class, the most reasonable approach is to store a set of many known trajectories, exploring reasonable variations in α but employing a single $\alpha = \gamma(\sigma)$ parametrization to generate new maneuvers. Then for the algorithms, choose the maneuvers with chart variables "closest" to the desired α as the seeds v_1 and v_2 for interpolation.

2.4 Maneuver Motion Capture

The maneuver interpolation procedures in Algorithms 1, 2, and 3 use feasible maneuvers as examples in creating parametrized families $p(\alpha)$. As seen in Figure 2-3, such feasible points may arise from off-line nonlinear optimization or come by capturing real-world human-piloted vehicle motions. The notion of using demonstrated flight performance as a guide for autonomous vehicle trajectory design has been explored in [54], [55], [56], and [114] for the case of aerobatic helicopter maneuvers. These works study multiple examples of a single maneuver and infer feedback state machines for closed-loop recreation of specific motions. Inspired by those examples, this thesis takes the viewpoint of using multiple observed maneuvers over varying boundary conditions to describe a continuously parametrized set of motions, capable of generalizing the finite examples sets. Thus the maneuver classes give an effective way of exploiting known system behaviors, taking advantage of the considerable flight capabilities of human pilots.

This section gives a brief and high-level discussion of transforming flight data into feasible maneuver points of the type $v = [p^T \alpha^T]^T$. The specific procedure for a given application will be vehicle and sensor-dependent. Chapter 4 fills in several of the details, working with real experimental data and performing interpolated maneuvers in closed-loop.

For now, assume that a vehicle model $\dot{x} = f(x, u)$ and finite behavior parametrization $z(t; p)$ have been selected and that the system constraint specifications are known and given in the form of Equation (2.39). The goal of motion capture is to turn observed system behavior $z_{data}(t)$ into a point p_{data} in the parametrized trajectory space. That is, the mapping $z_{data}(t) \rightarrow p_{data}$ moves from a feasible point of the *true system* to a feasible motion of the parametrized *system model.*

First, it is useful to obtain a reasonable estimate of p by a pure data matching procedure. Tools for data fitting are common for splines, exponentials, wavelets, and other types of basis functions. The initial estimate typically takes the form

$$p_{data,0} = \arg\min_{p} \sum_{k=0}^{N} ||z_{data}(t_k) - z(t_k; p)||_{W_k} \tag{2.52}$$

where t_k indicates discrete sampled data, N is the number of data points, and W_k is a user-selected weighting function employed to highlight particular signals of interest or perform time windowing. This matching procedure often takes the form of a projection or least-squares operation [14, 141].

What remains is to find the best point in the nonlinear feasibility program of Equation (2.39), which by definition includes model knowledge. Since processing of motion capture takes place off-line, it is appropriate to solve an explicit nonlinear program, whose objective is to match the data using an error metric e. The program itself is

$$\begin{aligned} &\min_{p} e(p) \\ \text{subject to} \quad &h(p, \alpha_{data}) = 0 \\ &g(p) \leq 0 \end{aligned}, \tag{2.53}$$

where α_{data} gives the maneuver boundary conditions as observed in the data. This vector does not have to be exact since some trajectory initial and final variables may be hard to estimate. However, the closer α_{data} is to the actual data, the better the performance of the motion capture process. The error metric can attempt to match the p estimate of Equation (2.52), as in

$$e(p) = ||p - p_{data,0}||. \tag{2.54}$$

However, it is generally more effective to the use the flight data explicitly in the objective function, as with

$$e(p) = \sum_{k=0}^{N_k} ||z_{data}(t_k) - z(t_k; p)||_{W_k} \tag{2.55}$$

using $p_{data,0}$ from Equation (2.52) as an initial guess for Program (2.53). Thus the motion capture process can be performed in two phases with the same error criterion. The first step obtains an initial guess of p and the second step rigorously fits the nonlinear model. Since nonlinear programming is generally very sensitive to initial guesses, using $p_{data,0}$ from Equation (2.52) tends to be far more effective than using some manually selected estimate. Indeed, if the model fidelity is high, the optimal p^* from Program (2.53) and $p_{data,0}$ will be very similar.

2.5 Alternative Formulations

Section 2.3.1 assumed that the vector α describes only boundary condition variations and time length of the maneuver. However, there are several other problem variations where trajectory interpolation is useful for system analysis and control. The following three subsections give brief descriptions of alternative interpolation objectives.

2.5.1 Model Interpolation

The preceding discussion has assumed a *fixed* vehicle model of the form $\dot{x} = f(x, u)$. However, it is possible that trajectories are known for one system model but then the model changes, due to a hardware modification, system reconfiguration, or even a variable physical parameter, such as air density. In such cases, generalize the equations of motion to the form $\dot{x} = f(x, u, \alpha)$. Recall from the discussion of Equations (2.43) and (2.45) that the functions h_e and h_{bc} depend directly on the system model since it is necessary to maintain dynamic feasibility as well as satisfying trim boundary conditions, obtained from a steady-state analysis of the equations of motion. As such, the appropriate modifications to these equations are

$$h(p, \alpha) = \begin{bmatrix} h_e(p, \alpha) \\ h_{bc}(p, \alpha) \end{bmatrix} \qquad \text{and} \qquad h_{bc}(p, \alpha) = \begin{bmatrix} h^0_{bc}(p, \alpha) \\ h_i(p, \alpha) \\ h_f(p, \alpha) \end{bmatrix}, \tag{2.56}$$

where now α represents a physical model parameter. It assumed that the model alterations are smooth and do not affect the state and control bounds so that the inequalities remain in the form $g(p) \leq 0$. Trajectory interpolation then proceeds using the as-stated algorithms.

2.5.2 Known Disturbances

A similar situation occurs when a known disturbance of variable magnitude and/or direction is present in the system. Examples include prevailing winds, water currents, and unknown ground surface inclinations for robotic devices. In principle, a robust controller can handle some degree of uncertainty but maneuver interpolation can be used to modify the reference trajectory (including inputs), making the control task more manageable. The parameterized disturbance enters the model in the form

$\dot{x} = f(x, u, \alpha)$, just as in the preceding subsection, leading to the same modifications of h_e and h_{bc}.

2.5.3 Parametrized Inequalities

A third alternative is that of variable inequalities. One might conceive of an obstacle avoidance motion, executing a jump over a previously unknown solid object or threat. Over short time intervals, a linear controller might not be able to modify a linearized trajectory rapidly enough to avoid collision. Here, the variable inequality requires the trajectory to pass over a wall or around a threat, with α giving the amount of deviation from nominal. The feasible space notation then has the form

$$\begin{aligned} h(p) &= 0 \\ g(p, \alpha) &\leq 0 \end{aligned} \tag{2.57}$$

with known solution v_1 corresponding to the nominal, unimpeded trajectory, and with v_2 storing knowledge of a dynamically feasible but extreme avoidance maneuver. In practice, sensors detect obstacle size and location, allowing calculation of α. On-line interpolation then produces a vehicle-preserving motion somewhere between the nominal and extreme motions. Modifications to Algorithm 3 would involve skipping lines 3, 4, and 5 and setting $\hat{u}$ to 0 in line 8.

2.6 Relationship to Existing Methods

The interpolation algorithm presented in this chapter is not an optimization method. Since the goal is a real-time trajectory generator capable of describing any feasible system trajectories, the emphasis is placed on using the known examples as implicit measures of maneuver quality or usefulness. If an objective function is clear for a given class of motion, then example maneuvers can be generated off-line via nonlinear programming. If the goal is to take human-piloted motions from real-world observation and then generalize them for use by autonomous agents, then motion capture is the appropriate starting point. However, once these trajectories are cast as feasible points, then they are in some sense "equal" from the perspective of trajectory interpolation, and an explicit objective function $f(p)$ is no longer required. However, the spirit and practice of the interpolation algorithms do share some similarities optimization-specific methods that are worth noting.

Recall that Algorithms 1, 2, and 3 follow from a generalization of nonlinear parametric programming, which is itself a finite-dimensional analog of neighboring extremal control. Both NLPP and NEC start with known optimal solutions and perturb them by differentiating the first-order optimality conditions (Karush-Kuhn-Tucker conditions for NLPP, Euler-Lagrange conditions for NEC). Thus, by taking a derivative of a derivative, they are both second-order methods and must be concerned with extremum type during perturbation. The interpolation algorithm here is a first-order method, taking only single derivatives of constraint (feasibility) functions to

allow perturbations. However, note that the method presented here is a practically-motivated relaxation of both NLPP and NEC, since the vectors p and α exist in finite-dimensions while describing an infinite-dimensioned continuous time behavior $z(t;p)$.

The interpolation procedure is similar to projective gradient methods since the primary subproblem is the projection of a direction vector onto a convex subset using a second-order error metric [14]. The difference comes in that the guide direction is a difference vector $u = v_2 - v_1$ instead of a gradient, and the projection is onto a convex feasible tangent plane instead of a convex interior set.

Note that the difference vector projection accounts for the *locally* active set, making the maneuver pathfollowing similar to manifold suboptimization methods [14], which seek a minimizing point while following feasible paths along constraint boundaries. While tracing a feasible arc through the general maneuver space, active inequality constraints switch "on" and "off" depending on where they are encountered.

Similar to sequential quadratic programming, a method applicable to general nonlinear, nonconvex spaces, the constraint tangent plane is always a local approximation to the underlying nonlinear constraint surface [14, 17, 18]. A key difference lies in that sequential quadratic programming can produce infeasible iterates as it seeks to solve the first-order necessary conditions, while trajectory interpolation uses either small differential motions or correction steps to maintain a strictly feasible path.

Finally, some primal-dual interior point methods, in particular those used to trace central paths of barrier functions in semidefinite programs, utilize predictor-corrector pathfollowing methods [14, 103, 148, 156]. In those cases the central path is parametrized in terms of a barrier function homotopy variable instead of a maneuver chart vector α. However in both cases, the objective is to trace a strictly feasible path through a nonlinearly-constrained space.

Chapter 3

Trajectory Interpolation Properties

The trajectory interpolation algorithms presented in Chapter 2 provide a simple method to create continuously parametrized maneuver classes, starting from user-selected feasible example trajectories. However, there are several questions to address before the method's merits can be ascertained.

In particular, it is important to consider the consequence of dropping the optimality condition from traditional parametric programming and moving into the more general feasible space. The tangent plane projections that supplant the nonlinear Karush-Kuhn-Tucker equations were chosen to provide an intuitively simple tool for maintaining performance fidelity to the given example maneuvers. The first section of this chapter investigates the cost behavior of parametrized families $p(\alpha)$ using position-change maneuvers of a double-integrator system as an example. The lessons learned in this case study will then be applied directly to motion design for a more complex nonlinear system in Chapters 4 and 5.

The second section determines bounds on the expected number of computations required to carry out trajectory interpolation. By analyzing the individual steps of Algorithm 3 in detail, it is possible to determine the effect of problem size on the number of elementary operations required to compute specific maneuvers within a class. The results give insight into the complexity of the method and useful techniques for applying it to on-line trajectory generation. Since the interpolation algorithms are based on a numerical integration procedure with known initial conditions, it is possible to obtain specific computation bounds, in contrast to nonlinear programming algorithms where iteration solution procedures are highly dependent on initial guesses.

The third section briefly takes up the topic of continuity in the parametrized family $p(\alpha)$. Since Algorithms 1 and 3 compute this family by integral methods, intuitive but useful statements can be made about the nature of $p(\alpha)$ as a connected path between two bounding examples v_1 and v_2.

Finally, the fourth section discusses the system and constraint types for which the interpolation method is and is not applicable. The main requirement is differentiability of the maps $h(p, \alpha)$ and $g(p)$, which makes the method useful for a large number of nonlinear models commonly used to describe autonomous vehicles and robotic systems.

3.1 Optimality Gap

The integration-based methods of Section 2.3 made no explicit provision for the cost performance of the parametrized family $p(\alpha)$. The projection approach considers only the general feasible space, made up of equality and inequality constraints, and does not include the necessary conditions or knowledge of the Lagrangian function L. As such, there is no hard guarantee that the optimality gap, defined here as the difference between the cost performance of the feasible family $p(\alpha)$ and the corresponding optimal family $p^*(\alpha)$, will be zero or even remain small. (Note that the term "optimality gap" may have other meanings in different contexts [14, 148]). Fortunately, as this section will show, using examples of certain minimum-time maneuvers, the optimality gap approaches zero in some situations and can be directly controlled in others. The ability to govern the time duration of maneuvers will be a consequence of a trajectory parametrization p that *explicitly* includes the maneuver duration T. Just as important, the extra degrees-of-freedom created by dropping the Lagrange multipliers and their necessary conditions allow the user direct control over many useful behaviors of $p(\alpha)$. In particular, for a vector p with n components subject to m equality constraints, one can include up to $n - m$ additional constraints to influence the trajectory interpolation process. This ability will be crucial in Chapter 5, where a specific hybrid motion planner requires parametrized maneuver classes to have linearly-varying boundary conditions, a property unlikely along a general optimal maneuver manifold. As will be shown there, the optimal cost curve can be investigated off-line, and a suitable linear boundary condition function chosen to closely approximate optimality, allowing an intuitive trade-off between quick solution time and maneuver cost performance.

3.1.1 Problem Definition

In the present setting, consider a unit-mass double-integrator system of the form

$$\ddot{x}(t) = u(t) \tag{3.1}$$

undergoing rest-to-rest "reposition" maneuvers from the origin to a variable final position $\alpha = x_f$. The maneuvers occur over a time interval $t \in [0, t_f]$, with t_f depending on α. Dynamic equilibrium is enforced at the beginning and ending of the trajectory, so that $u(0) = u(t_f) = 0$. To keep the problem well-posed, bound the control signal between $u_{max} = \overline{u}$ and $u_{min} = -\overline{u}$, where $\overline{u}$ is a positive number.

Although simple, this system is worth considering for several reasons. First, it is possible to determine analytically the form of the optimal minimum maneuver time t_f^* as a function of α. In addition, one may determine the effect of $\overline{u}$, giving a combined optimal time relationship of the form $t_f^*(\alpha, \overline{u})$. Second, even though the equation of motion (3.1) is linear, the specific trajectory parametrization p will lead to nonlinear equality and inequality constraints, making the maneuver interpolation procedure nontrivial. Finally, autonomous vehicles operating under bounded control often behave approximately as single or double integrators when regarded as mappings

between control inputs and the outermost kinematic states. Therefore, the lessons learned in this section are directly applicable to more complex nonlinear systems.

Assuming a finite trajectory parametrization $x(t;p)$, the minimum-time repositioning problem takes the following form when expressed as a general mathematical program in continuous-time variables:

$$\begin{aligned} \min_{p} \quad & t_f \\ \text{subject to:} & \\ x(0) &= 0 \\ x(t_f) - \alpha &= 0 \\ \dot{x}(0) &= 0 \\ \dot{x}(t_f) &= 0 \\ u(0) &= 0 \\ u(t_f) &= 0 \\ -\overline{u} - u(t) &\leq 0 \\ u(t) - \overline{u} &\leq 0. \end{aligned} \tag{3.2}$$

Here, the inclusion of α allows a variable final position; the $\dot{x}$ equality constraints enforce zero initial and final velocity; the u equality constraints enforce initial and final equilibrium; and the inequalities maintain bounded control.

Previous work in the optimal control field shows that the minimizing trajectory solution for Problem (3.2) follows a bang-bang strategy: applying a saturating control to accelerate the system for the first half of the motion, and then decelerating to a rest state using a saturating control of opposite magnitude [26, 51]. Given the existence of such an optimal solution for a *nominal* final position $x_{f,n}$, it is possible to infer the form of the general optimal maneuver time as a function of α, that is, $t_f^*(\alpha)$. Although this relationship has been derived in other works, it is worth approaching here from the general viewpoint of this thesis: that given one known solution of a trajectory generation problem, it is possible to uncover other related solutions without having to reoptimize from scratch. This is the same principle behind parametric programming, where knowledge of a single optimal solution made possible an entire parametrized optimal family, as well as behind trajectory interpolation in Chapter 2, where two known solutions are enough to define a continuous feasible family.

3.1.2 Optimal Cost Functions

Seeking an optimal cost function $t_f^*(\alpha)$, begin with a single optimal solution to Problem (3.2) for $\alpha = x_{f,n}$. The "n" subscript denotes the "nominal" problem and let $x_n(t)$ and $u_n(t)$ denote the optimal state and control solutions to this problem, respectively. The minimizing time in the nominal case is denoted as $t_{f,n}^*$.

The velocity of the optimal trajectory at any time s satisfying $0 \leq s \leq t_{f,n}^*$ follows

from the control as

$$\dot{x}_n(s) = \dot{x}_n(0) + \int_0^s u_n(\tau)d\tau = \int_0^s u_n(\tau)d\tau \tag{3.3}$$

where $\dot{x}_n(0) = 0$ and τ is dummy variable of integration. The final (nominal) position follows similarly as

$$\begin{aligned} x_{f,n} &= x_n(t^*_{f,n}) = x_n(0) + \int_0^{t^*_{f,n}} \dot{x}_n(s)ds \\ &= \int_0^{t^*_{f,n}} \int_0^s u_n(\tau)d\tau ds \end{aligned} \tag{3.4}$$

where $x_n(0) = 0$.

Now consider the case of a reposition from the origin to a *general* final position x_f, expressed as a real, positive multiple k of the nominal final position:

$$x_f = kx_{f,n}. \tag{3.5}$$

Now, without loss of generality, the optimal time t^*_f to reach equilibrium at a general x_f depends on k and is related to $t^*_{f,n}$ in terms of a premultiplying function $c(k)$ by

$$t^*_f = c(k)t^*_{f,n}, \tag{3.6}$$

where $c(k) > 1$ if $k > 1$, and $c(k) < 1$ if $k < 1$. To reach the final position x_f subject to the same control bounds, the strategy will be identical to the nominal bang-bang case, but scaled in time by a factor dependent on $c(k)$. That is, the control will still saturate to accelerate, switch at the midpoint, and then reverse-saturate to decelerate, so that the general control $u(t)$ follows

$$u(t) = u_n(t/c(k)) \tag{3.7}$$

in accordance with Equation (3.6). This relation gives the boundary values $u(0) = u_n(0) = 0$ and $u(t^*_f) = u_n(t^*_f/c(k)) = u_n(t^*_{f,n}) = 0$, corresponding to equilibrium at the initial and final positions. For $k > 1$, the maneuver must take longer and there is a time expansion; for $k < 1$, a time contraction occurs.

To determine $c(k)$, use the control relation of Equation (3.7) to compute x_f in terms of $x_{f,n}$. The velocity at a general time $0 \le r \le t^*_f$ is given by

$$\begin{aligned} \dot{x}(r) &= \dot{x}(0) + \int_0^r u(t)dt = \int_0^r u_n(t/c)dt \\ &= c\int_0^{r/c} u_n(\tau)d\tau \end{aligned} \tag{3.8}$$

where $\dot{x}(0) = 0$, the k-dependence of c has been temporarily suppressed, and $\tau = t/c$ is a change of time variable. The final position follows accordingly:

$$x_f = x(0) + \int_0^{t^*_f} \dot{x}(r)dr = c\int_0^{t^*_f} \int_0^{r/c} u_n(\tau)d\tau dr$$

$$
\begin{aligned}
&= c^2 \int_0^{t_f^*/c} \int_0^s u_n(\tau) d\tau ds = c^2 \int_0^{t_{f,n}^*} \int_0^s u_n(\tau) d\tau ds \qquad (3.9)\\
&= c^2 x_{f,n}
\end{aligned}
$$

where $x(0) = 0$, s is the change of variable $s = r/c$, and Equation (3.4) allows the final simplification to $x_{f,n}$. Comparing to Equation (3.5), it follows that $c(k) = \sqrt{k}$, so that

$$t_f^* = \sqrt{k} \cdot t_{f,n}^* \qquad (3.10)$$

giving a square root profile. Using the symbol α for the variable final position gives $k = \alpha / x_{f,n}$ so that $t_f^* \propto \sqrt{\alpha}$.

Following a similar line of analysis, it is possible to find the explicit dependence of t_f^* on the control bound $\overline{u}$ as well. Assume now that the control bound varies as $\overline{u} = \frac{1}{\kappa} \overline{u}_n$, where $\overline{u}_n$ is the bound for the nominal problem and $\kappa > 0$ is a real, positive scaling factor. For this derivation, assume that $x_f = x_{f,n}$ (same final position), so that if $\kappa > 1$, there is less control authority available and $t_f^* > t_{f,n}^*$. Conversely, $\kappa < 1$ allows greater control authority so that $t_f^* < t_{f,n}^*$. As before, the optimal times must obey a relationship of the form $t_f^* = d(\kappa) t_{f,n}^*$, using d as a function symbol now instead of c. A similar "same strategy" hypothesis indicates that the general control signal relates to the nominal through

$$u(t) = \frac{1}{\kappa} u_n(t/d(\kappa)), \qquad (3.11)$$

where the premultiplying $1/\kappa$ factor handles the varying control bounds and the argument of u_n gives the appropriate time scaling.

Proceeding similarly to the variable-final position case, the velocity at any time $0 \leq r \leq t_f^*$ is given by

$$
\begin{aligned}
\dot{x}(r) &= \dot{x}(0) + \int_0^r u(t) dt = \frac{1}{\kappa} \int_0^r u_n(t/d) dt \qquad (3.12)\\
&= \frac{d}{\kappa} \int_0^{r/d} u_n(\tau) d\tau
\end{aligned}
$$

where $\dot{x}(0) = 0$ and τ is the change of variable $\tau = t/d$. The position follows as

$$
\begin{aligned}
x_f &= x(t_f^*) = x(0) + \int_0^{t_f^*} \dot{x}(r) dr\\
&= \frac{d}{\kappa} \int_0^{t_f^*} \int_0^{r/d} u_n(\tau) d\tau dr = \frac{d^2}{\kappa} \int_0^{t_f^*} \int_0^s u_n(\tau) d\tau ds \qquad (3.13)\\
&= \frac{d^2}{\kappa} x_{f,n}
\end{aligned}
$$

where $x(0) = 0$, s is the change of variable $s = r/d$, and Equation (3.4) again gives the simplifying expression for $x_{f,n}$. The underlying stipulation that $x_f = x_{f,n}$, implies

$d^2/\kappa = 1$ so that $d(\kappa) = \sqrt{\kappa}$, giving

$$t_f^* = \sqrt{\kappa} t_{f,n}^*, \tag{3.14}$$

another square-root scaling coefficient. Combining this result with the above variable final position results gives that, in general

$$t_f^* \propto \sqrt{\alpha \cdot \kappa}. \tag{3.15}$$

3.1.3 Trajectory Parametrization

Given these analytic results for the behavior of $t_f^*(\alpha, \kappa)$, it is now possible to parametrize the motion of Equation (3.1), carry out the interpolation procedures of Section 2.3 and observe how the family $p(\alpha)$ behaves in terms of the maneuver time cost function.

Proceed by choosing a B-spline expansion of $x(t)$ following the examples of [30, 102, 113]. The simplicity of the double integrator system allows easy solution for the control signal u in terms of x. Note that this signal parametrization will closely match that of the three degree-of-freedom helicopter example in Chapter 4, although the model inversion there will be far more complicated due to the highly nonlinear system dynamics.

The spline parametrization relies on a normalized time variable τ defined as

$$\tau = t/T, \tag{3.16}$$

where $T \equiv t_f$ is the maneuver duration and will also be a free parameter, giving tremendous freedom over the shape of the trajectories. Now express the normalized-time position $x(\tau)$ signal in terms of a B-spline expansion of the form

$$x(\tau) = \sum_{i=1}^{N} c_i B_{i,k}(\tau), \tag{3.17}$$

where the spline order k is chosen as 6. The spline knot sequence is

$$S = \{0^6, 0.1, 0.2, \ldots, 0.8, 0.9, 1^6\}, \tag{3.18}$$

where 0^6 and 1^6 denote knots of multiplicity six at 0 and 1, respectively, allowing for nonzero initial and final positions and resulting in $N = 15$. See Appendix B for a detailed discussion of B-spline constructions and their properties. References [33, 34, 35] give a more comprehensive overview of B-spline methods.

Now choose the total trajectory parametrization p as the concatenation of the 15 spline coefficients c_i and the maneuver time variable T to obtain

$$p = \left[\{c_i\}_{i=1}^{15}, T\right], \tag{3.19}$$

giving $p \in R^{16}$. Expressed in the normalized time variable τ, the equations of motion

are

$$\frac{1}{T}x''(\tau) = u(\tau) \tag{3.20}$$

with the prime notation indicating differentiation with respect to τ. Spline derivatives can be taken using Equation (B.10), although many software packages contain tools to automate the differentiation process [35].

Now it is necessary to cast the general optimization problem of Program (3.2) in the form

$$\text{subject to} \quad \begin{array}{c} \min\limits_{p} f(p) \\ h(p,\alpha) = 0 \\ g(p) \leq 0, \end{array}, \tag{3.21}$$

consistent with the methods of Chapter 2. The constraint functions h and g define the parametrized feasible maneuver space (with "parameter" now referring to the variable final position $\alpha = x_f$). Because the p vector contains the maneuver duration T as one of its elements, the minimum time objective function is simply $f(p) = T = e_{16}^T p$, where e_{16} is the 16-element vector of 15 zeros followed by a 1.

The equality constraints in Program (3.2) dictate the initial and final positions, velocities, and accelerations. Therefore $h(p,\alpha)$ takes the form

$$h(p,\alpha) = \begin{bmatrix} x(0) \\ x(1) - \alpha \\ \frac{1}{T}x'(0) \\ \frac{1}{T}x'(1) \\ \frac{1}{T^2}x''(0) \\ \frac{1}{T^2}x''(1) \end{bmatrix} = 0, \tag{3.22}$$

where $x = x(\tau; p)$ is given by Equations (3.17) and (3.19).

The inequalities in Program (3.2) apply for all time $t \in [0, T]$, making them infinite-dimensional constraints impossible to express as a finite system $g(p) \leq 0$. Therefore, take a finite mesh sampling $\{0, \tau_k, 2\tau_k, \ldots, 1\}$ of the unit time interval to obtain

$$g(p) = \begin{bmatrix} -\overline{u} - \frac{1}{T^2}x''(0) \\ -\overline{u} - \frac{1}{T^2}x''(\tau_k) \\ \vdots \\ -\overline{u} - \frac{1}{T^2}x''(1) \\ \frac{1}{T^2}x''(0) - \overline{u} \\ \frac{1}{T^2}x''(\tau_k) - \overline{u} \\ \vdots \\ \frac{1}{T^2}x''(1) - \overline{u} \end{bmatrix} \leq 0, \tag{3.23}$$

where the control is cast explicitly in terms of the parametrized position signal x according to $u(\tau) = (1/T^2)x''(\tau)$. Choice of a suitable time mesh interval τ_k is problem dependent, often influenced by the system dynamics and the number of

spline coefficients. For this study, $\tau_k = 1/20$ is a satisfactory mesh.

3.1.4 Specific Cases

With the repositioning maneuver problem definition, optimal cost relationships, and trajectory parametrization in hand, it is possible to illustrate the interpolation method in practice with a (dimensionless) numerical example and observe how it performs. Consider the case of control bound $\overline{u} = 10$ and the class of "one-sided" reposition maneuvers where $\alpha > 0$. Specifically, design off-line two example trajectories, one suboptimal for $\alpha = 5$ and the other optimal for $\alpha = 35$. The suboptimal trajectory is obtained by solving an optimization problem for $\overline{u} = 9$, thus making it slower than the corresponding motion for $\overline{u} = 10$.

Figures 3-1 and 3-2 show the resulting cost performance and control profiles for the family of reposition maneuvers, created through application of Algorithm 3. Note that beginning with a suboptimal value at $\alpha = 5$, the parametrized family with cost $t_f(\alpha)$ integrates toward the known optimal solution at $\alpha = 35$, first using only equality boundary constraints (Equation (3.22)), and then, around $\alpha = 8$, encounters the control bound inequality constraints (Equation (3.23)) and adds them to the active set. From this point on, the parameterized maneuver family rides along the optimal cost square root profile on the way to the goal trajectory, spanning a family of fully saturated bang-bang profiles. In this case, the feasible maneuver projection (lines 1 through 4 of Algorithm 3) directs the integral solution towards, and then into, the control constraints, making the parametrized family $p(\alpha)$ an optimal family over a large portion of the solution arc, despite nonenforcement of the Karush-Kuhn-Tucker (KKT) optimality conditions. The example trajectories are close enough to the performance limit that the projection operation (occurring in the space of $v = [p^T \alpha^T]^T$) encounters the nonconvex optimality curve.

Note, that continually testing for and then introducing the inequality constraints to the active set does not come without some cost. As is clear in Algorithm 3 and will be quantified in Section 3.2, the introduction of active inequality constraints adds dimensionality to the tangent plane computation. However, this additional load is still less than that required for full nonlinear parametric programming, which tracks a vector (including Lagrange multipliers) essentially double the size of p. Fortunately, as shown in Figures 3-1 and 3-2, the additional computation may come at the benefit of producing high-quality, that is, low cost maneuvers.

However, if in a given situation it is preferable to either avoid testing for inequalities or to apply more direct control on the behavior of $p(\alpha)$, it is possible to impose suitable user-selected equality constraints. Beginning with the same example trajectory at $\alpha = 5$, but instead using a suboptimal example at $\alpha = 35$ (again obtained by solving a nonlinear program with $\alpha = 35$ and $\overline{u} = 9$), one can add a maneuver time constraint of the form $T(\alpha) = \hat{f}(\alpha)$ for some feasible, differentiable $\hat{f}$ function. Note that such user-added constraints are possible since there are $n - m = 16 - 6 = 10$ extra degrees-of-freedom available after relaxing the KKT optimality conditions. Figures 3-3 and 3-4 show the result when choosing $\hat{f}(\alpha) = k^* \sqrt{(10/9) \cdot \alpha}$ where k^* is a con-

stant such that $t_f^*(\alpha) = k^* \sqrt{\alpha}$. The continuous maneuver family interpolates without encountering the optimal performance bound and thus never saturates the controls, improving the algorithm speed (see Inequality (3.27) below). As will be shown in Chapter 5, this ability to directly prescribe $T(\alpha)$ constraints is essential to creating maneuver classes amenable to planning with mixed integer-linear programming. In cases of other motion planners with different maneuver cost and shape requirements, the appropriate user-constraints can be added to $h(p, \alpha) = 0$, providing exact compatibility between path planning and trajectory generation.

Note that one should not be under the impression that every interpolated maneuver is similar to that in Figures 3-1 and 3-2, always encountering the feasibility bounds and discovering a large family of optimal maneuvers for "free". Consider the same maneuver type, but now in a "two-sided" form, where α may be either negative or positive. Begin with suboptimal example maneuvers for $\alpha = -30$ and $\alpha = 30$. Give these maneuvers the same time duration and interpolate between them as shown in Figures 3-5 and 3-6. The difference vector $u(s) = v_2 - v(s)$ has a zero in the maneuver time component that the projection operation will not perturb. As such, the interpolation modifies the geometric shape of the trajectory and control signal, but never improving its performance. In fact, as seen in Figure 3-6, the control loses the bang-bang strategy, so that the given examples, even though suboptimal, did not endow the maneuver class with any useful attributes.

However, exploiting the already demonstrated extra degrees-of-freedom, the user can add a differentiable constraint of the form $T(\alpha) = \hat{f}(\alpha)$ as shown in Figure 3-7 to take advantage of the available control authority for this class of maneuvers. (The selected $\hat{f}(\alpha)$ was of the form $c_1 - c_2/(c_3\alpha^2+1)$ in this example, with each c_i a positive real number). Figure 3-8 shows the effect on the control strategy, where $u(\tau)$, and thus $u(t)$, take a more intuitively beneficial approach to driving the system, achieving improved performance over a much wider range of the parametrized family.

Based on these examples, there are several useful practices to adopt when dealing with general trajectories of nonlinear systems, where it is impossible to exhaustively investigate interpolation performance. For example, in those situations where it is important to obtain low cost maneuvers in real-time, an effective strategy is to explore the optimal cost boundaries off-line with batch nonlinear programming, while computational resources are abundant. From this effort, it is possible to then select example trajectories that lead to on-line interpolation paths at or near the performance limit. In cases where on-line resources are at a premium, for example on expendable vehicles or when emergency maneuvers are required, extra constraints similar to $T(\alpha) = \hat{f}(\alpha)$ can be introduced to avoid inequality boundaries and speed up computation.

For cases where pilot motion capture gives example trajectories, and there is no external objective function, interpolation pursues a feasible trajectory path through the flight envelope, encountering and dealing with inequalities as necessary.

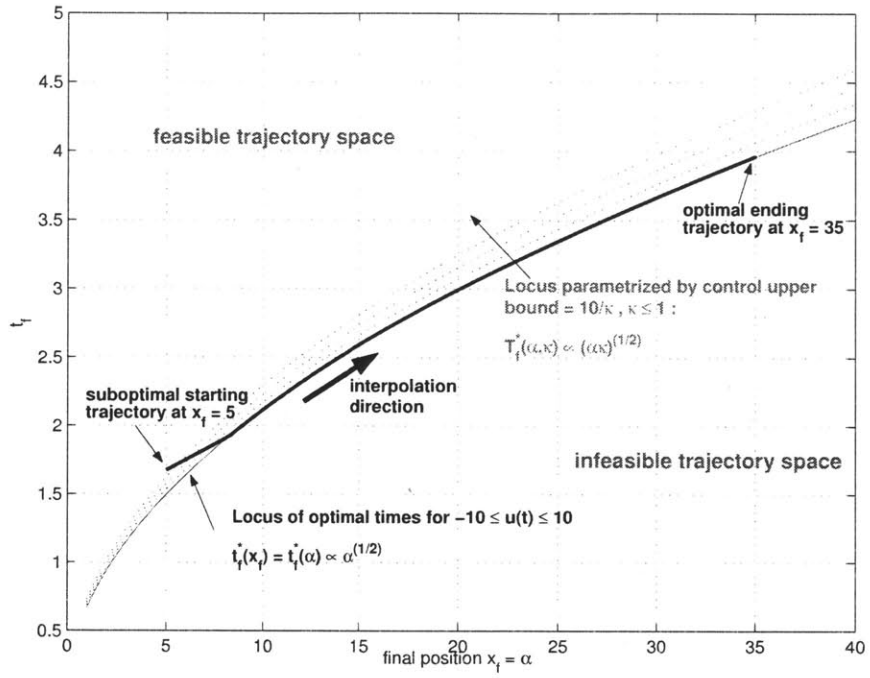


Figure 3-1: Cost performance of parametrized reposition maneuvers. Beginning with a suboptimal maneuver at $\alpha = 5$, the trajectory interpolation scheme integrates toward the known optimal maneuver at $\alpha = 35$, encountering the control bounds around $\alpha = 8$.

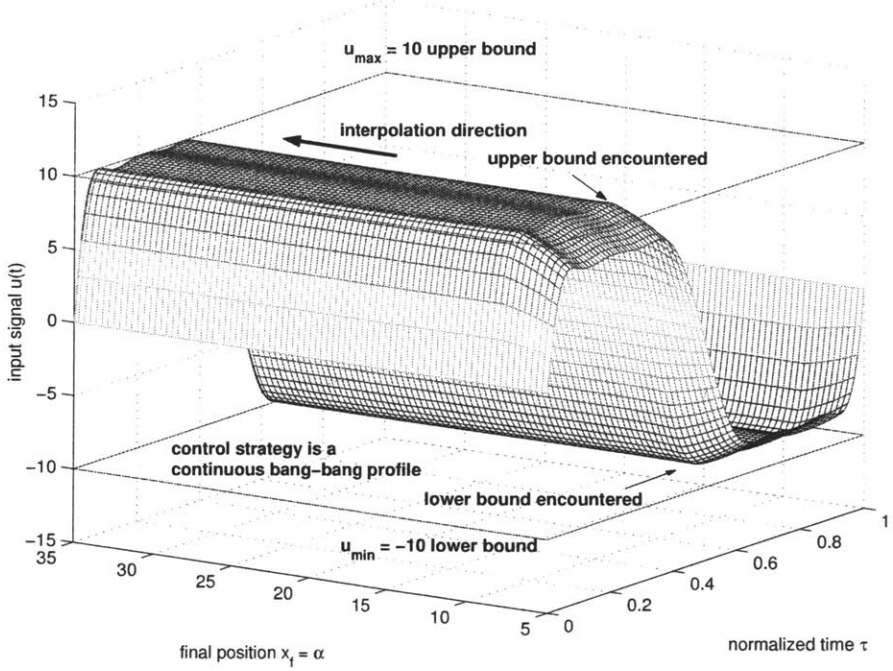


Figure 3-2: Interpolated control signals corresponding to Figure 3-1. Horizontal axes are normalized time τ and final position α (increasing right-to-left). The control signals encounter, and then ride along the upper and lower bounds.

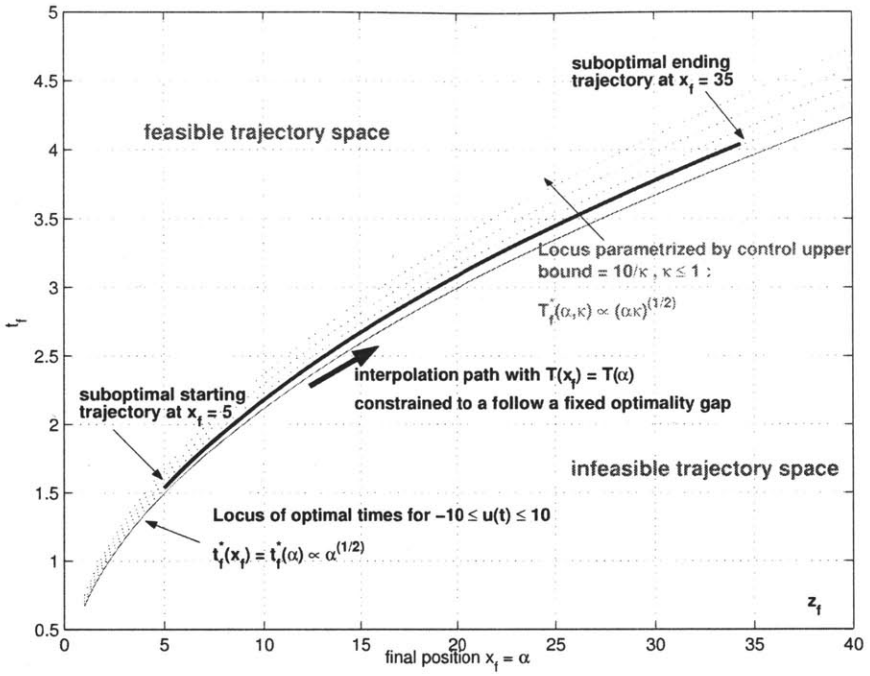


Figure 3-3: Cost performance when maneuver time is constrained to follow a suboptimal curve. The extra degrees-of-freedom created by dropping the Lagrange multipliers allow the user to add constraints affecting the shape of $p(\alpha)$. Here, a path constraint $T(\alpha) = \sqrt{10/9} \cdot t_f^*(\alpha)$ is added to avoid evaluating inequality constraints.

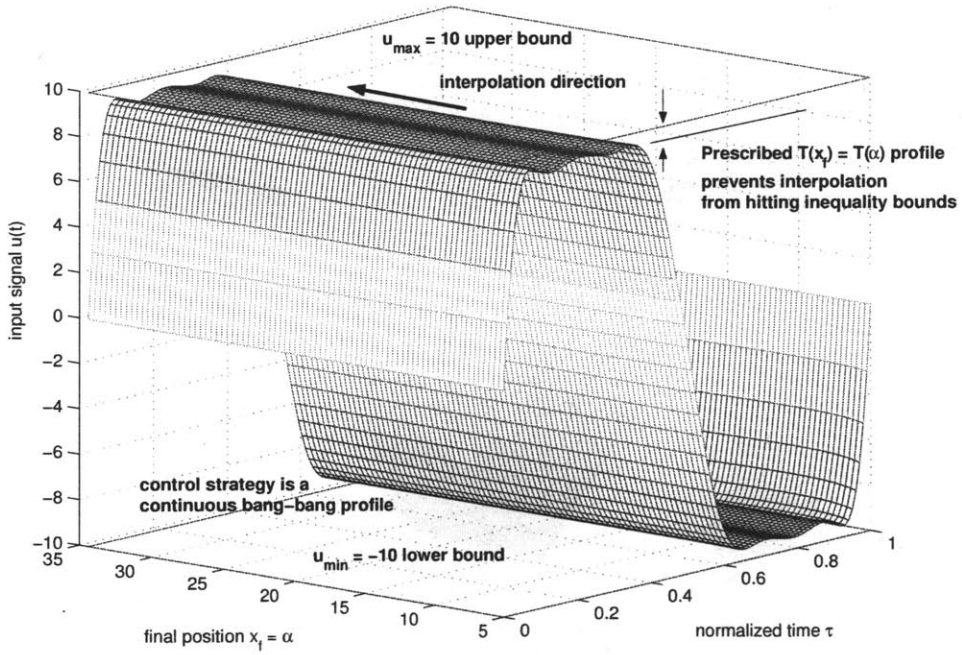


Figure 3-4: Interpolated control signals corresponding to Figure 3-3. The control signals maintain the bang-bang strategy of the bounding examples while staying away from the saturation bounds.

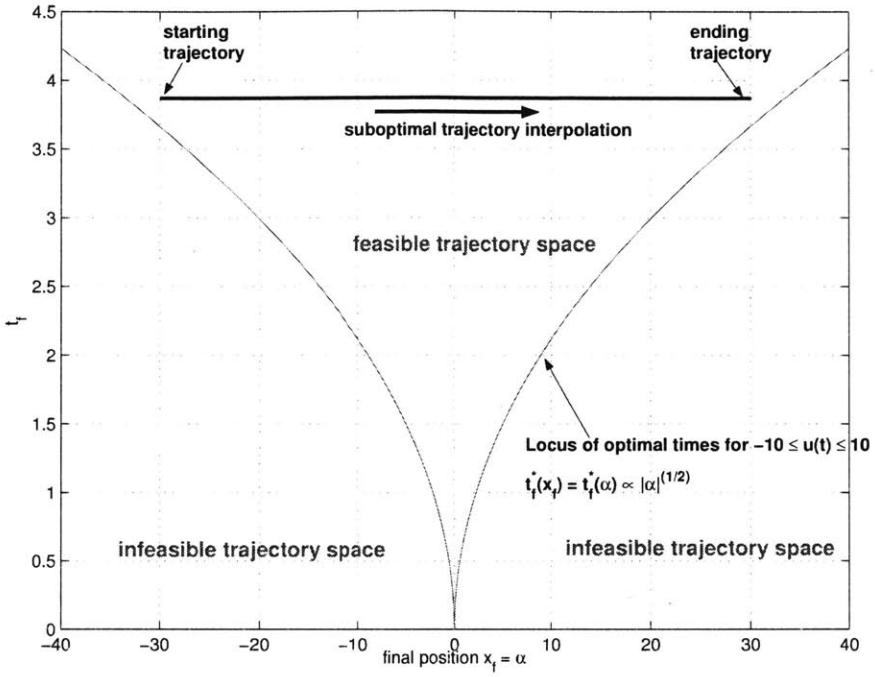


Figure 3-5: Cost performance of interpolated trajectories for a 2-sided reposition maneuver. The example trajectories have the same time duration t_f, so therefore all the intermediate motions do as well. The maneuver cost curve $T(\alpha)$ clearly does not exploit the large potential for improved performance.

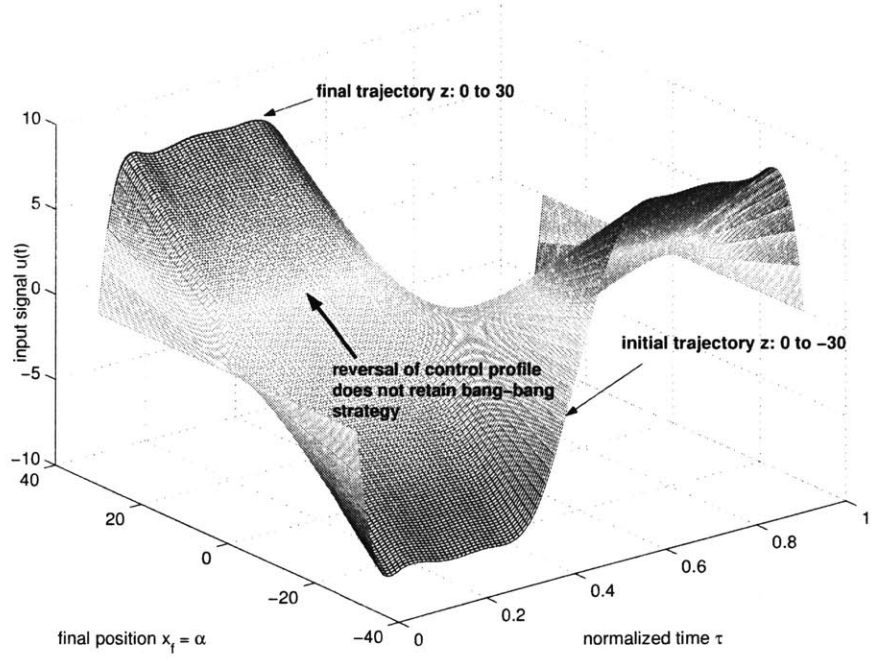


Figure 3-6: Interpolated control signals corresponding to Figure 3-5. The control profile becomes nonaggressive over much of the maneuver family, first losing and then reobtaining the bang-bang strategy of the known examples.

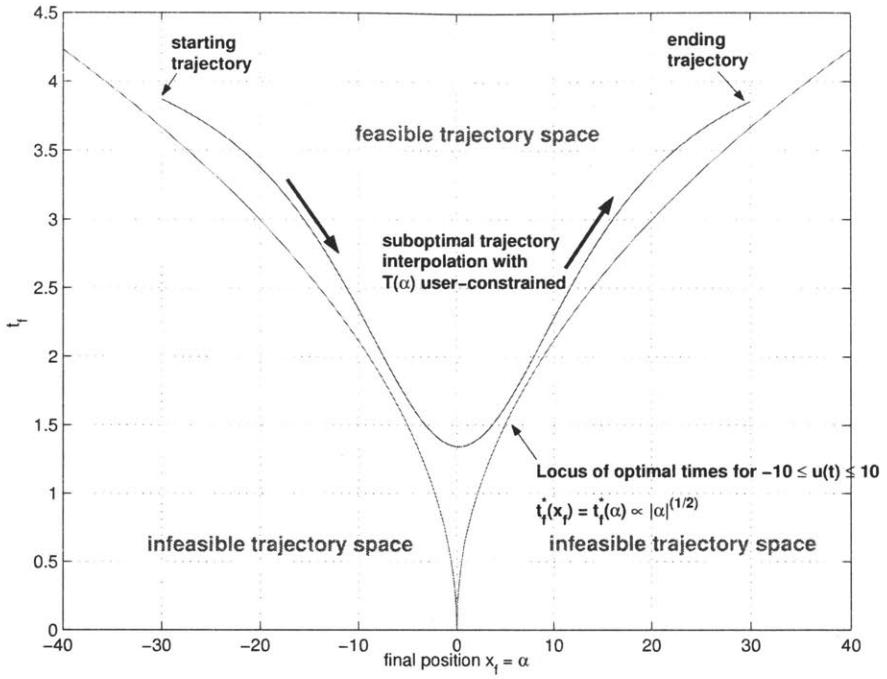


Figure 3-7: Cost performance of interpolated trajectories for a 2-sided reposition maneuver when the trajectory time is constrained to follow a user-selected $T(\alpha)$ curve. The specific curve is chosen to take advantage of available control authority for intermediate maneuvers.

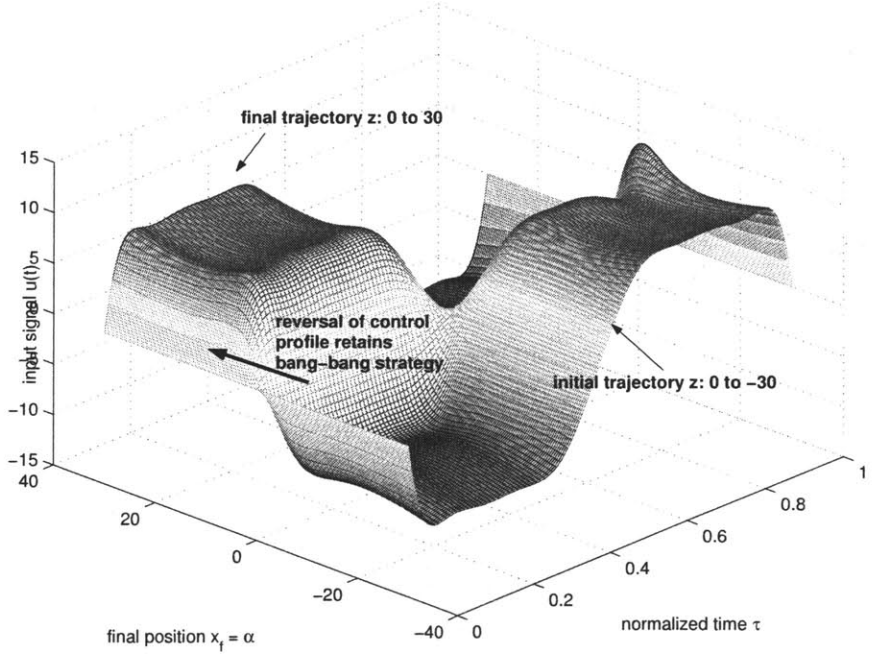


Figure 3-8: Interpolated control signals corresponding to Figure 3-7. The selection of a function $T(\alpha)$, made possible by available degrees-of-freedom, leads to more aggressive performance over the entire maneuver family without having to resort to full nonlinear parametric programming.

3.2 Computation Bounds

The maneuver interpolation algorithms of Section 2.3 present a simple alternative for on-line trajectory generation. Using known feasible motions as guides, all computation invested in the integral methods of Algorithms 1 and 3 goes towards producing members of a feasible maneuver class $p(\alpha)$. This feature is in contrast to nonlinear programming, where much effort can be spent searching before a single feasible solution is found.

This section determines bounds on the expected computational load when generating maneuvers according to the interpolation scheme. Here, computation is defined in terms of the number of elementary operations, that is, multiplications and additions. These bounds are useful in assessing the overall complexity of trajectory interpolation and for determining efficient ways to implement it on-line. The present analysis method is to pose the interpolation process of Algorithm 3 as a numerical integration problem, where the majority of the work goes into computing $\hat{u}$ derivative vectors. A lower bound on the derivative calculation occurs when there are no active inequalities, and the tangent plane involves only equality constraints, giving the maneuver the maximum possible number of degrees-of-freedom. The algorithm computation obtains the upper bound when the maximum allowable active set dimension (given by the number of components in p minus the number of components in $h(p, \alpha)$ [14, 92]) occurs throughout the entire interpolation process. Note for comparison, that a full parametric programming effort would essentially double the dimensionality of the present path-following algorithm, since Lagrange multipliers for all equality and active inequality constraints must be traced as well.

The computational bounds serve as guidelines based on the problem dimensionality. Many specific options, such as the choice of numerical integration method and algorithms for null space and projection operations, depend on the platform and available software. Further, it is difficult to quantify the effort in computing both the constraint values and first derivatives for a general trajectory parametrization p. Therefore, make the assumption that the calculation of a given component, say $h_i(p, \alpha)$ or $g_j(p)$, requires effort on the order of the argument size, that is, the number of components in p and/or α. If better estimates for these calculations are available, then they can be inserted into the appropriate places below.

Define the following nomenclature for the dimensions of quantities in Algorithm 3:

$$\begin{aligned} C &= \text{number of elementary operations during interpolation} \\ N_s &= \text{number of integration steps to reach goal maneuver} \\ N_p &= \text{number of trajectory parameters} \\ N_\alpha &= \text{number of maneuver coordinates/parameters} \\ N_v &= \text{total number of interpolation variables} = N_p + N_\alpha \\ N_h &= \text{number of equality constraint components} \\ N_g &= \text{number of inequality constraint components} \end{aligned} \tag{3.24}$$

N_{g_0} = number of currently active inequality constraints
M = derivative update interval, in number of integration steps.

For a well-posed problem it must hold that $N_h \leq N_p$, while the inequality constraints obey $0 \leq N_{g_0} \leq N_p - N_h$, as noted above.

The variable N_s gives the number of integration steps required to reach the goal trajectory from the given examples, making it a coarse measure of how far α_{goal} is from either α_1 or α_2. Naturally, one should proceed from whichever α endpoint is "closer" to α_{goal}.

The computation estimates here are based on a fourth-order Runge-Kutta integration method with fixed step size Δs, chosen for this analysis because of its generality and widespread use. In practice, the specific integration algorithm used is software and platform dependent. Improvements can be made by using variable-step size integration methods, accelerating the interpolation process and implicitly retaining some of the benefits of predictor-corrector methods. In addition, the quantity M is included in the calculations, highlighting the possibility of only updating the derivative information periodically. Other schemes such as periodic Broyden derivative updates are possibilities for reducing computational overhead as well [17].

In Algorithm 3, the numerical integration applies to an equation of the form $dv/ds = \hat{u}(s, v)$. The basic Runge-Kutta step [158] is an iterative process following

$$v_{k+1} = v_k + \frac{\Delta s}{6}(q_1 + 2q_2 + 2q_3 + q_4), \tag{3.25}$$

where each q_i is an evaluation of $\hat{u}(s, v)$. Specifically,

$$\begin{aligned} q_1 &= \hat{u}\left(s_k, v_k\right) \\ q_2 &= \hat{u}\left(s_k + \frac{\Delta s}{2}, v_k + \frac{1}{2}q_1\right) \\ q_3 &= \hat{u}\left(s_k + \frac{\Delta s}{2}, v_k + \frac{1}{2}q_2\right) \\ q_4 &= \hat{u}\left(s_{k+1}, v_k + q_3\right). \end{aligned} \tag{3.26}$$

Based on Equation (3.25), it will take $C = N_s\left(\left(\frac{1}{M}\right)N_{\hat{u}} + 8N_v\right)$ elementary operations to reach the goal maneuver. Here, $N_{\hat{u}}$ denotes the computation load in computing the q_i quantities, which follow directly from Algorithm 3. The remaining $8N_v$ computations come from the basic algebra of Equation (3.25).

To obtain the desired bounds, it is required to evaluate $N_{\hat{u}}$ for two cases: equality constraints only (lower bound) and equality plus *maximum* inequality constraints (upper bound). Proceeding through each line of Algorithm 3 gives the following subcalculations.

- Line 1 is simple difference, taking N_v operations.
- Line 2 differentiates the equality constraints h with respect to v. As mentioned above, assume the evaluation of a single equality constraint takes of order N_v computations so that evaluating the entire h vector takes $k_1 N_h N_v$ operations, with k_1 a

positive number. If the derivative matrix $D_v h$ is calculated numerically by a simple forward difference method, it will require N_v+1 evaluations of h (one nominal case and N_v perturbations) followed by N_v differencings of h and its perturbations, each time dividing by a small number. The procedure totals to $k_1 N_h N_v + N_v(k_1 N_h N_v + 2N_h)$ operations. If the differentiation is performed analytically (by evaluating a derived expression) and assuming that the vector $\partial h/\partial v_k$ with $k \in \{0, \ldots, N_v\}$ takes the same effort as evaluating h, the required computation for $D_v h$ is $k_1 N_h N_v^2$.

• Line 3 computes the null space of the matrix $D_v h$ (tangent hyperplane to equality constraint surface). The singular value decomposition $D_v h = U\Sigma V'$ is an effective method for doing so, with the last (at least) $N_v - N_h$ columns forming an orthonormal basis set Z for Null($D_v h$). There are many algorithms to compute the SVD or (reduced SVD). However most require $k_2 N_h N_v^2 + k_3 N_v^3$ elementary operations, where k_2 and k_3 are positive integers between 1 and 20 [60].

• Line 4 projects a vector $u \in R^{N_v}$ onto the range space of a matrix Z, which is generally a member of $R^{N_v \times (N_v - N_h)}$. Assuming a standard 2-norm least-squares error projection, the computational load is $k_4 N_v(N_v - N_h) + k_5(N_v - N_h)^3$, where k_4 and k_5 are positive numbers between 1 and 15, depending on the particular elimination algorithm used [60].

• Line 5 is a simple scaling, requiring N_v computations.

• Line 6 checks the values of all inequality constraints, and is assumed to take $N_p N_g + N_g$ operations (evaluation of N_g constraints, each dependent on N_p elements of p, followed by a sign check for all components of g).

• Line 10 computes the derivatives of up to $N_p - N_h$ additional inequality constraints. Following the discussion of Line 2 above, this will take $k_6(N_p - N_h)N_p + N_p(k_6(N_p - N_h)N_p + 2(N_p - N_h))$ operations, where k_6 is a positive number analogous to k_1 in the discussion of line 2. (Note that there are only $N_p - N_h$ components to evaluate and only N_p nontrivial derivatives to compute, the last N_α columns of the $D_v g_j$ being identically zero.) This derivative matrix is then multiplied by a vector of dimension N_v and compared to zero, requiring $2(N_p - N_h)N_v + (N_p - N_h)$ more operations.

• Line 14 follows similarly to Line 3, giving $k_2 N_p N_v^2 + k_3 N_v^3$ operations, the maximum possible number of rows for $[D_v h; D_v g_{J_2}]$ being N_p.

• Line 15 follows similarly to Line 4, giving $k_4 N_v N_\alpha + k_5 N_\alpha^3$ operations, the minimum dimension of the null space being $N_v - N_p = N_\alpha$.

• Line 16 is a simple scaling, taking N_v computations.

Given these subresults and Equation (3.25), it is now possible to sum the number of operations for each step and obtain the desired bounds. The lower bound for the equality constrained case uses lines 1 through 6. The full equality and inequality upper bound uses all of lines 1 through 16. The result is as follows:

$$C_{lb} \leq C \leq C_{ub} \tag{3.27}$$

where

$$C_{lb} = N_s \left(\left(\frac{1}{M} \right) 4N_{\hat{u},lb} + 8N_v \right) \tag{3.28}$$

$$C_{ub} = N_s\left(\left(\frac{1}{M}\right)4N_{\hat{u},ub} + 8N_v\right)$$

and

$$\begin{aligned} N_{\hat{u},lb} &= 2N_v + N_g + N_p N_g + (2+k_1)N_h N_v + (k_1+k_2)N_h N_v^2 \\ &\quad + k_3 N_v^3 + k_4 N_v(N_v - N_h) + k_5(N_v - N_h)^3 \\ N_{\hat{u},ub} &= N_{\hat{u},lb} + N_v + (N_p - N_h) + 2(N_p - N_h)N_v + k_4 N_v N_\alpha \\ &\quad + k_2 N_p N_v^2 + (2+k_6)(N_p - N_h)N_p + k_6(N_p - N_h)N_p^2 \\ &\quad + k_3 N_v^3 + k_5 N_\alpha^3. \end{aligned} \tag{3.29}$$

Based on this result, it is worth noting that the interpolation algorithm load is a polynomial function of the problem size variables. An efficient implementation for on-line use would involve first precomputing the parametrized family $p(\alpha)$ off-line and saving a number of interpolated solution points, instead of just the bounding examples v_1 and v_2. The storage load would be small since p is typically of vector of fewer than 100 numbers for an autonomous vehicle. Then on-line, the trajectory generation would proceeded as in Algorithm 3, but with interpolation occurring over a much shorter interval, reducing the multiplying factor N_s in Equations (3.28). As stated before, another large savings comes from choosing M as large as possible without degrading fidelity to the nonlinear model.

3.3 Continuity

The parametrized family $p(\alpha)$ that results from Algorithms 1 and 3 comes from an integrated feasible path of the form $v(s)$. Typically, proper scaling sets s equal to α (line 5 of Algorithm 1, lines 5 and 16 of Algorithm 3) so that $p(s) = p(\alpha)$ identically. Assume here that α is scalar without loss of generality, since a differentiable mapping $\gamma : R \to R^{N_\alpha}$ may be chosen as $\alpha = \gamma(\sigma)$ with $\sigma \in R$, as noted in Section 2.3.6.

The integrated vector v takes the form

$$v(s) \equiv \begin{bmatrix} p(s) \\ \alpha(s) \end{bmatrix} = \begin{bmatrix} p_1 \\ \alpha_1 \end{bmatrix} + \int_0^s \frac{dv}{ds}(\sigma')d\sigma' \tag{3.30}$$

where σ' is a dummy variable of integration and s may be interpreted as $\Delta\alpha$, making it possible to integrate exactly to a desired maneuver $p(\alpha_{goal})$.

As the output of an integral function, $v(s)$ is a continuous function [128], even though the derivative dv/ds may not be continuous throughout $[0, s]$. In particular, any time the active set changes, it is likely that the local tangent vector makes a sudden jump, as seen in Figure 2-2 for a linear programming polytope. Assuming that dv/ds is smooth between active set switching points (which holds when $h(p, \alpha)$ and $g(p)$ are continuously differentiable in their arguments), then a finite number of active set changes is sufficient [104] to guarantee that $v(s)$ in Equation (3.30) exits on the interval $[0, s]$.

Assuming that the feasible maneuver space is "well-posed" in that $v(s)$ will actually reach v_2 without getting stuck in any local entrapments (a condition that is extremely hard to guarantee for arbitrary systems and maneuvers but seems to hold in practice when the projection inner product $\hat{u}^T u$ stays well away from 0), then all components $v_i(s)$ are continuous functions with boundary values $v_{i,1}$ and $v_{i,2}$. From the intermediate value theorem, each v_i must therefore assume all values between $v_{i,1}$ and $v_{i,2}$ [128]. If the system behavior $z(t;p)$ is a continuous function of p, then the state and input functions will vary continuously with the integration variable s.

Lastly, any continuous functions $f(p)$ and $\hat{f}(\alpha)$ will vary continuously with s as well. This includes most standard cost functions, such as maneuver duration, control energy, and fuel consumption. Therefore, if the example maneuvers v_1 and v_2 were generated off-line according to nonlinear optimization of the f and/or $\hat{f}$ functions, then the on-line interpolation must assume cost values intermediate to the two examples. What is not guaranteed however, is that f and $\hat{f}$ will vary monotonically, assuming no values outside the range of the examples. If monotonic behavior is required, then the parametrized functions $f(p(\alpha))$ and $\hat{f}(\alpha)$ must be checked and the example trajectories modified until the desired behavior is observed.

3.4 Applicable Systems

The trajectory interpolation method introduced in Chapter 2 applies to nonlinear systems whose trajectory feasible space can be expressed by the relations

$$\begin{aligned} h(p,\alpha) &= 0 \\ g(p) &\leq 0. \end{aligned} \tag{3.31}$$

The nonlinear equations of motion $\dot{x} = f(x,u)$ must be either implicitly or explicitly included in Equations (3.31). Recall from Section 2.3.3 that the $h_e(p)$ partition of $h(p,\alpha) = 0$ often serves as a finite-dimensional approximation of $\dot{x} = f(x,u)$.

The essential requirements for interpolation algorithm applicability to a system are that there exist a parameter vector p such that $z(t;p)$ is differentiable with respect to p and that the functions $h(p,\alpha)$ and $g(p)$ are continuously differentiable in their arguments.

The methods of this thesis will not apply to systems that do not have smooth signal parametrizations $z(t;p)$ or that do not have differentiable maps h and g. Many system models contain nondifferentiable terms, look-up tables, discrete mode switching, or gain scheduling and will not fit the method requirements without some sort of approximation or revision.

Of those systems that *are* amenable to the methods of Chapter 2, there are several special model classes worth mentioning, especially since they are commonly for autonomous vehicles. Of particular interest are cases where the vector p describes the output or trajectory-space signals of a vehicle, with model inversion used to obtain intermediate states and control inputs. For invertible and near-invertible systems, it is preferable to have greatest possible control over the output space behavior.

Parametrization of input signals only, while mathematically possible, require forward integrations (shooting methods) to obtain the state x and any outputs y, a numerically costly process that does not lend itself to rapid constraint differentiation and maneuver interpolation [16].

Differentially flat system models are an emerging method for vehicle trajectory design [48, 102, 106, 113, 149]. They allow for parametrization of output signals and algebraic solution for inputs signals, making them almost ideal for trajectory interpolation. A key requirement is that the number of so-called flat outputs equals the number of system inputs, where the output depends on the state, the input, and n_1 input derivatives according to

$$y = y(x, u, \dot{u}, \ldots, u^{(n_1)}). \tag{3.32}$$

For some systems, it can be difficult to actually determine the flat outputs even if they can be proven mathematically to exist. For those systems where y *can* be found, solution for the state and input involves algebraic operations on the output and its first n_2 time derivatives:

$$\begin{aligned} x &= f_x(y, \dot{y}, \ldots, y^{(n_2)}) \\ u &= f_u(y, \dot{y}, \ldots, y^{(n_2)}). \end{aligned} \tag{3.33}$$

To design parametrized trajectories for such systems, one would pick p as a sufficiently smooth parametrization of y so that $y = y(t; p)$, making x and u differentiable functions of p as seen in Equations (3.33). In this case, there is no need for the $h_e(p)$ partition of $h(p, \alpha) = 0$, since the system model is explicit in the system inversion functions f_x and f_u.

It can be shown that flat systems include the class of full-state feedback linearizable systems [79, 106, 138, 150], another plant model amenable to trajectory interpolation. For these systems there exists a (diffeomorphic) change of variables $\xi = T(x)$ and a nonlinear state feedback relation of the form

$$u = \alpha(x) + \beta(x)v \tag{3.34}$$

such that the state-transformed system is a controllable, linear system:

$$\dot{\xi} = A\xi + Bv. \tag{3.35}$$

(The α appearing in Equation (3.34) should not be confused with the maneuver parametrization vector appearing throughout this thesis). In this setting, a clear trajectory parametrization is $\xi = \xi(t; p)$, since the transformed input can be put in the form $v(p)$ directly from Equation (3.35) and the state put in the form $x(p)$ using the inverse mapping $x(p) = T^{-1}\xi(p)$. Finally a differentiable $u(p)$ is then available from Equation (3.34). Again there is no need for a $h_e(p)$ partition since the system equations of motion directly influence the choice of T, α, and β.

In many situations, it is hard to find flat or linearizable outputs such that x and u are readily available through simple algebraic operations. Inversion of the system

model may involve nonlinear terms that are impossible to invert or would require numerical, rather than analytic, solution. In such cases, it may still be possible to find a workable signal parametrization that requires a nonholonomic constraint relation [31, 102]. Specifically, there may be a vector of (non-flat) outputs where the inputs and states can be found according to Equations (3.33) but also requiring an equality constraint of the form

$$\eta(y, \dot{y}, \ldots, y^{(n_3)}) = 0, \tag{3.36}$$

involving y and n_3 of its time derivatives. This situation arises when y contains more components than the number of system inputs, so a differential constraint of the form of Equation (3.36) is required to enable the "extra" degrees-of-freedom in y. Here, choose a parametrization $y = y(t; p)$ and now some $h_e(p)$ enforces Equation (3.36) along a discrete time mesh of the motion interval. Chapter 4 presents a detailed example of this type of system model.

Finally, for systems where it is impossible to find a useful output, it may make sense to use transcription (collocation) methods, which parametrize both the system state and input [16]. That is, the state and input are individually cast as $x = x(t; p_x)$ and $u = u(t; p_u)$, and vector p contains two partitions:

$$p = \begin{bmatrix} p_x \\ p_u \end{bmatrix}. \tag{3.37}$$

Again, since p appears to take more degrees-of-freedom than the system equations of motion will allow, it is necessary to impose discrete constraints of the form $c_k = x(t_{k+1}) - x(t_k) - \Delta t f(x_k, u_k) = 0$, based on a forward-Euler approximation to $\dot{x} = f(x, u)$ with time step Δt. A collected mesh of these so-called "defect" constraints then makes up the components of $h_e(p)$. If desired, higher-order approximations to the sampled equations of motion may be used.

Chapter 4

Application to a Three Degree-of-Freedom Helicopter

The preceding chapters laid out the motivation, derivation, and properties of a practical trajectory interpolation method and the resulting parametrized maneuver classes. It is now worthwhile to study the framework in practice on a nontrivial experimental rotorcraft. The three degree-of-freedom helicopter from Quanser Consulting [122] provides an easy-to-work-with dual-rotor helicopter platform amenable to system identification, piloted-flight motion capture, control design, and maneuver execution. Appendix A gives a detailed description of the helicopter's operating principles, the coordinate frame and state vector used for analysis, and the procedure employed to select and numerically identify descriptive linear and nonlinear models. The resulting nonlinear equations of motion, coupled with a specific trajectory parametrization vector p to be introduced presently, form the foundation for creating continuous helicopter maneuver classes.

This chapter opens by briefly discussing the nonlinear system model and the attributes of the Quanser vehicle as a research platform. The next section then discusses the technical details of trajectory parametrization, input signal calculation from model inversion, and selection of equality and inequality constraints. With the vector p and constraint functions $h(p)$ and $g(p)$ in place, one can then select interesting maneuver coordinate parameters α and find example trajectories through off-line nonlinear programming or piloted-flight motion capture. The chapter then discusses an intuitive feedforward control tracking design, making it possible to execute interpolated maneuvers on the vehicle, and finally concludes with four example parameterized trajectory classes. These maneuvers, a flaring quick-stop, a climbing maneuver, a dash-acceleration, and a hover-to-hover reposition motion, employ all the previously presented tools to compactly describe motions useful for typical rotorcraft operations. The following chapter then illustrates the integration of parametrized maneuvers into an existing hybrid motion planning framework, creating an extremely flexible vehicle guidance capability.

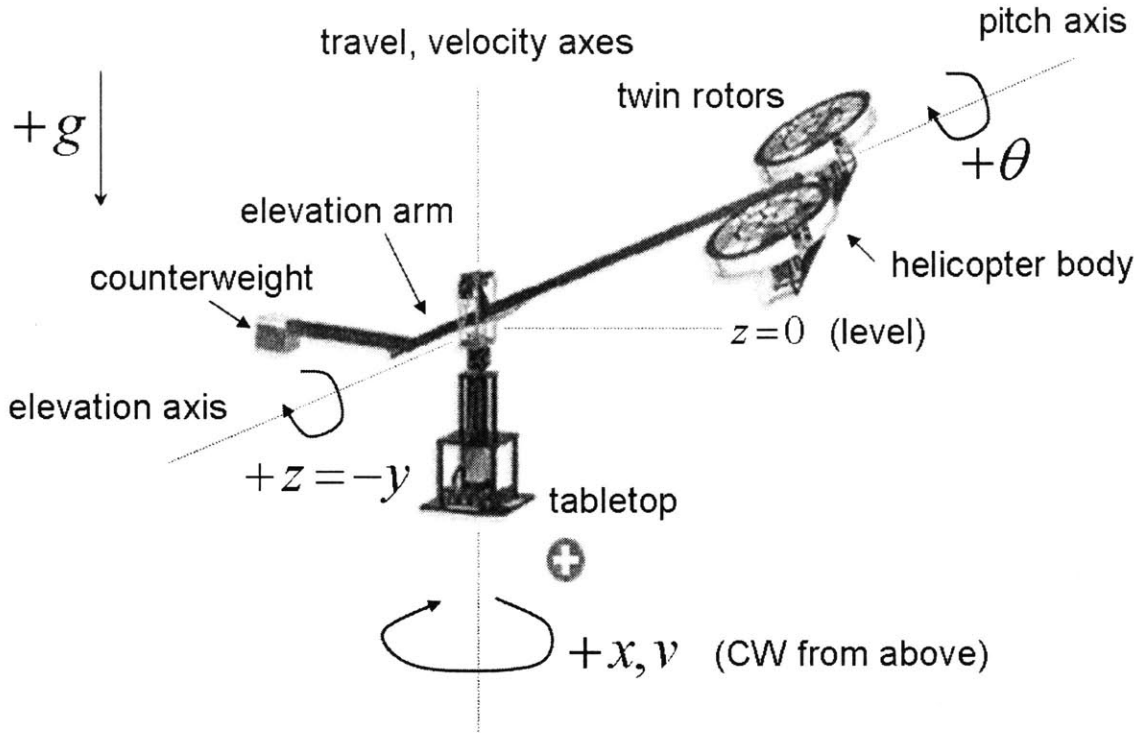


Figure 4-1: The three degree-of-freedom helicopter from Quanser Consulting. The instantaneous vehicle configuration is given by the travel angle x, pitch angle θ, and elevation angle z (downward), alternatively y (upward).

4.1 Vehicle Description

The rotorcraft considered in this chapter is the three degree-of-freedom (3-DOF) tabletop-mounted helicopter from Quanser Consulting (see Figure 4-1). Instead of the full six primary degrees-of-freedom seen in full-scale airborne helicopters (3 translational, 3 rotational), the Quanser experimental platform has three angular degrees-of-freedom based on the hinged motions of an essentially rigid arm and a main motor platform. As seen in the figure and discussed in-depth in Appendix A, the admissible motions are *elevation* (denoted by angular position z, measured positive downwards from horizontal), *travel* (denoted by angular position x or velocity v, measured positive counterclockwise from an arbitrary zero point when viewed from above), and *pitch* (denoted by angle θ, measured positive clockwise from horizontal when viewed radially inward). These three degrees-of-freedom emulate the two translational DOFs and one rotational DOF present in full-scale helicopter longitudinal motion, with the elevation variable z acting as an altitude.

Similar to standard rotorcraft, collective and cyclic input channels drive the Quanser system, with the collective signal providing a common voltage V_{coll} to two DC motors, each driving a fixed-blade pitch rotor. This input governs the net thrust produced by the twin rotors, tending to raise the vehicle when θ is close to zero or accelerate the system in the travel direction when θ has larger magnitude. The cyclic input channel provides a differential voltage V_{cyc} to the two motors, changing

the relative thrust of the two rotors and thus tending to rotate, or pitch, the vehicle. Similar to full-scale rotorcraft, the cyclic is particularly useful for reorienting the thrust vector and changing the horizontal acceleration component. See Appendix A for an in-depth discussion of the system operating principles as well as physical modeling and experimental identification of the input channels and vehicle dynamics.

As seen in Appendix A, and Equation (A.19) in particular, the system nonlinear equations of motion are as follows:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -a_1 v - a_2 V_{coll}^2 sin(\theta - \theta_a) \\ \ddot{\theta} &= -b_1 \dot{\theta} - b_2 sin(\theta) + b_0 + b_3 v|v| + b_4 V_{coll} V_{cyc} \\ \ddot{z} &= -d_1 \dot{z} + d_2 cos(z) - d_3 sin(z) - d_5 v^2 - d_4 V_{coll}^2 cos(\theta), \end{aligned} \tag{4.1}$$

where x, v, z, and θ are defined as above and are measured in angular units. The input signals V_{coll} and V_{cyc} are voltages and the remaining a_i, b_i, and d_i quantities are constant coefficients, with numerical values given in Table A.2.

Note that the structure of Equation (4.1) contains many similarities to standard rotorcraft vehicle models [54, 58, 88, 99]. There are numerous trigonometric terms to account for thrust vector tilting, kinematic effects, and a nonlinear pendulum restoring force on the pitch angle. In addition, there are linear damping terms on travel rate, elevation, and pitch, as well as a nonlinear $v|v|$ term that governs the speed-dependent effects of vehicle travel on pitch dynamics (see Section A.3.4). This latter term has the same mathematical form as many common nonlinear translational damping terms in helicopter and fixed-wing aircraft models [58]. Finally, as seen in line 2 of Equation (4.1), one input term appears as a product with a function of the vehicle state, a nonlinearity often occurring in helicopter rotor dynamic models [99].

However, there are also several complicating terms and effects not seen in full-scale rotorcraft models. First, as seen in the second and fourth lines of Equations (4.1), the collective voltage enters as a quadratic expression instead of the usual linear term, due to the steady-state relationship between rotor angular speed and the resulting thrust magnitude. In addition, the collective and cyclic appear as a product in the third line, a fairly rare occurrence in rotorcraft literature and certainly a complicating factor in this case, as it will necessitate working with V_{coll} directly instead of V_{coll}^2, thus requiring a square root during model inversion. Finally, because of the nonlinear pendulum action associated with the three degree-of-freedom helicopter, the vertical altitude dynamics (fourth line) are highly elevation-dependent, meaning that the static thrust inputs to maintain a hover state depend explicitly on z, adding a further complication. Note that such elevation, or altitude, dependence can appear when considering air density variations for rotorcraft and fixed-wing aircraft flight over large altitude ranges [13, 88].

The above commonalities and differences with full 6-DOF helicopters make the vehicle of Figure 4-1 worthy of study from a maneuver interpolation viewpoint. The model nonlinearities will help validate the thesis methods and demonstrate that extension to existing aerobatic vehicle testbeds should be straightforward, especially

as many agile maneuvers of interest take place in only two spatial dimensions. In addition, given the availability of reliable control systems, it is sometimes possible to employ further simplified special-purpose representations of vehicle dynamics [54, 55, 57, 114].

4.2 Trajectory Design

This section develops a finite trajectory parametrization p for the helicopter system of Figure 4-1. The emphasis is on finding a compact representation that will allow easy interpolation of the output-space behavior while enabling analytic solution for dynamically consistent input voltages. Fortunately, references [101] and [102] provide an effective B-spline-based design example useful for flat and near-flat vehicle models. The task then becomes one of tailoring these existing methods for the helicopter under current consideration, determining means to guarantee dynamic feasibility and model invertibility, and finding appropriate constraint functions to allow parametrization of interesting maneuvers. Together, these efforts cast the helicopter system into the finite-dimensional optimization framework of Section 2.1.1.

4.2.1 Signal Parametrization

In Equation (2.1), a behavior signal $z(t)$ gives a continuous-time description of the overall system trajectory and is a function of the state and/or input vectors. In the present context, use the bold symbol $\mathbf{z}$ to denote the system behavior to avoid confusion with the elevation state z in Equation (4.1). A suitable behavior choice for the helicopter is $\mathbf{z} = [v, z, \theta]^T$, meaning that $\mathbf{z}$ directly gives the values of all three DOFs. Experimentation with smaller dimension behaviors, such as $\mathbf{z} = [v, z]^T$ or $\mathbf{z} = [z, \theta]^T$ generally lead to difficulties with model inversion or required awkward forward integrations to determine travel behavior, thus nullifying the benefits of working directly in the output space.

Given a choice of $\mathbf{z}(t)$, the task is then to find a finite parametrization vector p and thus $\mathbf{z}(t; p)$, making each output signal an explicit function of time, given a particular value of p:

$$\begin{aligned} v(t) &= v(t; p) \\ z(t) &= z(t; p) \\ \theta(t) &= \theta(t; p), \end{aligned} \tag{4.2}$$

where $t \in [0, T]$ and the scalar T is the (finite) time duration of a trajectory. As demonstrated in references [101] and [102], it is extremely convenient to work with a unit time scale, so that the T may be extracted and used as a free variable. To this end, define the normalized time variable τ as

$$\tau = \frac{t}{T}, \tag{4.3}$$

where $\tau \in [0,1]$ throughout a maneuver. Given the unit time scale, define each of the behavior components as a B-spline function of $\tau \in [0,1]$ according to

$$\begin{aligned} v(\tau) &= \sum_{i=1}^{n_v} c_{i,v} B_{i,k}(\tau, i : i+k) \\ z(\tau) &= \sum_{j=1}^{n_z} c_{j,z} B_{j,k}(\tau, j : j+k) \\ \theta(\tau) &= \sum_{l=1}^{n_\theta} c_{l,\theta} B_{l,k}(\tau, l : l+k) \end{aligned} \tag{4.4}$$

See Appendix B for a discussion of B-spline construction, terminology, and properties. For the current work, each spline basis in Equation (4.4) is distributed on the same uniform knot sequence:

$$S_v = S_z = S_\theta = \{0^6, 1/10, 2/10, \ldots, 9/10, 1^6\}, \tag{4.5}$$

where the exponent 6 at 0 and 1 denote endpoint knots of multiplicity 6, thus allowing discontinuities from zero (i.e. nontrivial trim conditions) at the beginning and end of any maneuver. In addition, assign each spline in Equation (4.4) order $k = 6$, meaning that each basis function $B_{\cdot,k}$ is actually a continuous polynomial of order 5. These selections of knot sequence and order are sufficiently rich to describe interesting maneuvers without inflating the problem order excessively. The consequence is that each expansion in Equation (4.4) therefore has 15 terms:

$$n_v = n_z = n_\theta = 15. \tag{4.6}$$

A reasonable finite behavior parameter vector p is then the concatenation of the spline coefficients $c_{\cdot,\cdot}$ with the trajectory duration T to give:

$$p = \left[\{c_{i,v}\}_{i=1}^{n_v}, \{c_{j,z}\}_{j=1}^{n_z}, \{c_{l,\theta}\}_{l=1}^{n_\theta}, T\right]^T, \tag{4.7}$$

where the *superscript* T denotes a column vector. This vector has 15 components to describe the shape of each system DOF as well as a scalar time parameter to control the maneuver duration, making for a total vector $p \in R^{46}$.

As seen in Equations (4.2) and (4.4), it is sometimes useful to give the standard time argument t and other times the normalized argument τ. In the sequel, when discussing a system signal, the specific argument will either be explicit or will be clear from context. Note that differentiation of system signals with respect to time must be properly described. A standard dot notation denotes a derivative with respect to t, as in $\dot{y}$, and a prime superscript denotes differentiation respect to τ, as in y'. These derivatives scale according to the chain rule relations

$$\begin{aligned} \dot{y}(t) &= \frac{1}{T}y'(\tau) \\ \ddot{y}(t) &= \frac{1}{T^2}y''(\tau) \\ &\vdots \end{aligned} \tag{4.8}$$

since $\tau = t/T$. In Equation (4.8), the left-hand sides are evaluated at some $t \in [0, T]$ while the right-hand are evaluated at the corresponding $\tau = t/T \in [0, 1]$.

4.2.2 Solution for Input Signals

Given the system behavior parametrizations of Equations (4.2) and (4.4), it is a fairly simple matter to invert the system dynamics in Equations (4.1) and solve for the two input voltages. The inversion process is fairly easy given that all three system degrees-of-freedom v, z, and θ are "free" variables in terms of p. If the system behavior $\mathbf{z}$ contained only z and θ, for instance, solution for V_{coll} would first require a numerical *forward* integration to obtain v. This procedure is undesirable since it is generally advantageous for real-time computing methods to have analytical, noniterative expressions for the control inputs in terms of p. Similarly, a choice of $\mathbf{z}$ containing only v and z would require solution of complicated simultaneous inverse trigonometric expressions for θ, again precluding the possibility of realistic analytic inversion.

Given the choice of $\mathbf{z} = [v, z, \theta]^T$, an expression for the collective voltage on the unit time scale follows easily from manipulations of the fourth line of Equation (4.1), substituting standard time derivatives for their normalized-time equivalents:

$$V_{coll}(\tau; p) = \sqrt{\frac{(1/T^2)z'' + (1/T)z' - d_2 cos(z) + d_3 sin(z) + d_5 v^2}{-d_4 cos(\theta)}}. \tag{4.9}$$

This expression is well-defined when $\theta \neq \pm\pi/2$ rad, that is, when the helicopter is not oriented vertically up or down. The $(\tau; p)$ arguments appear on the left-hand side to emphasize that the control input time history is explicitly a function of the trajectory parametrization p, which includes the maneuver duration T as its last component.

Given this expression for $V_{coll}(\tau; p)$, a simple manipulation of the second line in Equation (4.1) gives the cyclic voltage expression

$$V_{cyc}(\tau; p) = \frac{(1/T^2)\theta'' + b_1(1/T)\theta' + b_2 sin(\theta) - b_0 - b_3(1/T^2)v|v|}{b_4 V_{coll}}, \tag{4.10}$$

which is well-defined for $V_{coll} \neq 0$. As noted in Appendix A, the helicopter hovers at a collective setting of 1.64 V and can operate entirely in the $V_{coll} > 0$ input domain. Motions requiring $V_{coll} \leq 0$ are not encountered in this thesis but could easily be prevented by suitable imposition of control input inequality constraint bounds. It is

worth noting that Equations (4.9) and (4.10) are highly nonlinear expressions in v, z, and θ and therefore in the vector p, preventing the use of linear superposition-type methods for creating parametrized maneuver classes, thus necessitating the methods of Chapter 2.

4.2.3 Trajectory Dynamic Feasibility

Given that $\mathbf{z}$ has three signal components and the helicopter system has only two inputs, it is necessary to impose a continuous-time equality constraint among the "free" quantities $v(t)$, $z(t)$, and $\theta(t)$ (and ultimately as a constraint on the vector p) to ensure dynamic feasibility. Such a constraint was seen as the $h_e(p)$ partition in Equation (2.43) and typically arises when enforcing physical feasibility of candidate p vectors [16] or when expressing nonholonomic path constraint functions, such as those often seen in ground robotic systems [2, 31].

In the present case, begin with a continuous-time constraint function $\hat{h}_e$ of p based on a continuous time function $\hat{f}$ of v, z, and θ evaluated over the entire maneuver duration:

$$\hat{h}_e(t;p) = \hat{f}(v(t;p), z(t;p), \theta(t;p)) = 0 \quad \forall\ t \in [0, T]. \tag{4.11}$$

These functions can then be converted to equivalent unit time functions $\tilde{h}_e$ and $\tilde{f}$ by appropriate substitution of τ for t and introduction of T according to Equations (4.3) and (4.8):

$$\tilde{h}_e(\tau;p) = \tilde{f}(v(\tau;p), z(\tau;p), \theta(\tau;p), T) = 0 \quad \forall\ \tau \in [0, 1]. \tag{4.12}$$

For the nonlinear helicopter system, the desired function $\hat{f}$ follows from elimination of V_{coll}^2 from the second and fourth lines of Equations (4.1), giving a suitable relation containing all three components of $\mathbf{z}$:

$$\hat{f}(v, z, \theta) = d_4(\dot{v} + a_1 v)cos(\theta) - a_2(\ddot{z} + d_1\dot{z} - d_2 cos(z) + d_3 sin(z) + d_5 v^2)sin(\theta - \theta_a). \tag{4.13}$$

Normalizing this relation to unit time gives the (nonlinear) $\tilde{f}$ function:

$$\begin{aligned} \tilde{f}(v, z, \theta, T) \;=\; & d_4\left(\frac{1}{T}v' + a_1 v\right) cos(\theta) \\ & -a_2\left(\frac{1}{T^2}z'' + d_1\frac{1}{T}z' - d_2 cos(z) + d_3 sin(z) + d_5 v^2\right) sin(\theta - \theta_a). \end{aligned} \tag{4.14}$$

Of course, Equations (4.12) and (4.14) are in continuous time and must be downsampled to a finite grid, or mesh, to be useful for nonlinear programming and trajectory interpolation. Therefore, to obtain a finite-dimensional vector constraint function $h_e(p)$, evaluate $\tilde{h}_e$ on a user-selected mesh S_e:

$$\tilde{h}_e(\tau;p) = 0 \quad \forall\ \tau \in S_e \tag{4.15}$$

where some suitable $S_e = \{\tau_1, \ldots, \tau_n\} \subset [0, 1]$ can be found through experimentation. A candidate S_e mesh is not sufficiently dense for a given trajectory vector p if the

control inputs obtained from Equations (4.9) and (4.10) do not forward-integrate through Equations (4.1) to the predicted $v(t;p)$, $z(t;p)$, and $\theta(t;p)$ signals. For the 3-DOF helicopter, the almost-uniform mesh

$$S_e = \{0, 1/30, 1/15, 2/15, \ldots, 14/15, 29/30, 59/60\} \tag{4.16}$$

provided sufficient dynamic consistency (a mesh point at $\tau = 1$ turned out to be redundant given the vehicle final trim state equality constraints covered in the next subsection). Combining the mesh-point evaluations of $\tilde{h}_e$ into a single constraint vector gives the desired h_e function:

$$h_e(p) = \tilde{h}^{S_e}(p) = 0. \tag{4.17}$$

where the intermediate mesh sample function $\tilde{h}^{S_e}$ has definition

$$\tilde{h}^{S_e}(p) \equiv \begin{bmatrix} \tilde{h}_e(\tau_1; p) \\ \vdots \\ \tilde{h}_e(\tau_n; p) \end{bmatrix}. \tag{4.18}$$

Note that in the case of differentially flat systems, where it is possible to find an invertible system behavior $\mathbf{z}$ with exactly as many components as system inputs [48, 105, 106], it is unnecessary to enforce dynamic consistency constraints in the form of Equation (4.11).

4.2.4 Boundary Value Equality Constraints

Now consider the boundary condition equality constraint partition $h_{bc}(p)$ of Equation (2.43). This vector is necessary to specify the initial and final conditions of a given trajectory (or trajectory set) and generally plays the greatest role in defining the fundamental variations within a maneuver class.

In this thesis, all maneuvers begin and end at a vehicle equilibrium, or trim, state. Examination of Equations (4.1) indicates that a steady velocity-elevation pair is sufficient to define the entire vehicle trim state (pitch , collective, and cyclic settings). Let an overbar denote a steady-state quantity and note that, in standard time domain notation, the initial (subscript i) and final (subscript f) vehicle trim states are defined entirely by the following four relations:

$$\begin{aligned} v(0) &= \overline{v}_i \\ v(T) &= \overline{v}_f \\ z(0) &= \overline{z}_i \\ z(T) &= \overline{z}_f. \end{aligned} \tag{4.19}$$

Here, $t = 0$ denotes the beginning of the maneuver while $t = T$ denotes the maneuver end time. A steady trim state requires that the velocity and elevation signals have

zero-valued initial and final time derivatives as well:

$$\begin{aligned} \dot{v}(0) &= 0 \\ \dot{v}(T) &= 0 \\ \dot{z}(0) &= 0 \\ \dot{z}(T) &= 0. \end{aligned} \tag{4.20}$$

Further, at a trim condition, the pitch angle must assume its steady-state value and have zero time derivative; the collective and cyclic voltages must take on their equilibrium values. Expressing these four conditions in standard time at both ends of the maneuver gives

$$\begin{aligned} \theta(0) &= \overline{\theta}_i \\ \theta(T) &= \overline{\theta}_f \\ \dot{\theta}(0) &= 0 \\ \dot{\theta}(T) &= 0 \\ V_{coll}(0) &= \overline{V}_{coll,i} \\ V_{coll}(T) &= \overline{V}_{coll,f} \\ V_{cyc}(0) &= \overline{V}_{cyc,i} \\ V_{cyc}(T) &= \overline{V}_{cyc,f}, \end{aligned} \tag{4.21}$$

where the steady pitch and input voltage values are functions of the steady velocity and elevation, as given by:

$$\begin{bmatrix} \bar{\theta} \\ \bar{V}_{coll} \\ \bar{V}_{cyc} \end{bmatrix} = \eta(\bar{v}, \bar{z}) \tag{4.22}$$

for a trim state mapping η obtained from the steady-state equations of motion.

As noted previously, analytic solution for θ in terms of v and z is not straightforward and requires and iterative numerical solution. To proceed, consider the steady-state version of Equation (4.13), given by

$$a_2 sin(\bar{\theta} - \theta_a)(d_2 cos(\bar{z}) - d_3 sin(\bar{z}) - d_5 \bar{v}^2) + a_1 d_4 \bar{v} cos(\bar{\theta}) = 0, \tag{4.23}$$

and use an iterative root-finding technique to solve for $\overline{\theta}$ [14]. Such methods generally work best when given a reasonable initial solution guess, which in this case follows from a steady-state solution of line 2 in Equation (4.1):

$$\hat{\theta}_0 = \theta_a + sin^{-1}\left(\frac{-a_1 \overline{v}}{a_2 \hat{V}_{coll,0}^2}\right), \tag{4.24}$$

with the collective voltage initial guess $\hat{V}_{coll,0}$ coming from a small-θ approximation

to line 4 of Equation (4.1):

$$\hat{V}_{coll,0} = \sqrt{\frac{d_2 cos(\overline{z}) - d_3 sin(\overline{z}) - d_5 \overline{v}^2}{d_4}}. \tag{4.25}$$

Note that iterative trim state solutions are common in helicopter dynamic modeling, such as the recursive thrust-inflow iterations required to compute a main rotor steady-state condition [88]. Fortunately, numerical solution for $\overline{\theta}$ is only necessary when generating feasible example trajectories, not during the actual maneuver interpolation process, since in this latter setting, only differential expressions for $\overline{\theta}$ are required, and follow from an analytic implicit differentiation of Equation (4.23).

Proceeding to the steady voltage calculations, once the quantity $\overline{\theta}$ is available from solution of Equation (4.23), $\overline{V}_{coll}$ and $\overline{V}_{cyc}$ follow from the steady-state standard time analogs of Equations (4.9) and (4.10):

$$\bar{V}_{coll} = \sqrt{\frac{d_2 cos(\bar{z}) - d_3 sin(\bar{z}) - d_5 \bar{v}^2}{d_4 cos(\bar{\theta})}}, \tag{4.26}$$

and

$$\bar{V}_{cyc} = \frac{b_2 sin(\bar{\theta}) - b_0 - b_3 \bar{v}|\bar{v}|}{b_4 \bar{V}_{coll}}. \tag{4.27}$$

Combining the initial and final vehicle boundary conditions of Equations (4.19), (4.20), and (4.21) into a single constraint vector $\hat{h}_{bc}$ gives

$$\hat{h}_{bc}(p) \equiv \begin{bmatrix} v(0;p) - \overline{v}_i \\ v(T;p) - \overline{v}_f \\ z(0;p) - \overline{z}_i \\ z(T;p) - \overline{z}_f \\ \theta(0;p) - \overline{\theta}_i \\ \theta(T;p) - \overline{\theta}_f \\ V_{coll}(0;p) - \overline{V}_{coll,i} \\ V_{coll}(T;p) - \overline{V}_{coll,f} \\ V_{cyc}(0;p) - \overline{V}_{cyc,i} \\ V_{cyc}(T;p) - \overline{V}_{cyc,f} \\ \dot{z}(0;p) - 0 \\ \dot{z}(T;p) - 0 \\ \dot{\theta}(0;p) - 0 \\ \dot{\theta}(T;p) - 0 \end{bmatrix} = 0, \tag{4.28}$$

where the p notation is included to differentiate the quantities that depend on the free vector p from those that are constants depending on the 4-tuple $(\overline{v}_i, \overline{v}_f, \overline{z}_i, \overline{z}_f)$. Note that $\dot{v}(0) = 0$ and $\dot{v}(T) = 0$ trim conditions are excluded since they proved redundant, given the trim voltage constraints.

Conversion of the above vector to normalized time gives the desired constraint relation:

$$h_{bc}(p) \equiv \begin{bmatrix} v(0;p) - \overline{v}_i \\ v(1;p) - \overline{v}_f \\ z(0;p) - \overline{z}_i \\ z(1;p) - \overline{z}_f \\ \theta(0;p) - \overline{\theta}_i \\ \theta(1;p) - \overline{\theta}_f \\ V_{coll}(0;p) - \overline{V}_{coll,i} \\ V_{coll}(1;p) - \overline{V}_{coll,f} \\ V_{cyc}(0;p) - \overline{V}_{cyc,i} \\ V_{cyc}(1;p) - \overline{V}_{cyc,f} \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \end{bmatrix} = 0. \tag{4.29}$$

Note that the last eight rows of this equation are nonlinear in p. In Section 4.5, a maneuver parameter α, typically equal to an initial or final velocity and/or elevation, will be introduced in Equation (4.29) to create a parametrized maneuver class. Together with $h_e(p)$ from Equation (4.17), $h_{bc}(p)$ forms the needed equality constraint set to design interesting vehicle trajectory sets.

4.2.5 State and Control Inequality Constraints

Finally, in addition to the equality constraints, it is useful to impose a vector inequality constraint $g(p) \leq 0$ as discussed in Section 2.3.5. For the three degree-of-freedom helicopter, the useful continuous time bounds on system states include upper and lower bounds on the elevation variable

$$z_{min} \leq z(t;p) \leq z_{max} \quad \forall\ t \in [0,T] \tag{4.30}$$

preventing contact with both the supporting tabletop (at $z \approx 25$ degrees) and the upper mechanical restraints (at $z \approx -37$ degrees). Recall that z is positive *down*wards from a level horizontal origin. It is also useful to bound the pitch deflection to avoid singularity in Equation (4.9) and prevent the motor platform from hitting its hard mechanical limits, which occur around ± 90 degrees:

$$\theta_{min} \leq \theta(t;p) \leq \theta_{max} \quad \forall\ t \in [0,T]. \tag{4.31}$$

Particularly useful are constraints on the input voltage, especially given the highly nonlinear expressions in Equations (4.9) and (4.10). Here, the continuous standard time inequalities take the form

$$V_{coll,min} \leq V_{coll}(t;p) \leq V_{coll,max} \quad \forall\ t \in [0,T] \tag{4.32}$$

and

$$V_{cyc,min} \leq V_{cyc}(t;p) \leq V_{cyc,max} \quad \forall\ t \in [0,T]. \tag{4.33}$$

For now, the specific "min" and "max" bounds are given symbolically for generality. The specific values typically will depend on the particular maneuver class of interest.

To be useful for nonlinear programming and trajectory interpolation, it is necessary to consider Inequalities (4.30) through (4.33) on the unit time interval and sample them on a discrete mesh, as practiced with the dynamic feasibility equality constraint in Equation (4.17). Consider user-selected time meshes $S_{i,1}$ through $S_{i,4}$ for each of the four inequality constraints above, obtaining the following discrete bounds

$$\begin{array}{rcll} z_{min} \leq & z(\tau;p) & \leq z_{max} & \forall\ \tau \in S_{i,1} \subset [0,1] \\ \theta_{min} \leq & \theta(\tau;p) & \leq \theta_{max} & \forall\ \tau \in S_{i,2} \subset [0,1] \\ V_{coll,min} \leq & V_{coll}(\tau;p) & \leq V_{coll,max} & \forall\ \tau \in S_{i,3} \subset [0,1] \\ V_{cyc,min} \leq & V_{cyc}(\tau;p) & \leq V_{cyc,max} & \forall\ \tau \in S_{i,4} \subset [0,1]. \end{array} \tag{4.34}$$

For the maneuvers of interest in this thesis, it is generally sufficient to consider an identical 21-point uniform mesh of $[0,1]$ for each of the inequality constraint types, that is, take

$$S_{i,1} = S_{i,2} = S_{i,3} = S_{i,4} = \{0, 1/20, \ldots, 1\}. \tag{4.35}$$

Naturally, there is a trade-off in mesh density between the ability to approximate the continuous time (infinite-dimensional) bounds of Inequalities (4.30) through (4.33) and the dimensionality of the trajectory generation and interpolation algorithms.

To place the eight sampled upper and lower bounds of Inequalities (4.34) into the required vector format, simply choose the vector function $g(p)$ defined as

$$g(p) \equiv \begin{bmatrix} z_{min} - z^{S_{i,1}}(p) \\ z^{S_{i,1}}(p) - z_{max} \\ \theta_{min} - \theta^{S_{i,2}}(p) \\ \theta^{S_{i,2}}(p) - \theta_{max} \\ V_{coll,min} - V_{coll}^{S_{i,3}}(p) \\ V_{coll}^{S_{i,3}}(p) - V_{coll,max} \\ V_{cyc,min} - V_{cyc}^{S_{i,4}}(p) \\ V_{cyc}^{S_{i,4}}(p) - V_{cyc,max} \end{bmatrix} \leq 0, \tag{4.36}$$

where the superscript set notation has the same meaning as in Equations (4.17) and (4.18). Together with the equality constraint vector $h(p) = [h_e(p); h_{bc}(p)] = 0$, the inequality expression $g(p) \leq 0$ defines a feasible vehicle trajectory space and forms the foundation for defining parametrized maneuver classes. This chapter now continues with a discussion of generating feasible example motions, tracking trajectories on the experimental apparatus, and then presenting examples of interpolated maneuvers.

4.3 Example Trajectory Generation

With the helicopter trajectory parametrization and constraint functions in place, it is now possible to generate example motions as the starting points for maneuver interpolation. Recall from Chapters 2 and 3 that, within a given maneuver type, the example motions are allowed to be any related feasible trajectories, whether optimal or suboptimal. This section briefly discusses two of the most useful techniques for creating example trajectories: nonlinear programming and piloted-flight motion capture. The motivation for these methods was laid out in earlier chapters, so the emphasis here is on application to the three degree-of-freedom helicopter.

4.3.1 Nonlinear Programming

One very useful method for creating example trajectories is off-line nonlinear programming (NLP). The considerable computational resources available in NLP solvers can be applied to rigorously design prototype motions that then define entire maneuver classes. As the trajectory interpolation algorithms do not employ an explicit objective function, but instead use a projection method to form families in the *spirit* of the examples, nonlinear optimization is an opportunity to find prototype maneuvers meeting some mathematical engineering objective.

As noted in Section 2.1.1, the generic trajectory nonlinear program takes the form

$$\begin{aligned} & \min_{p} f(p) \\ \text{s.t.} \quad & \begin{array}{l} h(p) = 0 \\ g(p) \leq 0 \end{array} \end{aligned} , \tag{4.37}$$

where f is the objective function, and h and g are the equality and inequality constraint functions, respectively, all defined in terms of the vector p. Sections 4.2.3 and 4.2.4 demonstrated how to create the equality constraint set for the helicopter, where the h in Program (4.37) is the concatenation of the dynamic feasibility and boundary condition partitions:

$$h(p) = \begin{bmatrix} h_e(p) \\ h_{bc}(p) \end{bmatrix} . \tag{4.38}$$

The vector $h_e(p)$ comes from Equations (4.17) and (4.18); the vector $h_{bc}(p)$ follows from Equation (4.29). The inequality constraint vector g is defined in Equation (4.36).

Choice of objective function depends on what quality of maneuvers are of interest. In this thesis, agile and aggressive motions are most often considered, since they are typically useful for autonomous vehicles in dynamic urban settings or in time-critical reconnaissance and search-and-rescue missions. Such rapid example motions typically follow from a minimum-time objective function:

$$f(p) = e_N^T p = T \tag{4.39}$$

where e_N is a unit vector with unity as its Nth component, with $N = n_v + n_z + n_\theta + 1$. Naturally, to keep minimum time solutions well-posed, bounded control is required,

where the specific numeric bounds are set in Inequality (4.36).

As general practice in this document, the control bounds during nonlinear programming example generation are set slightly tighter than those employed when using Algorithm 3 of Section 2.3.5 for maneuver class interpolation. This technique generally allows the projection algorithm the greatest freedom in choosing the initial feasible path towards other known motions, preventing excessive numbers of initially active inequality constraints (which function as strict equalities) from reducing the projection dimension and increasing the algorithm computational load (see Section 3.2). However, as seen in Section 3.1.4, if the projection interpolation path immediately leads into the bounded control constraints, they will be added back into the local active set.

Most nonlinear programming solvers allow for an initial solution guess $\hat{p}_0$, which typically has a large effect on the final optimal solution p^*, given the presence of local minima in general nonlinear programming settings. In the majority of cases, an initial guess $\hat{p}_0$ with *linear* state transitions between the initial and final trim conditions is adequate for eventual convergence to agile motions [30, 102]. However, other methods for choosing $\hat{p}_0$ may be advantageous in some settings and can be obtained, for example, by initial polynomial profiles of a candidate system behavior $\mathbf{z}(t; \hat{p}_0)$, or by choosing $\hat{p}_0$ to match a "sketch" of the desired solution.

For the helicopter maneuvers encountered in this thesis, solution of Program (4.37) involves a parameter vector p with $\dim(p) = 46$, an equality constraint vector with $\dim(h(p)) \approx 30$ and an inequality constraint vector with $\dim(g(p)) \approx 168$. NLP solution times varied widely depending on the particular maneuver of interest and the closeness of the linear initial guess to the ultimate locally optimal solution p^*. However, the majority of solution times seemed to fall between one minute and ten minutes on a Pentium III 930 MHz processor running the MATLAB® version 2.2 optimization toolbox [97]. These solution times, while essentially insignificant for off-line trajectory generation are certainly too long for any real-time implementation, thus helping to motivate the need for faster methods such as trajectory interpolation.

Section 4.5 will discuss specific maneuver instances and examples of introducing a class parameter α into the boundary condition equality constraint function, as seen in Equation (2.43). Then, given the choice and desired range of variation in α, specific nonlinear programs of the form (4.37) for creating prototype maneuvers are evident.

4.3.2 Motion Capture

Section 2.4 discusses a two-step procedure for capturing human pilot-flown flight data as a model feasible point. This method gives a viable alternative to pure NLP as a means of creating example trajectories; note that the second step does use a NLP solution, however one whose objective function attempts to match an observed pilot-flown system behavior. Section 4.5 will illustrate two parametrized maneuver families created from pilot data. In each case, two bounding motions for trajectory interpolation were found through the methods of Section 2.4.

The three helicopter signals of interest during motion capture are velocity, elevation, and pitch angle, the exact three components of the system behavior vector $\mathbf{z}$.

Given the definition of the trajectory parametrization p in Equation (4.7), the motion capture task is to find a model-feasible p vector, closely matching a system behavior expressed in flight data $\mathbf{z}_{data}$.

Following the first step of Section 2.4, the initial task is to obtain a reasonable estimate vector $p_{data,0}$ based purely on a data matching procedure, without regard to the underlying system dynamics. Given that the free vector p contains the maneuver time T as its last element, the corresponding element of $p_{data,0}$ is set exactly to match the duration of the observed pilot-trajectory, that is, $T = T_{data}$. Then, treating each of the three signal partitions of p individually and in succession, a least squares data matching objective of the form of Equation (2.52), applied with a set of unit-time boundary condition equality constraints, fills out the remaining elements of $p_{data,0}$. The solution procedure for these signal partitions, say $p^v_{data,0}$, $p^z_{data,0}$, and $p^\theta_{data,0}$, follows from a nonlinear least squares procedure. The boundary condition constraints are user-selections employed to enforce the desired trim signal values, as seen in Equation (4.29). These trim conditions are known from the maneuver type under consideration and the specific maneuver instance α_{data} being captured, which follows from the maneuver class descriptor α.

The final step is to use this data matching initial guess

$$p_{data,0} = \begin{bmatrix} p^v_{data,0} \\ p^z_{data,0} \\ p^\theta_{data,0} \\ T_{data} \end{bmatrix} \tag{4.40}$$

as the starting point of a model feasibility NLP in the form of Program (2.53), repeated here:

$$\begin{aligned} &\min_p e(p) \\ \text{subject to} \quad &h(p, \alpha_{data}) = 0 \\ &g(p) \leq 0 \end{aligned} . \tag{4.41}$$

Of the two proposed objective functions in Equations (2.54) and (2.55), the latter, which attempts to match the recorded behavior data seems to work best for the 3-DOF helicopter system. That objective,

$$e(p) = \sum_{k=0}^{N_k} ||z_{data}(t_k) - z(t_k; p)||_{W_k}, \tag{4.42}$$

seeks a feasible point p that closely matches the motion capture data, which in this case is 100 Hz samples of velocity, elevation, and pitch signals. Since the data sample interval is fixed, the number of data points N_k depends on the maneuver duration. A fixed, diagonal weighting matrix $W_k = \text{diag}(3, 3, 1)$ is useful for normalizing the error between the three measured signals (whose numerical values vary over different ranges) and is found to produce satisfactory fitting results

Sections 4.5.4 and 4.5.5 present results of the motion capture procedure and then

apply trajectory interpolation to create new maneuver instances.

4.4 Tracking Controller Design

The trajectory parametrization p has been selected in Equation (4.7) so that knowledge of p and the helicopter model implies complete knowledge of both the system behavior $\mathbf{z}(t;p) = [v(t;p), z(t;p), \theta(t;p)]^T$ and the voltage inputs $V_{coll}(t;p)$ and $V_{cyc}(t;p)$. Since the system signals exist in terms of the simple B-spline basis elements of Equation (4.4), knowledge of related system signals $x(t;p)$, $\dot{\theta}(t;p)$, and $\dot{z}(t;p)$ is available through simple analytic integrations or differentiations (see [34, 35] and Appendix B). These collected time domain signals provide a complete reference input and state history, essential for a feedforward maneuver tracking control design. With dynamically feasible input signals available, the control system's burden becomes not one of error-driven command following [108], but instead one of regulating fairly small state perturbations and compensating for inevitable nonlinear modeling errors. This section reviews the fundamentals of a gain-scheduled linear-quadratic (LQ) servo design [26, 93] for tracking reference maneuvers produced by trajectory interpolation. The first subjection reviews the linearized model; the second section discusses the LQ design itself.

4.4.1 Linearized Models

In this thesis, parametrized maneuver families by definition involve a nonlinear system that cannot be approximated by a single linearized state-space model. However, for any given maneuver, a dynamically feasible input-output reference is available, making it possible to design a set of LQ-servos scheduled over a wide range of system operating conditions. Each of these servo designs requires a locally accurate linear system model.

Examining the nonlinear helicopter equations of motion (4.1), a reasonable system state $\mathbf{x}$ and control input $\mathbf{u}$ for a first-order linear state-space model are given by:

$$\mathbf{x} \equiv \begin{bmatrix} x \\ v \\ \theta \\ \dot{\theta} \\ z \\ \dot{z} \end{bmatrix} \qquad \mathbf{u} \equiv \begin{bmatrix} V_{coll} \\ V_{cyc} \end{bmatrix}. \tag{4.43}$$

For a linearized model, it is necessary to define a perturbed state $\delta\mathbf{x}(t) \equiv \mathbf{x}(t) - \mathbf{x}_{cmd}$ and perturbed input $\delta\mathbf{u}(t) \equiv \mathbf{u}(t) - \mathbf{u}_{cmd}$ relative to some known state $\mathbf{x}_{cmd}$ and dynamically consistent control input $\mathbf{u}_{cmd}$. Each component of these difference vectors is then a perturbed version of the corresponding components in $\mathbf{x}$ and $\mathbf{u}$, as in

$$\delta\mathbf{x} = \begin{bmatrix} \delta x \\ \delta v \\ \delta\theta \\ \delta\dot{\theta} \\ \delta z \\ \delta\dot{z} \end{bmatrix} \qquad \delta\mathbf{u} = \begin{bmatrix} \delta V_{coll} \\ \delta V_{cyc} \end{bmatrix}. \tag{4.44}$$

Recalling from Section 4.2.4 that a steady system state is completely defined by a steady pair $(\bar{v}, \bar{z})$, the linearized state-space matrices are then functions the steady velocity and elevation, as in

$$\dot{\delta\mathbf{x}} = A(\bar{v}, \bar{z})\delta\mathbf{x} + B(\bar{v}, \bar{z})\delta\mathbf{u}. \tag{4.45}$$

By analytically linearizing Equations (4.1) and using the steady pitch and voltage relations of Equations (4.22) through (4.27), the linearized state and input matrices follow easily as

$$A(\bar{v}, \bar{z}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -a_1 & -a_2\bar{V}_{coll}^2 cos(\bar{\theta} - \theta_a) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2b_3|\bar{v}| & -b_2 cos(\bar{\theta}) & -b_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2d_5\bar{v} & d_4\bar{V}_{coll}^2 sin(\bar{\theta}) & 0 & -d_2 sin(\bar{z}) - d_3 cos(\bar{z}) & -d_1 \end{bmatrix} \tag{4.46}$$

and

$$B(\bar{v}, \bar{z}) = \begin{bmatrix} 0 & 0 \\ -2a_2\bar{V}_{coll} sin(\bar{\theta} - \theta_a) & 0 \\ 0 & 0 \\ b_4\bar{V}_{cyc} & b_4\bar{V}_{coll} \\ 0 & 0 \\ -2d_4\bar{V}_{coll} cos(\bar{\theta}) & 0 \end{bmatrix}. \tag{4.47}$$

These matrices then form the basis for a control design scheduled on any desired partition of the vehicle flight envelope.

4.4.2 Gain-Scheduled LQ-Servo Design

Given the linearized state-space matrices of Equations (4.46) and (4.47) and a partitioning of the flight envelope, it is possible to design a gain-scheduled LQ-servo loop based on standard multivariable feedback design principles [26, 93]. The servo design takes the form shown in Figure 4-2, where the control loop essentially regulates state perturbations $\delta\mathbf{x}$ around the commanded maneuver state trajectory $\mathbf{x}_{cmd}$ using control perturbations $\delta\mathbf{u}$ about the dynamically consistent commanded control signal $\mathbf{u}_{cmd}$.

To help minimize error in tracking the reference position, velocity, and elevation

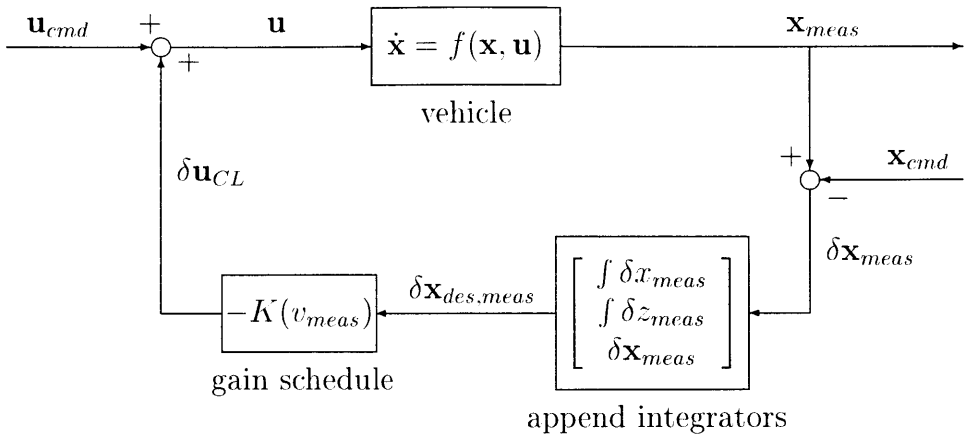


Figure 4-2: Closed-loop LQ-servo design.

states, the controller uses an augmented system design state $\delta\mathbf{x}_{des}$ given by

$$\delta\mathbf{x}_{des} = \begin{bmatrix} \int \delta x \\ \int \delta z \\ \delta x \\ \delta v \\ \delta\theta \\ \delta\dot{\theta} \\ \delta z \\ \delta\dot{z} \end{bmatrix}, \tag{4.48}$$

where the $\int \delta x$ and $\int \delta z$ denote integrator states on travel position and elevation. The design state matrices are then augmented forms of those given in Equations (4.46) and (4.47) with additional rows and columns to account for the integrators:

$$A_{des} = \begin{bmatrix} 0_{2x2} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\ 0_{6x2} & A(\overline{v}_{des}, \overline{z}_{des}) \end{bmatrix} \qquad B_{des} = \begin{bmatrix} 0_{2x2} \\ B(\overline{v}_{des}, \overline{z}_{des}) \end{bmatrix}. \tag{4.49}$$

Here, $(\overline{v}_{des}, \overline{z}_{des})$ denotes a velocity-elevation setpoint design pair. The local full-state feedback gain K comes from a standard linear quadratic regular (LQR) Riccati solution, minimizing the infinite time horizon cost functional

$$J = \int_0^\infty (\delta\mathbf{x}_{des}^T Q \delta\mathbf{x}_{des} + \delta\mathbf{u}^T R \delta\mathbf{u}) dt. \tag{4.50}$$

In this thesis, the state and control weighting matrices, Q and R, respectively, are specified in a simple diagonal form: $Q = \text{diag}(q_{11}, \ldots, q_{88})$ and $R = \text{diag}(r_{11}, r_{22})$. The specific scalars q_{ii} and r_{jj} depend on the particular design point and follow from

an iterative design process of gain calculation, response evaluation, and numerical revision. The weights are generally chosen to provide reasonable tracking of velocity v and/or elevation z, with less emphasis placed on tracking pitch θ and the time derivatives $\dot{\theta}$ and $\dot{z}$.

As shown in Figure 4-2, when operating in closed-loop, the system state measurement $\mathbf{x}_{meas}$ is differenced with the reference state trajectory $\mathbf{x}_{cmd}$ to obtain the *measured* perturbed state according to

$$\delta\mathbf{x}_{meas} = \mathbf{x}_{meas} - \mathbf{x}_{cmd}. \tag{4.51}$$

The helicopter experimental apparatus allows direct measurement of x, z, and θ from which simple and reliable estimates of v, $\dot{z}$, and $\dot{\theta}$ follow from transfer function approximations of continuous-time differentiators cascaded with low-pass filters to reduce noise. In the current experimental configuration, this state measurement approach provides performance comparable to Kalman filtering and is implemented to avoid the necessity of scheduling an additional set of estimator gains.

Continuing with the figure, the measured perturbed state is then augmented through two integrators on position and elevation, obtaining the measured *design* state

$$\delta\mathbf{x}_{des,meas} = \begin{bmatrix} \int \delta\mathbf{x}_{meas,1} \\ \int \delta\mathbf{x}_{meas,5} \\ \delta\mathbf{x}_{meas} \end{bmatrix} = \begin{bmatrix} \int \delta x_{meas} \\ \int \delta z_{meas} \\ \delta\mathbf{x}_{meas} \end{bmatrix}, \tag{4.52}$$

where the subscripts 1 and 5 denote the first (position) and fifth (elevation) elements of $\delta\mathbf{x}_{meas}$. The closed-loop control $\delta\mathbf{u}_{CL}$ then follows by a gain-scheduled multiplication of the design state, expressed in equation form as

$$\begin{aligned} \mathbf{u} &= \mathbf{u}_{cmd} + \delta\mathbf{u}_{CL} \\ &= \mathbf{u}_{cmd} - K(v_{meas})\delta\mathbf{x}_{des,meas}. \end{aligned} \tag{4.53}$$

Note from the figure and from line 2 of Equation (4.53) that the gain is scheduled on velocity only and thus the actual K matrix in closed-loop depends only on the currently measured velocity v_{meas}. Experimentation with various flight envelope partitions showed that it is sufficient to use a fixed, zero elevation design value $z_{des} = 0$ while varying the design velocity v_{des} over a series of nine set points given by $\{-80, -60, -40, -20, 0, 20, 40, 60, 80\}$ deg/sec. During on-line closed-loop operation, changes between the nine servo designs occurred at the switching points $v_{meas} = \{-70, -50, -30, -10, 10, 30, 50, 70\}$ deg/sec with a 4 deg/sec hysteresis bound placed about each switching point to prevent control system limit-cycling.

4.5 Maneuver Class Examples

The preceding sections of this chapter demonstrated trajectory parametrization and constraint formulation for the helicopter testbed, similar to the framework one might employ for a full-scale autonomous vehicle. With the mathematical tools of a spline

signal basis set, an invertible system model, and a well-defined feasible space in hand, it is now possible to give examples of continuously parametrized maneuver classes. This section presents four such examples, illustrating the applicability of the maneuvering framework to useful trajectories common in real-world helicopter flight. The first two cases, a flaring quick-stop and an elevation change maneuver, interpolate example motions generated from nonlinear programming. The second two cases, a hover-to-hover position change and a hover-to-cruise dash-acceleration motion, use prototypes obtained from motion capture of human-piloted examples. In each of these cases, an intuitive scalar maneuver class parameter α describes a variable initial or final boundary condition. In Chapter 5, where maneuver classes are integrated into a hybrid system model to create a flexible motion planning scheme, the α variable will be used to capture multiple and/or simultaneous variations in initial and final trim states.

4.5.1 Bounded-Control Quick-Stops

A quick-stop maneuver involves transitioning the helicopter from a steady cruise state to a steady, resting hover state. In this section, choose $\alpha = \overline{v}_i$ to create a maneuver class continuously parametrized by initial trim speed $\overline{v}_i$. To complete the initial trim state, take $\overline{z}_i = 0$ to be fixed, so the helicopter starts at a "level" elevation. Define a final hover state with velocity $\overline{v}_f = 0$ and $\overline{z}_f = 0$, so the vehicle comes to rest at a level elevation. During the maneuver, the helicopter feedforward controls must apply a collective-cyclic combination to pitch the helicopter "backward", develop a strong thrust to slow the vehicle down, and then pitch the vehicle forward again to reach a steady resting state.

Recall that over a general maneuver class, the trajectory equality constraint vector takes the form $h(p) = [h_e(p); h_{bc}(p, \alpha)]$ as seen in Equation (2.43). (The semicolon notation denotes here the vertical concatenation of two column vectors). For all helicopter maneuvers, Equations (4.17) and (4.18) define the $h_e(p)$ model feasibility partition. Now, to create a maneuver class, the boundary constraint partition $h_{bc}(p)$ must include a second argument α and take the form $h_{bc}(p, \alpha)$ as discussed in Section 2.3.3.

Here, given the variable initial velocity $\alpha = \overline{v}_i$, modify the boundary condition constraint definition of Equation (4.29) by introducing α to obtain

$$h_{bc}(p,\alpha) = \begin{bmatrix} v(0;p) - \alpha \\ v(1;p) - 0 \\ z(0;p) - 0 \\ z(1;p) - 0 \\ \theta(0;p) - \overline{\theta}_i(\alpha) \\ \theta(1;p) - \overline{\theta}_{hov} \\ V_{coll}(0;p) - \overline{V}_{coll,i}(\alpha) \\ V_{coll}(1;p) - \overline{V}_{coll,hov} \\ V_{cyc}(0;p) - \overline{V}_{cyc,i}(\alpha) \\ V_{cyc}(1;p) - \overline{V}_{cyc,hov} \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \end{bmatrix} = 0. \tag{4.54}$$

Note that α must appear in each of lines 1, 5, 7, and 9 in Equation (4.54), since as noted in Equation (4.22), the initial trim pitch $\overline{\theta}_i$, trim collective $\overline{V}_{coll,i}$, and trim cyclic $\overline{V}_{cyc,i}$ all depend on the initial trim speed $\alpha = \overline{v}_i$. Therefore, over the continuous quick-stop maneuver class, each of the four vector components will vary in some continuous fashion, according to the behavior of $\eta(\alpha, 0)$. (Note that the third row of Equation (4.54) implies that $z(0;p) = 0$ over the maneuver class, as dictated by the assumption of initial level elevation). The final trim state quantities are fixed at their hover values, according to $[\overline{\theta}_{hov}, \overline{V}_{coll,hov}, \overline{V}_{cyc,hov}]^T = \eta(0,0)$. The inequality constraint vector expression $g(p) \leq 0$ is unaltered by the quantity α since the elevation, pitch, collective and cyclic bounds are fixed over the entire maneuver class.

Creation of the parametrized quick-stop maneuver class follows from application of Algorithm 3 of Section 2.3.5, which requires the computation of the Jacobian matrices $D_v h$ and $D_v g_{J_0}$. Therefore, the equality constraint derivative takes the form

$$D_v h(p,\alpha) = \begin{bmatrix} D_p h_e(p) & 0 \\ D_p h_{bc}(p,\alpha) & D_\alpha h_{bc}(p,\alpha) \end{bmatrix}. \tag{4.55}$$

Any of these derivative matrices can be found numerically by applying small perturbations to the arguments of the constraint functions $h_e(p)$ and $h_{bc}(p,\alpha)$ and then dividing the appropriate forward difference vectors by the perturbation step size [158]. A faster method with less numerical error is to derive the derivative expressions once and then simply reevaluate then during interpolation. This practice generally speeds up the evaluation of $D_v h$ and $D_v g_{J_0}$ by a factor of roughly 3, a considerable savings given that derivative computation is by far the most costly operation in Algorithms 1, 2, and 3.

Analytic derivative expressions for components of $D_p h_e(p)$, $D_p h_{bc}(p,\alpha)$, and $D_p g_{J0}(p)$ follow by applying the spline differentiation rules for nonlinear functions of B-spline coefficients (derived in Section B.2 of Appendix B) to the nonlinear functions in Equations (4.9), (4.10), (4.12), (4.14), (4.18), and (4.36), using the signal spline expansions

of Equations (4.4).

The remaining derivative partition is a vector of the form $D_\alpha h_{bc}(p, \alpha)$, involving derivatives of boundary condition trim values with respect to α. From Equation (4.54), only four components of $D_\alpha h_{bc}$ are nonzero. However, these components are critical to the interpolation process since they fundamentally drive the continuation algorithm along the desired maneuver class.

The partial derivative $\partial\overline{\theta}_i/\partial\alpha = \partial\overline{\theta}_i/\partial\overline{v}_i$ follows from an implicit differentiation of Equation (4.23) keeping in mind that $\overline{z}_i \equiv 0$ over the entire maneuver class. Similarly, the expressions $\partial\overline{V}_{coll,i}/\partial\alpha = \partial\overline{V}_{coll,i}/\partial\overline{v}_i$ and $\partial\overline{V}_{cyc,i}/\partial\alpha = \partial\overline{V}_{cyc,i}/\partial\overline{v}_i$ can be found using Equations (4.26) and (4.27), respectively, while reusing the above result for $\partial\overline{\theta}_i/\partial\overline{v}_i$. Note that since the continuation process is driven by differential curve tracing methods from a known solution, there is no need to iteratively solve Equation (4.23) during the integration process; knowledge of the example feasible trajectories contains all the necessary information about the numeric values of $\overline{\theta}$.

Figures 4-3 and 4-4 show the velocity and pitch profiles, for a class of quick-stops over a range $\alpha = \overline{v}_i \in [-50, -10]$ deg/sec. Horizontal axes give the variable initial velocity $\alpha = \overline{v}_i$ and time t. Vertical axes give the signal values in degree units. (Note that because the helicopter has a twin-rotor symmetric fore-aft design, a negative velocity or position value simply indicates the flight direction sign-convention, and does not imply the system is traveling "backwards"). Two bounding prototype trajectories can be generated by solving minimum time nonlinear programs of the form of Program (4.37) with the equality constraint definition of Equation (4.38) for $\alpha_1 = -10$ deg/sec and $\alpha_2 = -50$ deg/sec. Trajectory interpolation Algorithm 3 then produces the entire family. As seen in Figure 4-3, the initial trim speed varies continuously over the maneuver class with the overall velocity profile showing a smooth transition to a resting hover state. The pitch profile in Figure 4-4 has a smoothly transitioning initial trim value (from $\overline{\theta}_i = 6.5$ deg for $\alpha = -10$ deg/sec to $\overline{\theta}_i = 14.70$ deg for $\alpha = -50$ deg/sec) while demonstrating a substantial flare to reverse the thrust vector horizontal component and decelerate the helicopter.

As seen in the corresponding control profiles of Figures 4-5 and 4-6 (plotted versus normalized time for visual clarity) the collective and cyclic voltage work at the beginning of the maneuver to first pitch the helicopter backwards, then apply a large collective (i.e. thrust) and pitch-sustaining cyclic to reduce velocity, and finally bring the vehicle to an equilibrium hover state. During the interpolation, the collective and cyclic voltages occasionally encounter and ride along the enforced constraint boundaries. Note that following the methods of Section 4.3.1, the bounding example maneuvers are created from nonlinear programming with control limits slightly tighter than those employed during interpolation, giving the projection algorithm maximum freedom in choosing the initial feasible projection path. Note that the pitch signals never approach the enforced ± 88 deg limits.

While Figures 4-5 and 4-6 have a normalized time scale τ, Figures 4-3 and 4-4 have a standard time scale t and demonstrate that over the maneuver class, the quick-stop motions exhibit continuous monotonic variation in time duration. That is, $T(\alpha)$ follows the logical behavior of becoming longer as the initial speed magnitude increases since, with bounded control, it takes longer to make a larger state change.

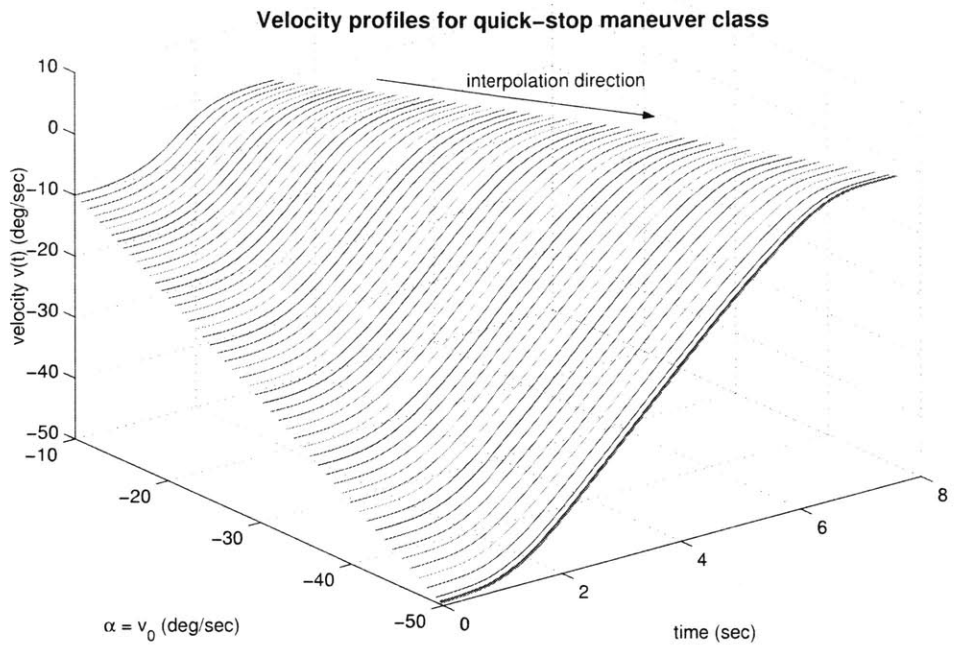


Figure 4-3: Velocity profiles for quick-stop maneuver class.

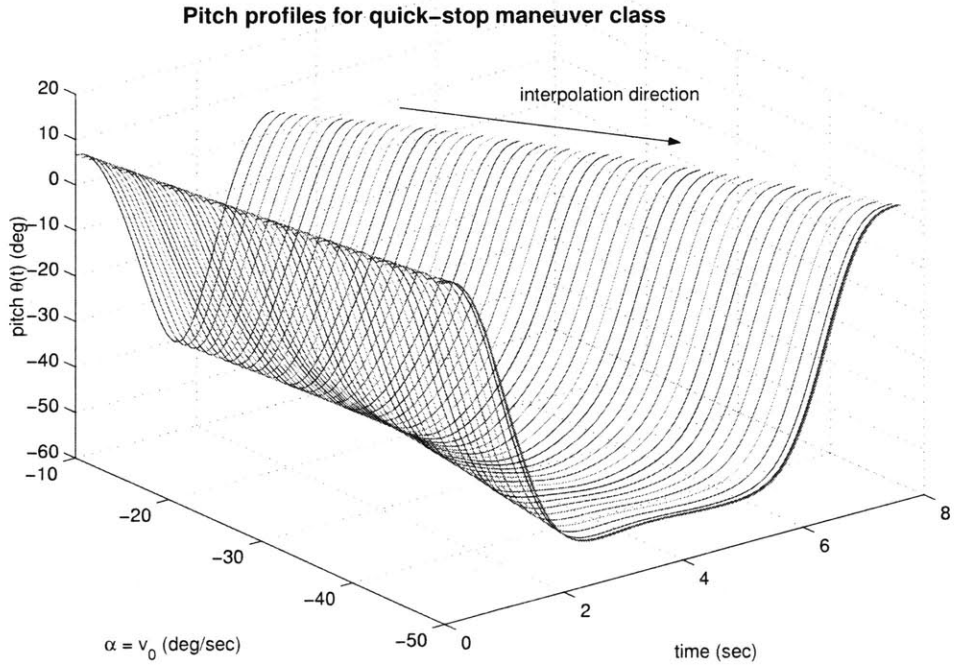


Figure 4-4: Pitch profiles for quick-stop maneuver class.

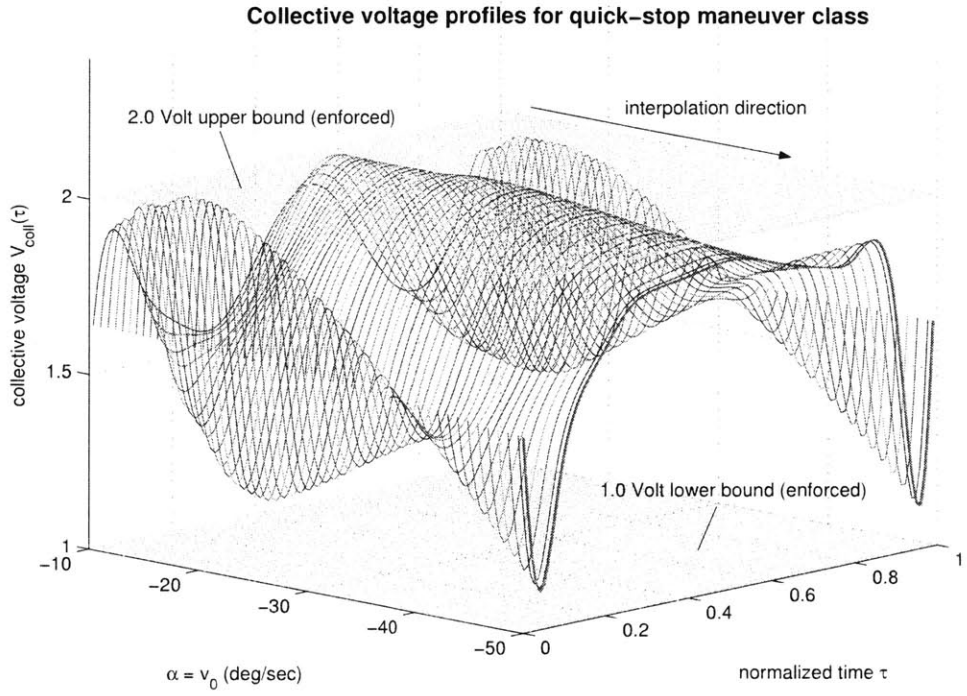


Figure 4-5: Collective voltage profiles for quick-stop maneuver class.

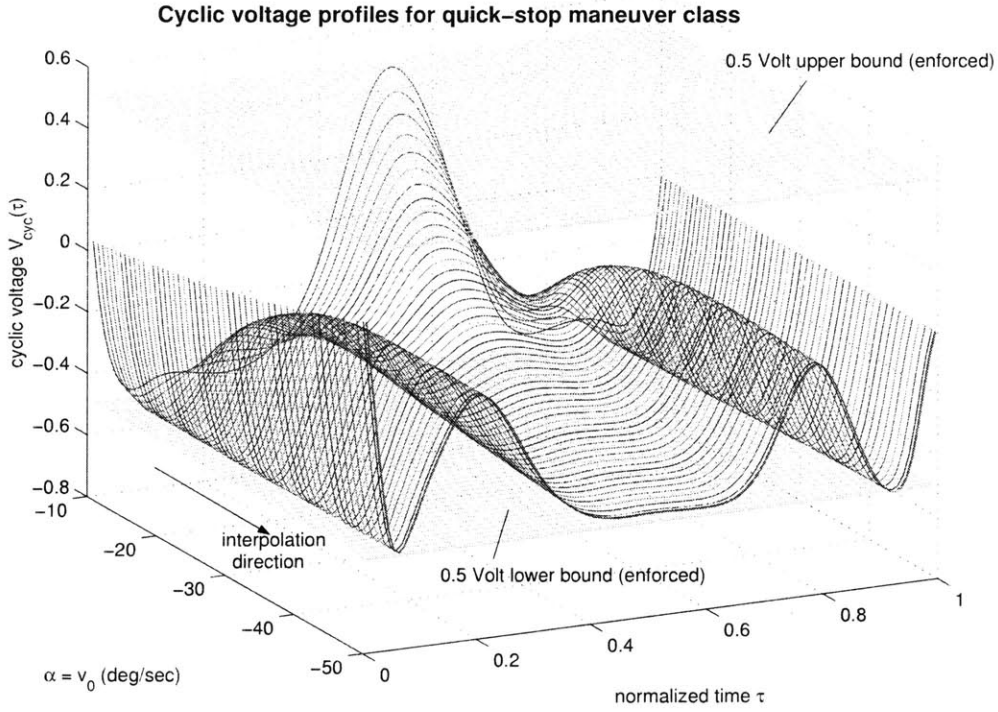


Figure 4-6: Cyclic voltage profiles for quick-stop maneuver class.

4.5.2 Bounded-Control Climbing Maneuver

Now consider a maneuver with the same initial and final velocity, but with variable net elevation change. Specifically, consider $\overline{v}_i = \overline{v}_f = -30$ deg/sec but with $\overline{z}_i = 0$ and $\alpha = \overline{z}_f$ as a variable final boundary condition. This α-parametrization achieves a continuous class of climb (or descent) maneuvers for the helicopter.

As with the previous quick-stop maneuver, the $h_e(p)$ and $g(p)$ constraint vectors are unmodified by the presence of α. However, the boundary condition partition $h_{bc}(p, \alpha)$ must have a second argument as shown in the following equation:

$$h_{bc}(p,\alpha) = \begin{bmatrix} v(0;p) - \overline{v}_i \\ v(1;p) - \overline{v}_f \\ z(0;p) - 0 \\ z(1;p) - \alpha \\ \theta(0;p) - \overline{\theta}_i \\ \theta(1;p) - \overline{\theta}_f(\alpha) \\ V_{coll}(0;p) - \overline{V}_{coll,i} \\ V_{coll}(1;p) - \overline{V}_{coll,f}(\alpha) \\ V_{cyc}(0;p) - \overline{V}_{cyc,i} \\ V_{cyc}(1;p) - \overline{V}_{cyc,f}(\alpha) \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \end{bmatrix} = 0. \tag{4.56}$$

Note that again, α must appear in the expressions for the final pitch, collective, and cyclic settings since, as seen in Equation (4.22), system trim values depend on both the steady velocity and elevation. For the case of an airborne helicopter operating over a small altitude range (and thus with essentially constant air density ρ), there would be no need to modify the final trim settings with α since the vertical dynamics would not depend on altitude. However, for the 3-DOF helicopter of this thesis, the elevation dynamics themselves depend substantially on z, as seen in Equations (4.1). As such, the requirement that $\overline{z}_i = 0$ is important over this class of maneuvers, since a variable initial elevation would require an additional component of α, making for a multidimensional maneuver class. Note that Chapter 5 will introduce a variable initial (and final) elevation maneuver with simultaneous and independent variations allowed in initial (and final) velocity as well.

For the present, consider the variable climb from level flight as defined by the boundary condition of Equation (4.56) over a range $\alpha \in [-35, -10]$ deg. Recall that z is defined positive *down*wards, so a negative elevation change implies a climbing motion. This section uses two types of prototype motions generated by minimum time nonlinear programming solutions. The first is a pair of "ordinary" or "standard" climbs, one for $\alpha_1 = -10$ deg and another for $\alpha_2 = -35$ deg. For the second set of prototypes, a temporary ad-hoc final position boundary constraint can be used to induce a nonstandard, loop-like motion during the climb. This difference will illustrate

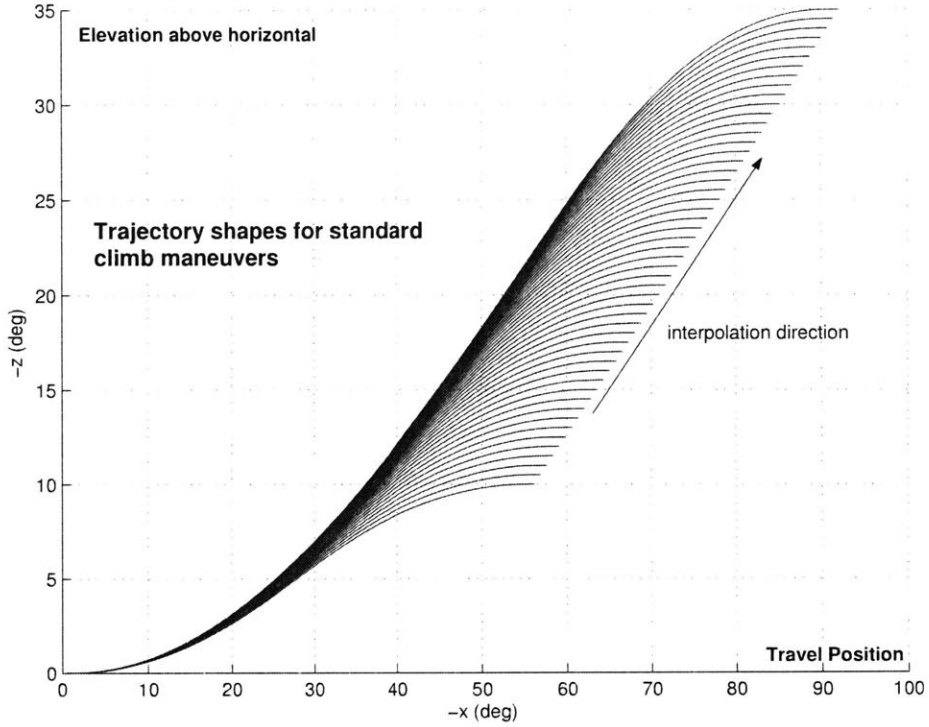


Figure 4-7: Trajectory profiles for parametrized climbing maneuver.

the fundamental point that the outcome of trajectory interpolation depends critically on the "style" of the example motions, giving the engineer tremendous flexibility in designing maneuver classes. The interpolation process then generates continuous trajectory families in the same style of the prototypes; the ad-hoc position constraint used during loop example generation is unnecessary during interpolation since the bounding examples contain all information necessary to define a maneuver style. Note that for all climb maneuvers, analytic partial derivative calculation follows exactly the same lines as discussed in Section 4.5.1, but with the $D_\alpha h_{bc}$ vector now based the defining Equation (4.56).

Figure 4-7 shows several samples of the continuously parametrized "standard" climbing motion, generated using Algorithm 3. The climbs at either extreme came from NLP solutions p_1 and p_2 while all other trajectories followed from interpolation, thus possessing similar position-elevation profiles. (Note that the horizontal axis plots $-x$ and the vertical axis plots $-z$, so that the motions appear in their correct, physical rightward-upward direction). Figures 4-8 and 4-9 show the constrained collective and cyclic control profiles for this maneuver family. The collective profile follows a bang-bang-type strategy, inducing a rapid upward acceleration followed by a deceleration to achieve vertical equilibrium at the new cruise elevation. The cyclic profile essentially keeps the vehicle pitch angle θ (not shown here) near the initial and final trim values and the helicopter velocity close to -30 deg/sec throughout the climbing motion. (Maximum pitch variation over the maneuver class was around 10 deg; maximum velocity variation was around 1.5 deg/sec).

The numerical control bounds are slightly changed relative to the quick-stop maneuver, allowing greater authority in producing upward accelerations. The collective upper bound and cyclic lower bounds are unnecessary during interpolation for this particular maneuver class and are relaxed to speed up computation time. Naturally,

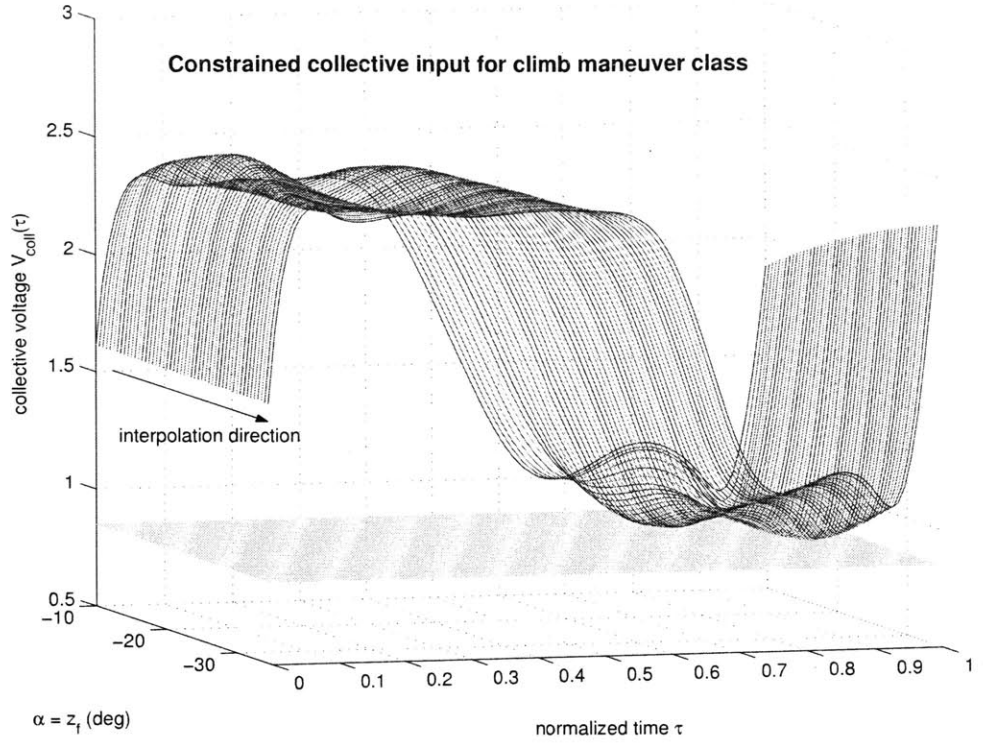


Figure 4-8: Collective voltage input constrained above 0.85 V for climb maneuver class.

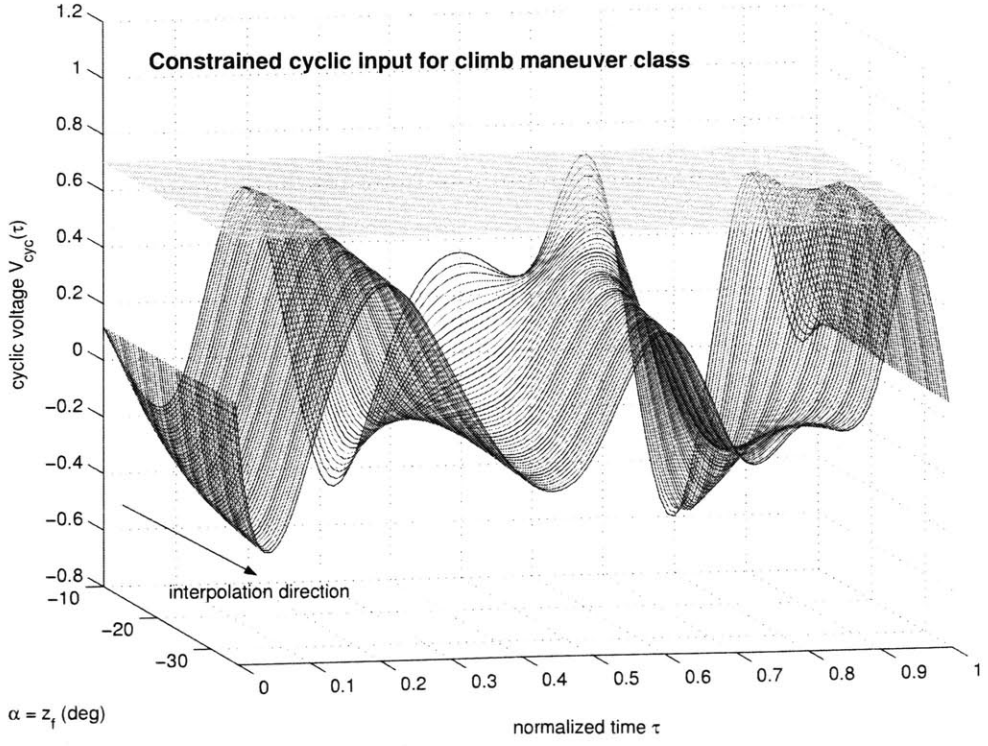


Figure 4-9: Cyclic voltage input constrained below 0.70 V for climb maneuver class.

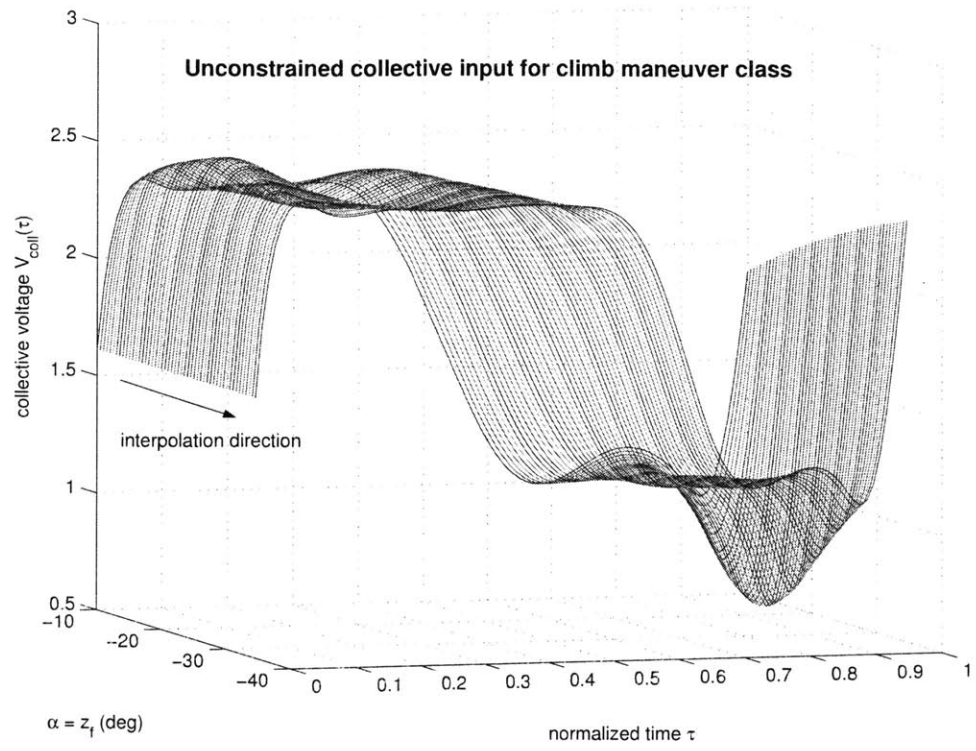


Figure 4-10: Collective voltage input for unconstrained climb maneuver class.

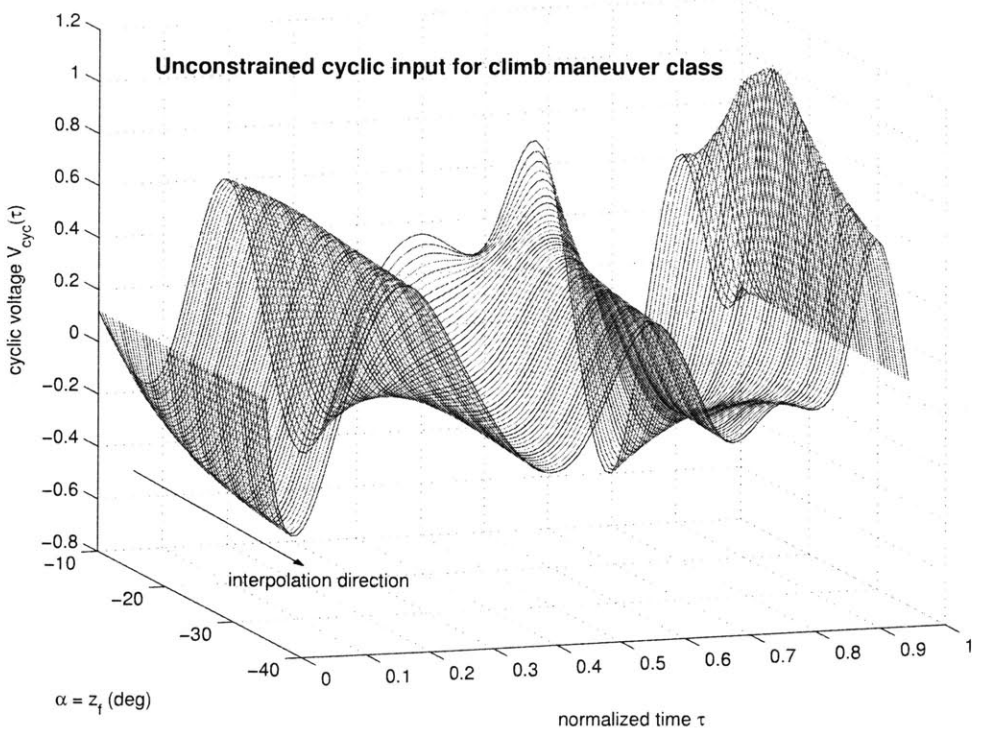


Figure 4-11: Cyclic voltage input for unconstrained climb maneuver class.

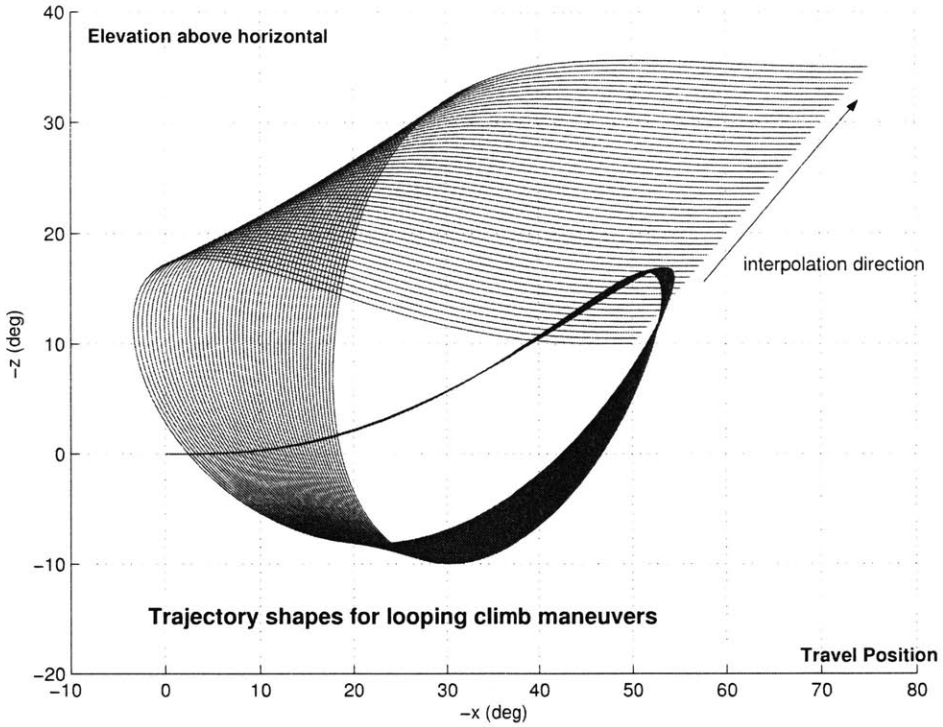


Figure 4-12: Trajectory profiles for looping climb maneuver class.

full upper and lower bounds are necessary during NLP-based example generation to keep the minimum-time solution well-posed. In many situations, because the interpolation process attempts to generate maneuvers in the same style of the examples, control magnitudes stray little over the entire family, allowing reasonable relaxation of some bounds and thus less computational burden, as discussed in Section 3.2.

However, for comparison, Figures 4-10 and 4-11 show the control signal behavior over the maneuver class when the collective and cyclic bounds are lifted. Nonlinearities in the p-to-voltage mappings of Equations (4.9) and (4.10) allow large, unreasonable swells in the control profiles, taking the input signals outside of their modeled ranges and excessively violating the assumptions behind Equations (A.8) and (A.9).

Figure 4-12 shows samples of the climb maneuver $(-x, -z)$ profile when the bounding examples have the nonstandard, or looping nature. Note that although one might not be interested in using such a climbing strategy in practice, the emphasis here is that the interpolation algorithms create maneuver families in style of the known feasible solutions. Recall, that the ability to easily define a maneuver family "style" by simply providing a small number of example motions is one of the primary motivations for trajectory interpolation; see Section 1.1.2.

Unlike for the standard climb maneuver, the vehicle pitch angle for the looping climbs undergoes large excursions, as seen in Figure 4-13. From the figure, it is again apparent that the entire maneuver family retains the physical characteristics of the feasible examples and that the interpolation process is well-suited for motions outside the range of linearized vehicle dynamics.

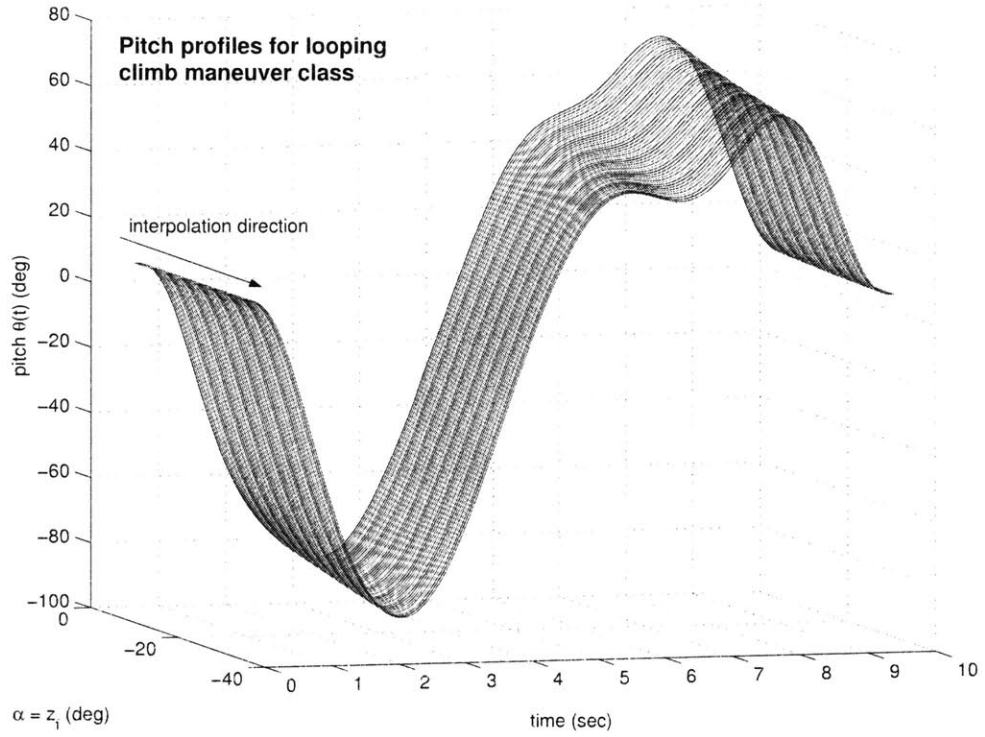


Figure 4-13: Pitch profiles for looping climb maneuver class.

4.5.3 Maneuver "Interpolability"

It is important to note that not all feasible maneuvers can be interpolated with the methods of Algorithms 1 through 3. For example, taking one of the above standard climb maneuvers to final elevation $\alpha_1 = \overline{z}_f = -10$ deg and one of the loop-style climbs to final elevation $\alpha_2 = \overline{z}_f = -35$ deg, attempts to create parametrized climb families generally fail. These motions are clearly not of the same qualitative *style* (even given the inexact use of that term), since the velocity and pitch profiles differ widely between the two examples. The interpolation process generally fails to drive the maneuver chart variable $\alpha = \overline{z}_f$ in the correct direction, since velocity and pitch coefficient differences, rather than elevation coefficient differences, dominate the projection operation, resulting in mathematically unreasonable control profiles (swinging erratically through double-digit magnitude changes even over small variations in $\alpha = \overline{z}_f$).

Put another way, these two maneuvers, while achieving similar *net* vehicle state changes, use very different strategies to accomplish the change. As such, when the interpolation algorithm uses the difference projection to compute a first-order feasible direction, the difference vector primarily reflects the velocity and pitch profile disparities, not the elevation variations. In this case, the projected difference has a small elevation component compared to the other variables, leading to a poorly conditioned projection onto the tangent plane and a subsequently infeasible maneuver class, with nonsensical control profiles.

In other situations, the user-chosen example maneuvers may be of such different types that there does not exist an intermediate connected space in which to com-

pute a feasible interpolation path. In still other cases, the maneuvers may be of the same general type, but the solution surface geometry leads into the constraint entrapments, typically because of solution path "dead-ends" created by numerous or highly nonlinear inequality constraints. In both of these cases, the interpolation algorithm stalls, resulting in no feasible projection or a "null" projection where $u^T \hat{u} \approx 0$ in Algorithms 1 through 3. Loosely speaking, such situations arise because of an excessively high manifold curvature or a topological inconsistency between the example motions in the full-dimension trajectory vector v-space, making it difficult to trace a feasible arc with the simple homotopy-based projection methods [62, 110]. Possible solutions in such situations include specialized algorithm modifications, such as using a matrix-weighted projection operation to obtain a modified $\hat{u}$ vector, working exclusively in subspaces of the v that avoid topological entrapments, or by changing the algorithm more fundamentally, using pathfollowing and homotopy algorithms based explicitly on manifold curvature tensors [109] that follow even feasible arcs exhibiting the $u^T \hat{u} \approx 0$ "orthogonality" condition.

For now, the main engineering guideline for creating maneuver classes is to choose example motions that use the same fundamental control strategy and trajectory characteristics. A general guide for practical use of the interpolation algorithms is to monitor the behavior of the projection inner product $u^T \hat{u}$ along the feasible solution arc. If this quantity approaches zero, it is possible that the as-given algorithms will create unsatisfactory maneuver families. Off-line solution for another "intermediate" feasible maneuver can help fix this situation, providing another trajectory guidepoint and allowing the interpolation algorithm to trace shorter arcs through the possibly highly nonlinear feasible interior space.

4.5.4 Pilot-Demonstrated Reposition Maneuver

The next two helicopter maneuver examples illustrate the process of using pilot motion capture data to create a pair of known, feasible trajectories. Through the interpolation process, these maneuvers then define a parametrized maneuver class, members of which can be flown under closed-loop control architecture of Section 4.4, demonstrating the entire maneuver design process on real hardware.

In this section, consider a pilot-demonstrated reposition maneuver, where a human operator flies the three degree-of-freedom helicopter via a joystick input [122], demonstrating two hover-to-hover position change motions. As seen in the overhead pictorial of Figure 4-14, the first position change is "short", with a displacement of $x_f - x_i = -47$ degrees; the second position change is "long" with an angular offset of -367 degrees. (Note: these particular numbers correspond to a specific sample maneuver pair from a batch of "satisfactory" demonstrations, with fairly smooth motions and a clean deceleration to the final hover state. The goal was to collect maneuvers with offsets of around -45 and -360 degrees). As noted in the figure, once these two maneuvers are available as prototypes, it is then possible to perform trajectory interpolation and fly any reposition maneuver with offset *between* -47 and -367 degrees.

For a hover-to-hover reposition maneuver, it is necessary to add an extra row to

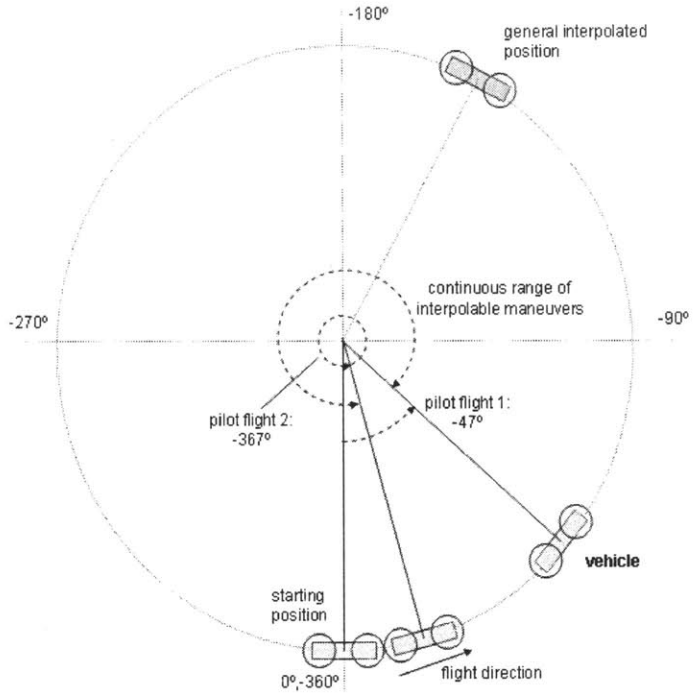


Figure 4-14: Overhead view of reposition maneuver interpolation.

the standard boundary condition equality constraint of Equation (4.29). The additional constraint integrates and scales the p-parametrized unit time domain velocity to obtain an α-parametrized final position constraint with $\alpha \equiv x_f = x(T)$ (it is understood that $x_i = x(0) \equiv 0$):

$$h_{bc}(p,\alpha) = \begin{bmatrix} v(0;p) - 0 \\ v(1;p) - 0 \\ z(0;p) - 0 \\ z(1;p) - 0 \\ \theta(0;p) - \overline{\theta}_{hov} \\ \theta(1;p) - \overline{\theta}_{hov} \\ V_{coll}(0;p) - \overline{V}_{coll,hov} \\ V_{coll}(1;p) - \overline{V}_{coll,hov} \\ V_{cyc}(0;p) - \overline{V}_{cyc,hov} \\ V_{cyc}(1;p) - \overline{V}_{cyc,hov} \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \\ T\int_0^1 v(\tau;p)d\tau - \alpha \end{bmatrix} = 0. \tag{4.57}$$

This last row has the same general form as isoperimetric constraints often seen in variational optimization problems [115].

Unlike the previous examples, the initial and final hover trim states do not require

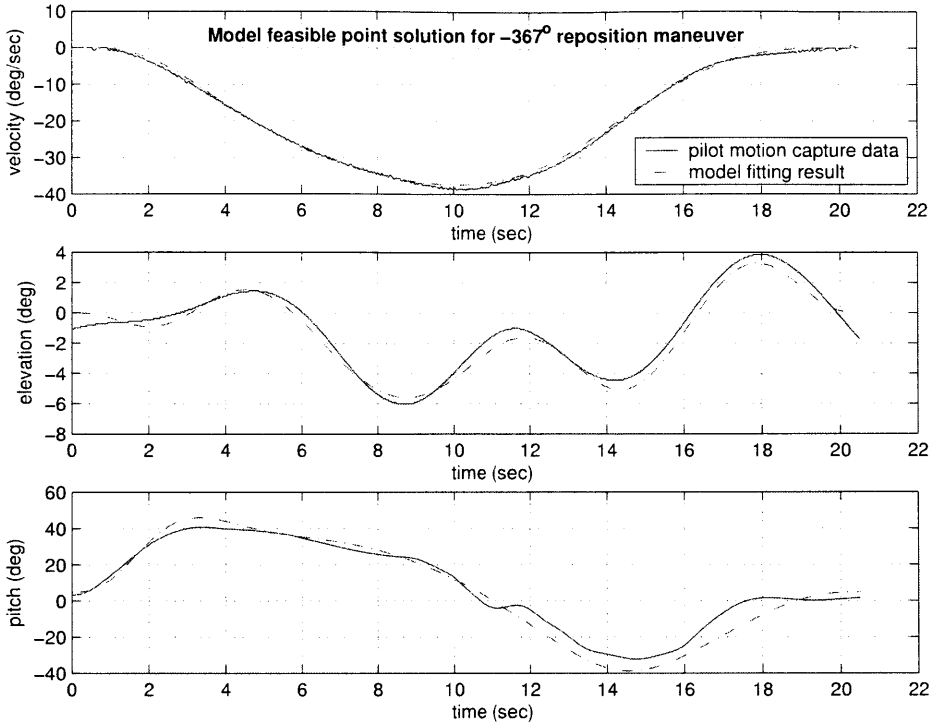


Figure 4-15: Motion capture result for -367^o reposition maneuver.

any α functions and simply use the static trim settings $[\overline{\theta}_{hov}, \overline{V}_{coll,hov}, \overline{V}_{cyc,hov}]^T = \eta(0,0)$. Analytic derivatives with respect to p of the integral position constraint follow from integration of each of the B-spline basis element in the first line of Equation (4.4), with numerical tools found in the spline literature [34] or in software toolboxes [35].

Motion capture of pilot-flown helicopter trajectories follows by recording the system behavior (here, 100 Hz sampled data of velocity, elevation, and pitch) and applying the methods of Section 4.3.2. Figure 4-15 shows the result when finding a model feasible point p_2 corresponding to the "long" $\alpha_2 = -367$ deg motion. The model feasible system behavior appears as the smoother dash-dot signal and matches closely the recorded (solid line) data. Note that as part of the feasibility Program (4.41), vehicle trim conditions impose exact initial and final hover state, so that "imperfect" or slowly drifting hover states observed in flight data get mathematically corrected to zero velocity and zero elevation.

Figure 4-16 shows the corresponding voltage input signals for the piloted run and the resulting feasible point. Note that as discussed in Section 4.3.2, the motion capture error metric $e(p)$ is a function of the system behavior $\mathbf{z} = [v, z, \theta]^T$ only and does not attempt to match the input signals. Therefore Figure 4-16 gives an interesting viewpoint on the model fidelity as well as the specific method the pilot uses to fly the particular motion. Interestingly, the pilot input shows no use of collective during the maneuver, using only the cyclic input channel to accelerate and decelerate the helicopter. In general, open-loop use of collective can easily excite the lightly-damped elevation mode, making it difficult to return the helicopter to a hover trim state without significant overshoot in velocity and elevation. The feasible model inputs (dash-dot curves in Figure 4-16) very closely match at lower frequency the pilot

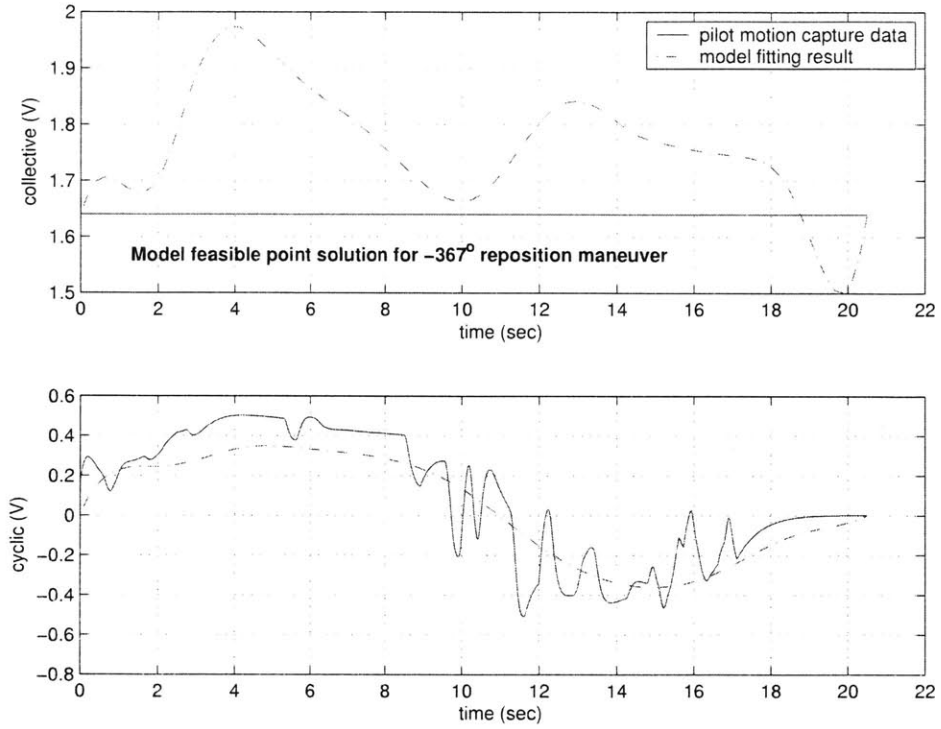


Figure 4-16: Input behavior for -367^o reposition maneuver.

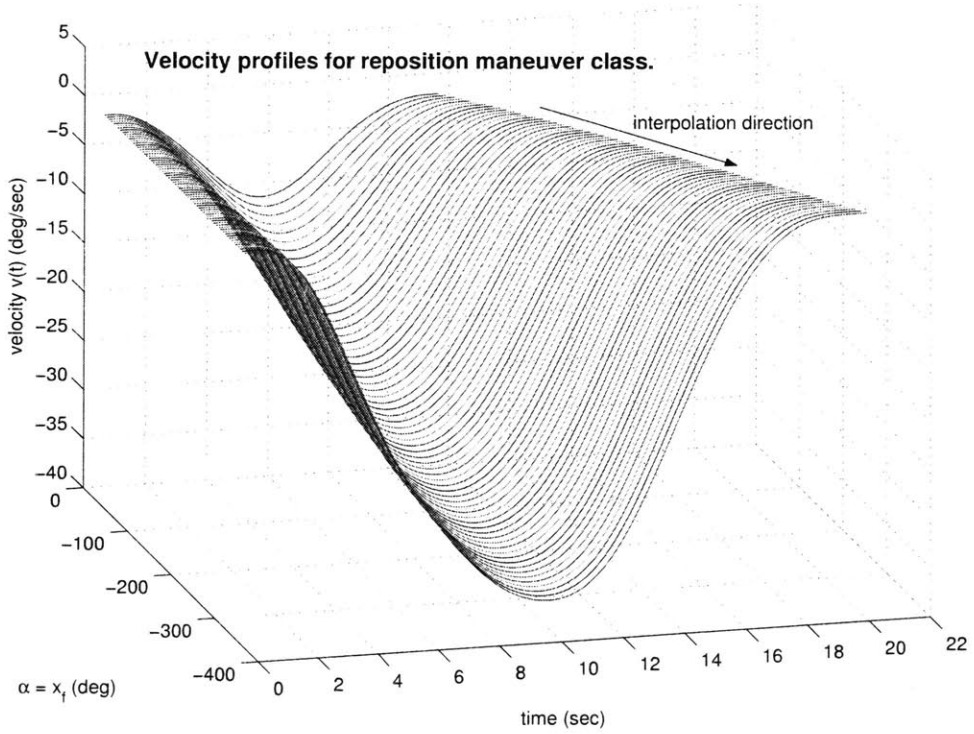


Figure 4-17: Velocity profiles for reposition maneuver class.

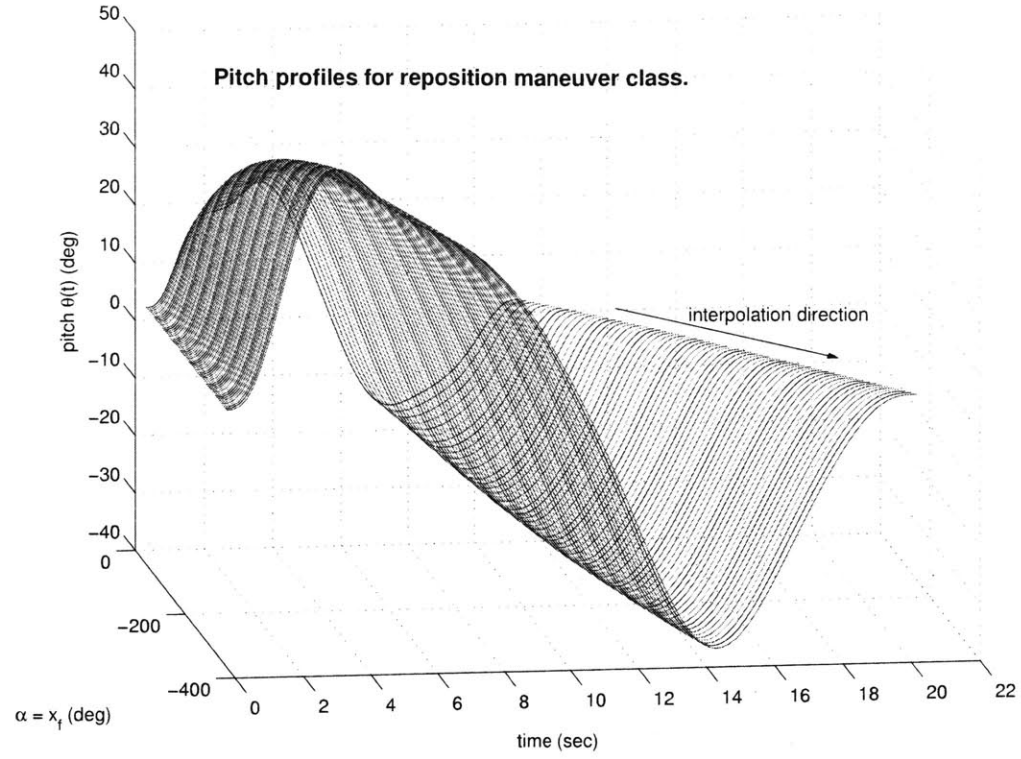


Figure 4-18: Pitch profiles for reposition maneuver class.

cyclic input while using some collective input to increase thrust and accelerate the helicopter. Motion capture plots for the p_1 feasible point corresponding to $\alpha_1 = -47$ deg closely resembles the plots shown here.

Once the feasible points p_1 and p_2 are available, trajectory interpolation fills out the entire reposition maneuver family. Figures 4-17 and 4-18 show the corresponding α-parametrized velocity and pitch profiles, respectively. It is then possible to select and fly, in closed-loop, the intermediate maneuvers corresponding to the values $\alpha = -180$ deg and $\alpha = -270$ deg. Taking sensor data snapshots of the corresponding motions on the actual helicopter gives the visualizations seen in the second and third plots of Figure 4-19. The first and fourth plots show snapshots of the pilot-flown α_1 and α_2 trajectories. The interpolated closed-loop trajectories very clearly resemble their generating examples, giving an experimental validation of the methods of Section 2.3.

4.5.5 Pilot-Demonstrated Dash-Accelerations

This subsection gives another example of piloted-flight motion capture, creating a parametrized family of hover-to-cruise dash-acceleration maneuvers. Here, the helicopter begins at a level hover state with $\overline{v}_i = 0$ and $\overline{z}_i = 0$. The final elevation is also fixed at $\overline{z}_f = 0$ but the final steady cruise speed is variable with $\alpha \equiv \overline{v}_f \in [-70, -22.5]$ deg/sec. These bounds follow from the particular recorded pilot maneuvers deemed to have satisfactory attributes, such as steady accelerations, little-to-no overshoot of the final velocity, and small elevation excursions from level.

Following the earlier maneuver examples of this section, the parametrized bound-

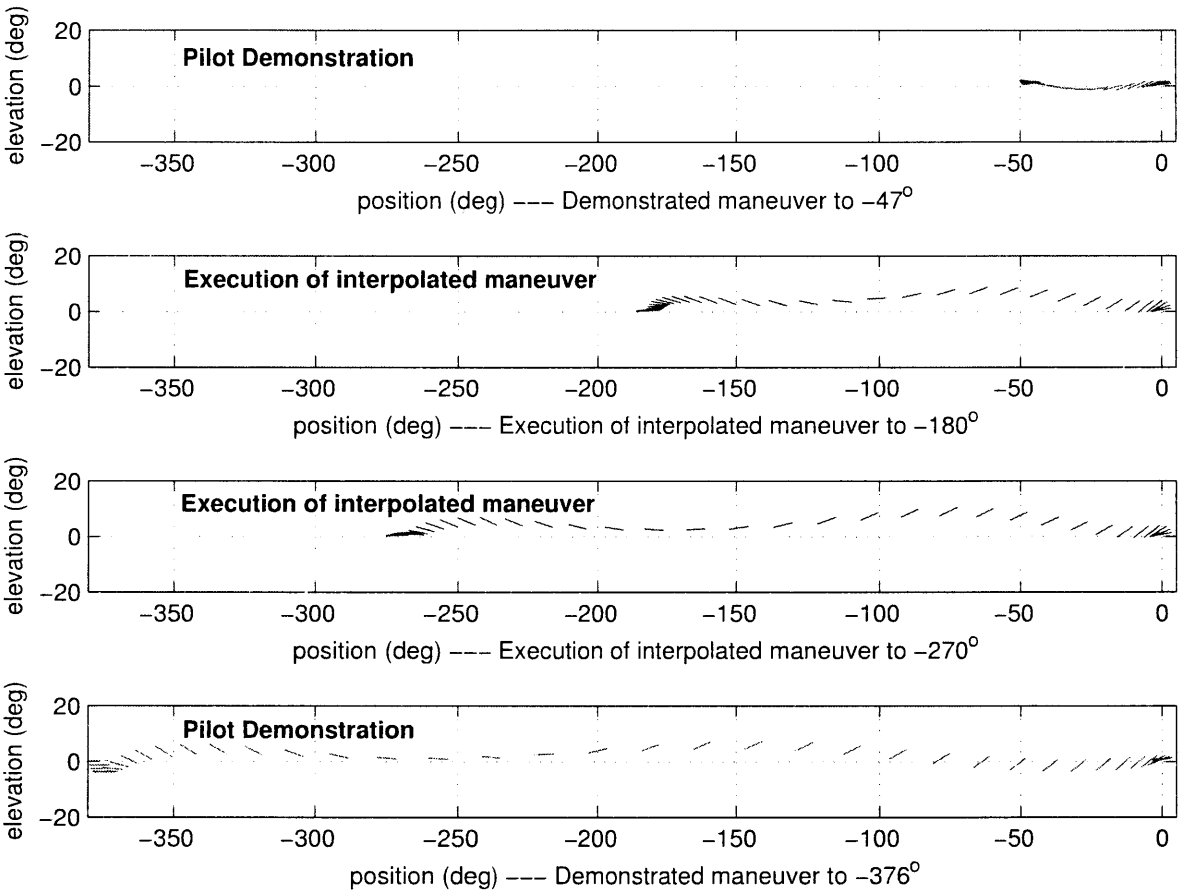


Figure 4-19: Visualization of demonstrated and interpolated reposition maneuver data.

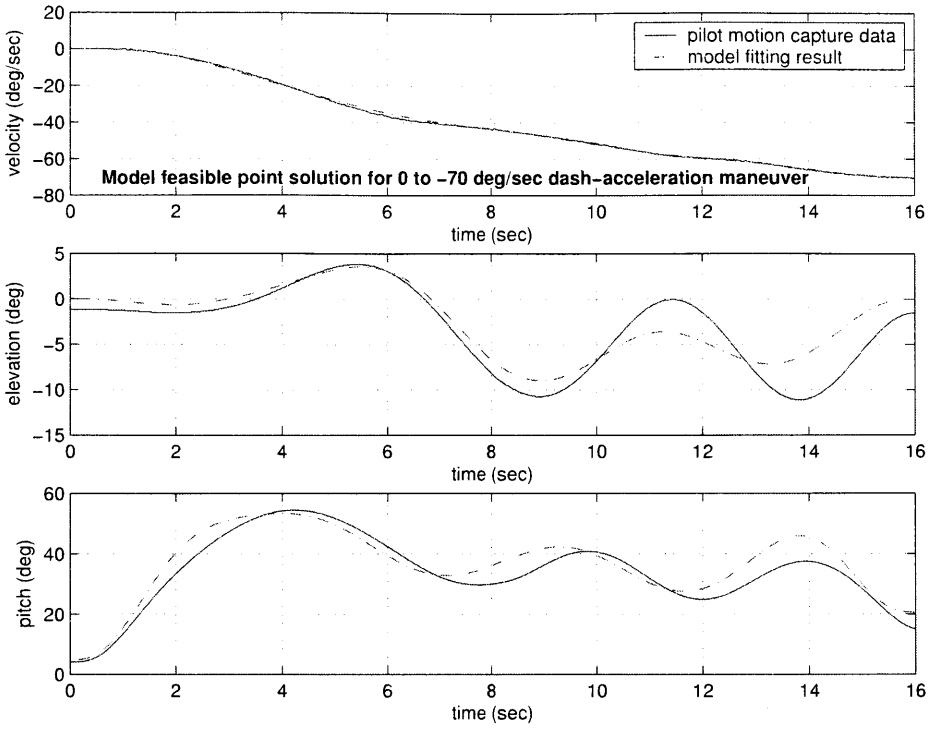


Figure 4-20: Motion capture result for dash-acceleration to -70 deg/sec maneuver.

ary condition constraint takes the form

$$h_{bc}(p,\alpha) = \begin{bmatrix} v(0;p) - 0 \\ v(1;p) - \alpha \\ z(0;p) - 0 \\ z(1;p) - 0 \\ \theta(0;p) - \overline{\theta}_{hov} \\ \theta(1;p) - \overline{\theta}_{f}(\alpha) \\ V_{coll}(0;p) - \overline{V}_{coll,hov} \\ V_{coll}(1;p) - \overline{V}_{coll,f}(\alpha) \\ V_{cyc}(0;p) - \overline{V}_{cyc,hov} \\ V_{cyc}(1;p) - \overline{V}_{cyc,f}(\alpha) \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \end{bmatrix} = 0 \tag{4.58}$$

where there is no need for the final position constraint of the preceding reposition maneuver.

Figure 4-20 shows the system behaviors of the second piloted motion prototype, corresponding to $\alpha_2 = -70$ deg/sec, and the resulting model feasible point, arrived at through the methods of Section 4.3.2. Figures 4-21 and 4-22 show the velocity and pitch profiles of the resulting dash-acceleration family, respectively. It is interesting to note that as the maneuver duration T increases with decreasing α (that is, with

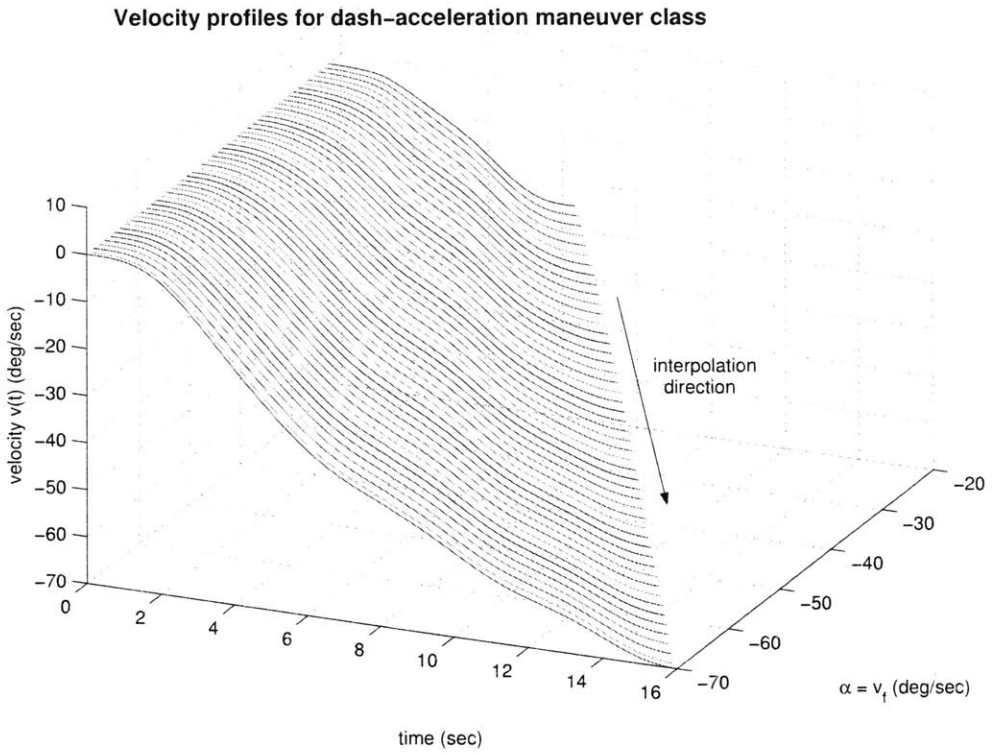


Figure 4-21: Velocity profiles for dash-acceleration maneuver class.

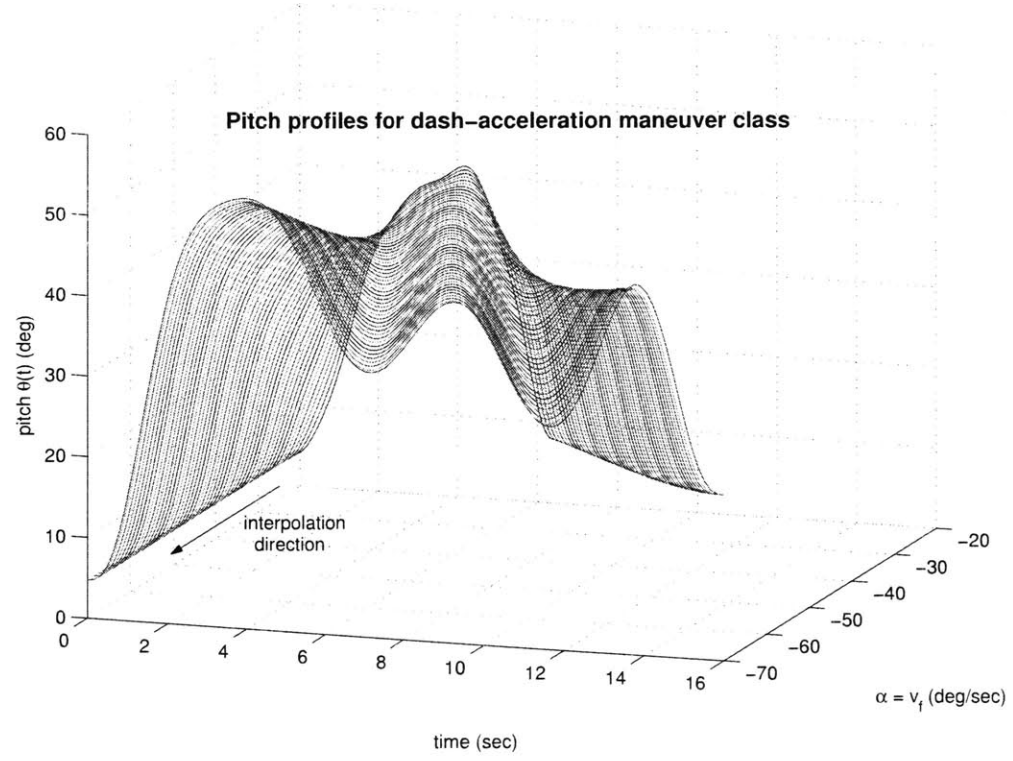


Figure 4-22: Pitch profiles for dash-acceleration maneuver class.

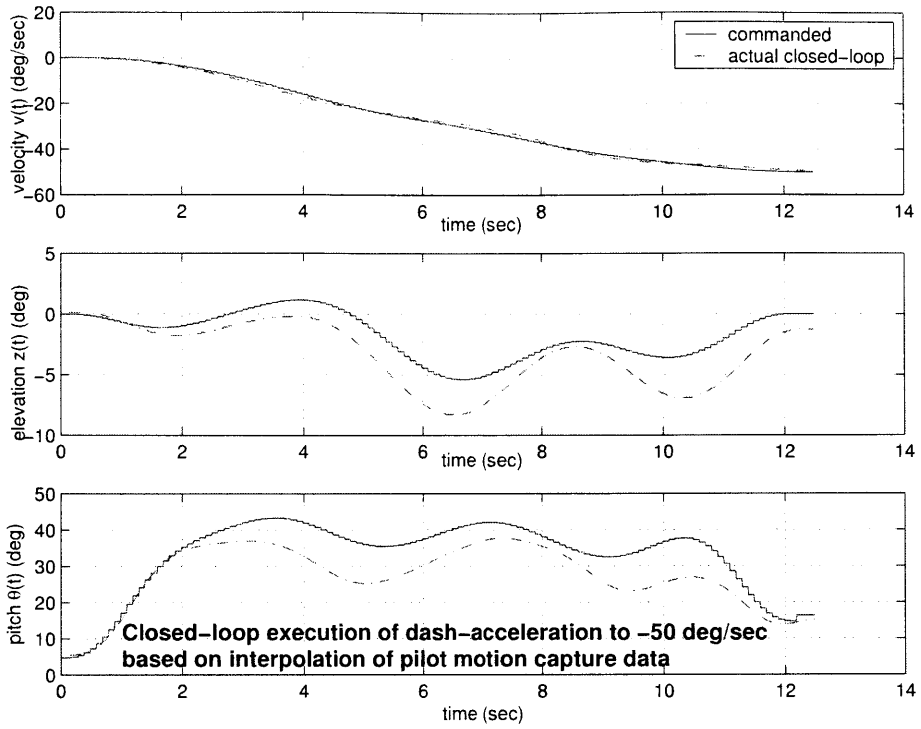


Figure 4-23: Closed-loop execution of pilot-inspired dash from hover to -50 deg/sec.

larger final velocity *magnitude*) the pitch signals display more oscillation periods. This behavior makes intuitive sense, as the linear hover pitch period is roughly 5 seconds (see Appendix A) so that longer maneuvers allow more time for this mode to develop. Similar behaviors hold for the parametrized elevation profiles (not shown here), which through nonlinear coupling to the pitch dynamics (see Equations 4.1), tend to oscillate with a roughly 5 second period. This behavior is evident in the single maneuver motion sample of Figure 4-20.

Just as with the parametrized reposition family, it is straightforward to take a particular maneuver instance from the continuous family $p(\alpha)$ and execute it in closed-loop on the helicopter. Such "playback" demonstrates the end result of the trajectory interpolation process. It also helps illustrate a potential training tool for machine learning algorithms that use human-demonstrated examples to provide general motion primitives for autonomous systems. Figure 4-23 shows the reference and closed-loop tracking behavior for a dash-acceleration with $\alpha = -50$ deg/sec. As discussed in Section 4.4, state and control weights for the gain-scheduled LQ-servo design emphasize velocity tracking, leading to fairly small errors in the first subplot of the figure. However, the (lower weight) feedback on elevation and pitch tracking still maintains reliable performance for z and θ, with typically no more than 3 degrees of error in elevation and 10 degrees of error in pitch for this maneuver class.

Chapter 5

Maneuver Classes in Hybrid Motion Planning

With the parametrized maneuver framework established, it is now possible to use trajectory families for hierarchical vehicle path planning. High-level hybrid system motion planners often use abstracted dynamic models to generate vehicle guidance solutions. In the present context, the dynamics-based algorithms of Chapter 2 allow high-level planners to consider maneuver classes as both simple motion building blocks and inner-loop reference trajectory generators for low-level tracking controllers.

This chapter illustrates the use of parametrized maneuvers as motion primitives in an existing mixed integer-linear programming (MILP) optimization framework. As noted in Section 1.3.1, MILP-based methods provide powerful tools for formulating diverse vehicle guidance problems. These techniques accommodate simplified high-level representations of inner-loop control systems, offer a natural interface for including agile maneuver elements, and have been successfully demonstrated in numerous practical vehicle guidance settings [8, 95, 131, 132, 133, 134]. The MILP problem formulation uses continuous-valued state variables to describe a system's configuration space and discrete, binary variables to capture mode-switching operations as well as logic-based and combinatorial constraint sets. However, to date, most MILP planners have only employed static maneuvering elements with fixed, invariant maneuver boundary conditions [133]. With the parametrized trajectory capability, it is now possible to generalize the maneuvering operation, thus adding richness to the existing guidance framework and creating an extremely flexible path planning capability.

This chapter begins with a brief introduction to the MILP guidance framework, giving an overview of objective function and constraint types useful for planning motions of the three degree-of-freedom helicopter. This overview follows from examples provided by numerous MILP-based rotorcraft and fixed-wing references [8, 40, 95, 125, 132, 133, 134], and is included here to provide sufficient background and to demonstrate the versatility of the MILP method.

The chapter then illustrates how to both simplify the system model of Equations (4.1) for planning purposes and adapt the parametrized maneuvering constraint formulations of Chapters 2, 3, and 4 to the MILP format requirements. Finally, the

last two sections use the combined guidance-maneuvering framework to solve interesting one and two-dimensional planning problems. The one-dimensional case shows the benefits of parametrized maneuver boundary conditions and details the specification of nonlinear system motions that meet MILP constraint requirements. The two-dimensional case illustrates the ease with which interesting maneuver classes can be designed for particular mission requirements. Both of these cases solve MILP problems using the CPLEX® optimization software [69] and an AMPL® encoding language preprocessing environment [50].

5.1 Mixed Integer-Linear Programming for Trajectory Planning

The core of the mixed integer-linear programming framework is a set of discrete-time linearly controlled system models, representing a vehicle in closed-loop operation. The linear model states need only include those quantities, such as vehicle position and velocity, necessary for solving a guidance problem. As a means of reducing problem dimensionality without sacrificing high-level model fidelity, many inner-loop quantities, such as pitch angle for the 3-DOF helicopter or pitch and rotor states for airborne helicopters, can be excluded from the MILP formulation.

Accompanying the linear system closed-loop models are a set of binary variables, taking values of either 0 or 1 at any given decision step. These variables allow for switching between different closed-loop models (as might be appropriate in different regions of a vehicle flight envelope), decisions to execute agile maneuvers, and complex constraint sets possibly involving conditional logic. These binary variables can appear in both the objective and constraint functions, often activating or deactivating expressions that correspond to the different system models, maneuver state transformations, and physical guidance constraints.

This section presents several useful MILP objective function and constraint tools for constructing interesting vehicle guidance problems, based on examples in the literature [8, 40, 95, 125, 132, 133, 134]. The discussion begins with a review of the basic problem variables, including the planning decision horizon, state and binary variable types, and linear command-following dynamic system models. Then the novel parametrized *maneuver class* representations, generalizing existing fixed maneuver elements, appear as additional planning alternatives to complement the linearly controlled modes. With these problem building blocks in place, the section concludes by reviewing existing minimum time and minimum state error objective functions and other useful constraint types.

5.1.1 Basic Definitions

The MILP hybrid system model operates on a discrete *decision-step* planning horizon, with each individual decision opportunity given an integer index t. In general, this index variable does not represent the system at some *time* t but a decision made at *step* t (one may use the symbol k in place of t, if preferred). When operating

in a linearly-controlled mode (a so-called "linear-time invariant mode" (LTI-mode)), the decision step intervals coincide with the sampling rate T_s of the linear models. However, when executing a maneuver trajectory, the decision steps correspond to the actual duration of the maneuver, since it is only at maneuver completion that the next guidance decision can be made. This distinction between time step and decision step will be important particularly when solving minimum time guidance problems.

Consider a discrete planning horizon over H decision steps, each denoted by indices t with

$$t \in \{0, 1, 2, \ldots, H\}. \tag{5.1}$$

Define a closed-loop continuous-valued system state vector $\mathbf{x}$ with N_x components. This variable gives the high-level vehicle state at any given decisions step t according to

$$\mathbf{x}[t] = \begin{bmatrix} x_1[t] \\ \vdots \\ x_{N_x}[t] \end{bmatrix}. \tag{5.2}$$

The components of this vector typically correspond to closed-loop vehicle states, including accelerations, velocities, and positions, but do not include detailed "internal" states unnecessary for the guidance problem. (Note that the bold vector quantities of this chapter do not correspond to the boldface vectors in Section 4.4). Similarly, define a closed-loop continuous-valued system command input vector $\mathbf{u}$ with N_u components, given at each decision step t according to

$$\mathbf{u}[t] = \begin{bmatrix} u_1[t] \\ \vdots \\ u_{N_u}[t] \end{bmatrix}. \tag{5.3}$$

The components of $\mathbf{u}$ typically correspond to commands issued to the closed-loop system, such as velocity or position setpoints.

Define the following four index sets to denote the entire integer decision horizon, a decision horizon excluding the final decision step H, the set of state vector component indices, and the set of input vector component indices, respectively:

$$\begin{aligned} \mathbf{H} &= \{0, \ldots, H\} \\ \mathbf{H}_{-1} &= \{0, \ldots, H-1\} \\ \mathbf{X} &= \{1, \ldots, N_x\} \\ \mathbf{U} &= \{1, \ldots, N_u\} \end{aligned} \tag{5.4}$$

These notations will be useful later for describing the discrete domains of the problem binary variables.

Further, given an ordered subset of $n \leq N_x$ state indices $S = \{s_1, \ldots, s_n\} \subset \mathbf{X}$,

define the corresponding state component vector $\mathbf{x}_S[t]$ at step t as

$$\mathbf{x}_S[t] \equiv \begin{bmatrix} x_{s_1}[t] \\ \vdots \\ x_{s_n}[t] \end{bmatrix}. \tag{5.5}$$

Similarly, for an ordered subset of $m \leq N_u$ input indices $P = \{p_1, \ldots . p_m\} \subset \mathbf{U}$, define the corresponding state function $\mathbf{u}_P[t]$ at step t as

$$\mathbf{u}_P[t] \equiv \begin{bmatrix} u_{p_1}[t] \\ \vdots \\ u_{p_m}[t] \end{bmatrix}. \tag{5.6}$$

5.1.2 Linear Operating Modes and Binary Decision Variables

In a general MILP planning problem, the system may switch between L linearly-controlled models (LTI-modes) [133, 134]. These models may correspond to different control architectures or to different controller gain selections over various system operating regions. As such, define an LTI-mode index set $\mathbf{L}$ as

$$\mathbf{L} = \{1, \ldots, L\}. \tag{5.7}$$

For each LTI-mode and each decision step up to but excluding step H, define a binary variable b_i a value of either 0 or 1:

$$b_i[t] \in \{0, 1\} \quad \forall\ t \in \mathbf{H}_{-1}, \forall\ i \in \mathbf{L}. \tag{5.8}$$

Here, $b_i[t] = 1$ if the system is in LTI-mode i at step t. Alternatively, $b_i[t] = 0$ if the system is *not* in LTI model i at step t. For each LTI-mode and each decision step, there exists a set of linear constraints with state matrix A_i and input matrix B_i describing the *closed-loop* system state evolution. That is, $\forall\ t \in \mathbf{H}_{-1}, \forall\ i \in \mathbf{L}$ it holds that

$$\begin{aligned} \mathbf{x}[t+1] - A_i\mathbf{x}[t] - B_i\mathbf{u}[t] &\leq K(1 - b_i[t]) \\ -\mathbf{x}[t+1] + A_i\mathbf{x}[t] + B_i\mathbf{u}[t] &\leq K(1 - b_i[t]), \end{aligned} \tag{5.9}$$

where K is an appropriately sized vector of large positive numbers, many times greater than the typical problem data. The role of the binaries is made clear by Inequalities (5.9). If $b_i[t] = 1$ at step t, then the right hand sides both equal zero and the system must obey the state transition of the ith LTI-mode, that is, $\mathbf{x}[t+1] = A_i\mathbf{x}[t] + B_i\mathbf{u}[t]$. Alternatively, if $b_i[t] = 0$ at step t, the right hand sides are equal to vectors of large numbers, meaning that the constraints of Inequalities (5.9) are relaxed.

5.1.3 Parametrized Maneuver Class Representation

As a planning alternative to the LTI-modes, a vehicle may execute a member of an agile maneuver family at decision step t. Consider a set of M parametrized maneuver classes, each given an index j and define the corresponding maneuver index set $\mathbf{M}$ as

$$\mathbf{M} = \{1, \ldots, M\}. \tag{5.10}$$

For each maneuver class and decision step, define a corresponding maneuver binary variable $m_j[t]$ according to

$$m_j[t] \in \{0, 1\} \quad \forall\ t \in \mathbf{H}_{-1}, \forall\ j \in \mathbf{M}. \tag{5.11}$$

These binaries play essentially identical roles to the LTI binaries of Equation (5.8), with $m_j[t] = 1$ if a maneuver from class j is executed at step t and $m_j[t] = 0$ if a maneuver from class j is *not* executed at step t. Similar to the LTI-mode control-driven state transitions of Inequalities (5.9), a set of inequalities describe the state transitions of the *entire* jth maneuver class. That is, employ a set of maneuver constraints $\forall\ t \in \mathbf{H}_{-1}, \forall\ j \in \mathbf{M}$ such that

$$\begin{aligned} \mathbf{x}[t+1] - M_{s,j}\mathbf{x}[t] - M_{c,j} - \mathbf{x}[t] &\leq K(1 - m_j[t]) \\ -\mathbf{x}[t+1] + M_{s,j}\mathbf{x}[t] + M_{c,j} + \mathbf{x}[t] &\leq K(1 - m_j[t]), \end{aligned} \tag{5.12}$$

with the constraint interpretation and role of K identical to that in Inequalities (5.9). Note that these constraints imply that the jth maneuver class obey a set of *affine* state transitions (with transition matrix $M_{s,j}$ and constant vector $M_{c,j}$) based on the state $\mathbf{x}[t]$ at decision step t. Later sections of this chapter will describe how this affine transformation requirement can be stipulated for parametrized maneuvers of nonlinear systems. It is important to note that this affine representation of maneuver classes mimics existing methods for including single, fixed maneuver elements [133, 134], and therefore provides a means of adding richness to set of available guidance solutions without significantly altering the basic problem formulation.

5.1.4 Mode Switching

Further, it is clear that a vehicle cannot operate in more than one LTI-mode or maneuvering motion at a given decision step. Therefore, it is necessary to impose a mutual exclusivity summation constraint on the LTI-mode and maneuver binaries at each decision step, so that, $\forall\ t \in \mathbf{H}_{-1}$

$$\sum_{i=1}^{L} b_i[t] + \sum_{j=1}^{M} m_j[t] = 1. \tag{5.13}$$

In a given problem, if there is only one LTI-mode, Equation (5.13) can be dropped for simplicity, with a maneuver binary summation of the form $K\sum_{j=1}^{M} m_j[t]$ replacing the right-hand side of Inequalities (5.9).

5.1.5 Minimum Time Formulation

Given a vehicle initial condition and a desired goal state, two useful objective functions for MILP guidance problems are those minimizing either vehicle arrival time or an accumulated state error function [132, 133, 134]. To compute minimum time solutions, first introduce an additional binary variable $d[t]$ for every decision step, according to

$$d[t] \in \{0, 1\} \quad \forall \ t \in \mathbf{H}. \tag{5.14}$$

Let $\mathbf{x}_{goal}$ denote the desired system terminal goal state, with a subset $S \subset \mathbf{X}$ used to select the specific components of $\mathbf{x}$ contained in $\mathbf{x}_{goal}$. (For example, the goal condition may involve only a position, even though the full planning state $\mathbf{x}$ contains both position and velocity). Now add the following terminal state inequality constraint functions $\forall \ t \in \mathbf{H}$:

$$\begin{aligned} \mathbf{x}_S[t] - \mathbf{x}_{goal} &\leq K(1 - d[t]) \\ -\mathbf{x}_S[t] + \mathbf{x}_{goal} &\leq K(1 - d[t]). \end{aligned} \tag{5.15}$$

In coordination with the binary variable definition of Equation (5.14), when $d[t] = 1$, the system must be in the goal configuration at decision step t, with $\mathbf{x}_S[t] = \mathbf{x}_{goal}$. For any step where the system state is not consistent with the goal, it must hold that $d[t] = 0$. To require that the goal state be reached at a unique step in the planning horizon $\mathbf{H}$, introduce the binary equality constraint

$$\sum_{t=0}^{H} d[t] = 1. \tag{5.16}$$

This uniqueness specification is necessary to pose the minimum time objective function J in the form

$$J = \sum_{t=0}^{H} d[t] t T_s + \sum_{t=0}^{H-1} (c_m[t] - \sum_{j=1}^{M} m_j[t] T_s). \tag{5.17}$$

The first term sums the elapsed time to reach the goal state $\mathbf{x}_{goal}$ assuming the guidance solution is entirely composed of LTI-modes with fixed sampling interval T_s. The second terms corrects the summation using the M maneuver binaries $m_j[t]$ and a special $c_m[t]$ cost term in case any maneuvers (whose durations are not equal to T_s, in general) are employed during the problem solution. This novel maneuver duration $c_m[t]$ term equals the *actual* time duration of a maneuver executed at decision step t and is set according to the following constraints, specified $\forall \ t \in \mathbf{H}_{-1}, \forall \ j \in \mathbf{M}$:

$$\begin{aligned} c_m[t] - J_{s,j}\mathbf{x}[t] - J_{c,j} &\leq K(1 - m_j[t]) \\ -c_m[t] + J_{s,j}\mathbf{x}[t] + J_{c,j} &\leq K(1 - m_j[t]), \end{aligned} \tag{5.18}$$

and

$$
\begin{aligned}
c_m[t] &\leq K \sum_{j=1}^{M} m_j[t] \\
-c_m[t] &\leq K \sum_{j=1}^{M} m_j[t].
\end{aligned}
\tag{5.19}
$$

Inequalities (5.18) update the maneuvering cost using a $1 \times N_x$ vector $J_{s,j}$ and scalar $J_{c,j}$ for each maneuver class j according to $c_m[t] = J_{s,j}\mathbf{x}[t] + J_{c,j}$ (if $m_j[t] = 1$). The $J_{s,j}$ and $J_{c,j}$ indicate that the maneuver duration is an *affine* function of the system state at the commencement of the maneuver. Similar to the affine state transformations of Inequalities (5.12), the linear duration requirement can be imposed during the design of maneuver families, as will be illustrated later in this chapter. Note that if a maneuver is *not* executed at decision step t, then Inequalities (5.19) set $c_m[t] = 0$, and the cost function J then instead counts the LTI-mode sampling interval T_s at time t. Given the objective function of Equation (5.17), the binary variables of Equations (5.8), (5.11), and (5.14) and the continuous-valued, decision-step state variables of Equation (5.2) and control variables of Equation (5.3) and finally the maneuvering cost $c_m[t]$, the minimum time guidance problem statement is [40, 133, 134]:

$$
\min_{b_i[t], m_j[t], d[t]; \mathbf{x}[t], \mathbf{u}[t], c_m[t]} J. \tag{5.20}
$$

The constraint set comes from the equalities and inequalities given so far and any additional constraint appearing in later subsections used to describe a particular guidance scenario.

Note that it is occasionally useful to add a term of the form $\sum_{t=0}^{H-1} \sum_{l=1}^{N_u} \epsilon_u |u_l[t]|$ to the cost function of Equation (5.17), where ϵ_u is a small number satisfying $0 < \epsilon_u \ll 1$ [133]. This expression helps prevent undesired erratic or oscillating control inputs when there might be a smoother, more intuitive control profile that drives the system to the same goal state with roughly the same cost performance. Note that this correction term contains a nonlinear absolute value expression that, at first glance, does not appear to satisfy the linearity requirement of a MILP problem statement. References [15, 17] discuss the introduction of linear slack variables to model the absolute value function.

5.1.6 Minimum State Error formulation

While the minimum time formulation is fairly intuitive and straightforward to implement, it requires the introduction of the $d[t]$ binaries at every decision step and the use of the linear maneuver cost function $c_m[t]$. These additional terms add complexity to the problem formulation and can increase solution time, an important consideration for prospective real-time applications. In addition, the requirement of linear time variation within a maneuver class requires an optimality gap for most useful motions, thus inducing an inherent performance degradation.

A useful alternative in some settings is an accumulated state error metric [134], similar to an integral 1-norm objective in continuous-time problems. Here, the MILP objective function takes the form

$$J = \sum_{t=0}^{H} ||W\mathbf{x}_{err}[t]||_1, \tag{5.21}$$

where $\mathbf{x}_{err}[t]$ gives the error between the system state $\mathbf{x}[t]$ at time t and a desired goal state $\mathbf{x}_{goal}$, according to the following constraints $\forall\ t \in \mathbf{H}$

$$\begin{aligned} \mathbf{x}_{err}[t] - (\mathbf{x}_E[t] - \mathbf{x}_{goal}) &\leq 0 \\ -\mathbf{x}_{err}[t] + (\mathbf{x}_E[t] - \mathbf{x}_{goal}) &\leq 0. \end{aligned} \tag{5.22}$$

Like the subset S in Inequalities (5.15), the subset $E \subset \mathbf{X}$ is useful for extracting those system state components required to define the guidance error. The W in Equation (5.21) denotes a diagonal weighting matrix used to normalize state physical dimensions and/or emphasize some error components over others. The vector 1-norm in Equation (5.21) naturally includes scalar absolute value functions which can again be accommodated using the slack variable methods of references [15, 17].

Given these definitions, the minimum error problem statement is:

$$\min_{b_i[t], m_j[t]; \mathbf{x}[t], \mathbf{u}[t]} J, \tag{5.23}$$

which does not contain the $d[t]$ and $c_m[t]$ free variables seen in Equation (5.20).

5.1.7 Other Existing Constraint Types

In addition to the above system representations and objective functions, there are many other constraint formulations useful for fully modeling a guidance optimization problem. This subsection reviews some of the standard constraint types in vehicle planning problems with previous examples available in the literature [8, 40, 95, 125, 131, 132, 133, 134].

Naturally, it is necessary to specify an initial system state $\mathbf{x}_0$ and control command vector $\mathbf{u}_0$. In addition, for minimum time problems, it is useful to set the maneuvering cost function $c_m[t]$ to zero at the problem outset. These three conditions are given by the simple assignments:

$$\mathbf{x}[0] = \mathbf{x}_0, \qquad \mathbf{u}[0] = \mathbf{u}_0, \qquad c_m[0] = 0. \tag{5.24}$$

To impose a global (i.e. at every decision step) upper bound $\overline{\mathbf{x}}$ on a index subset $S_{\overline{x}} \subset \mathbf{X}$ of the state vector and, similarly, a global lower bound $\underline{\mathbf{x}}$ on an index subset $S_{\underline{x}} \subset \mathbf{X}$, apply constraints, $\forall\ t \in \mathbf{H}$, of the form

$$\begin{aligned} \mathbf{x}_{S_{\overline{x}}}[t] - \overline{\mathbf{x}} &\leq 0 \\ -\mathbf{x}_{S_{\underline{x}}}[t] + \underline{\mathbf{x}} &\leq 0. \end{aligned} \tag{5.25}$$

Global control bounds, often necessary for well-posed optimization problems, follow identically to Inequalities (5.25) and, $\forall\ t \in \mathbf{H}$, take the form

$$\begin{aligned} \mathbf{u}_{S_{\overline{u}}}[t] - \overline{\mathbf{u}} &\leq 0 \\ -\mathbf{u}_{S_{\underline{u}}}[t] + \underline{\mathbf{u}} &\leq 0, \end{aligned} \tag{5.26}$$

with $\overline{\mathbf{u}}$ giving the upper bound on an index subset $S_{\overline{u}} \subset \mathbf{U}$ of the control and $\underline{\mathbf{u}}$ giving a lower bound on an index subset $S_{\underline{u}} \subset \mathbf{U}$.

An extremely useful set of bounds are restrictions on the vehicle initial conditions for the M maneuver classes. If a certain maneuver is only parametrized or feasible over some finite range of initial states (for example, over certain velocity or altitude range), it is necessary to impose these restrictions on the MILP-based guidance solution. Fortunately, such bounds are easy to specify and are based on a user-provided upper bound $\overline{\mathbf{x}}_{I_j,0}$ over an index subset $\overline{I}_j \subset \mathbf{X}$ of the state vector for the jth maneuver, with corresponding lower bounds and index subset given by $\underline{\mathbf{x}}_{I_j,0}$ and $\underline{I}_j \subset \mathbf{X}$, respectively. These "maneuver authorization" constraints then take the following form $\forall\ t \in \mathbf{H}_{-1}, \forall\ j \in \mathbf{M}$:

$$\begin{aligned} \mathbf{x}_{\overline{I}_j}[t] - \overline{\mathbf{x}}_{j,0} &\leq K(1 - m_j[t]) \\ -\mathbf{x}_{\underline{I}_j}[t] + \underline{\mathbf{x}}_{j,0} &\leq K(1 - m_j[t]). \end{aligned} \tag{5.27}$$

Recalling the interpretation of the maneuver binary variables from Equation (5.11), the right-hand sides of Inequalities (5.27) enforce the authorization bounds for a maneuver executed at step t. Note that in other contexts, constraints of this type can also be employed using LTI-mode binaries $b_i[t]$, forming a partition of the state space that triggers switchings between the various state-space models of Inequalities (5.9) [133, 134].

Bounds on the number of maneuvers executed during a particular guidance solution are easily imposed with linear summation constraints on the maneuver binaries themselves. For example, to place an upper bound N_m on the total number of maneuvers executed, specify a constraint of the following form $\forall\ t \in \mathbf{H}_{-1}$:

$$\sum_{t=0}^{H-1} \sum_{j=1}^{M} m_j[t] \leq N_m. \tag{5.28}$$

It is a simple matter to impose such bounds on individual maneuver classes by splitting this summation into multiple constraints.

Finally, and particularly useful for many interesting guidance problems, is the ability to place obstacles in a vehicle flight space, thus forcing a solution path to navigate around the obstacles while obeying the (dynamically feasible) closed-loop system dynamics. References [131, 133] describe a method for approximating general nonpolytopic constraints by a set of feasible half-planes.

The example in this thesis will treat only 4-sided rectangular obstacles (which require the planning state $\mathbf{x}$ to contain at least two physical position components). Consider the qth member of a set of Q rectangular obstacles. This obstruction will have

four faces, denoted here by the symbols f_1, f_2, f_3, f_4. For each face, define obstacle face binary variables at each decision step and denote them by $b^q_{f_1}[t], b^q_{f_2}[t], b^q_{f_3}[t], b^q_{f_4}[t]$, with $b^q_{f,i} \in \{0,1\}\ \forall\ t \in \mathbf{H}$. A given face binary is 0 if the constraint enforcing no penetration of that face is active; the binary is 1 if the no collision constraint is inactive. Now, in very general notation, express the total rectangular obstacle avoidance condition as the inequalities

$$\begin{aligned} \pm(M^q_1 \mathbf{x}_{S^q_{f_1}}[t] - \mathbf{p}^q_{f_1}) &\leq K b^q_{f_1}[t] \\ \pm(M^q_2 \mathbf{x}_{S^q_{f_2}}[t] - \mathbf{p}^q_{f_2}) &\leq K b^q_{f_2}[t] \\ \pm(M^q_3 \mathbf{x}_{S^q_{f_3}}[t] - \mathbf{p}^q_{f_3}) &\leq K b^q_{f_3}[t] \\ \pm(M^q_4 \mathbf{x}_{S^q_{f_4}}[t] - \mathbf{p}^q_{f_4}) &\leq K b^q_{f_4}[t], \end{aligned} \tag{5.29}$$

where the M^q_i and $\mathbf{p}^q_{f_i}$ are vectors and constants, respectively, to the describe the orientation and position of the ith face of the qth obstacle; the $S^q_{f_i} \subset \mathbf{X}$ are the index subsets necessary to compute vehicle position with respect to the faces; and the $\pm$ notation allows generality in choosing the face orientations (i.e. which sides of the faces are the obstacle interior or exterior). In actuality, it is impossible for a feasible vehicle path to satisfy all four of these constraints simultaneously. Instead, all that is required is to satisfy *at least one* of the constraints at every decision step [131, 133]. Therefore, the following obstacle face binary upper bound must accompany the Inequalities (5.29), so that, $t \in \mathbf{H}$:

$$\sum_{i=1}^{4} b^q_{f_i}[t] \leq 3. \tag{5.30}$$

Note that Inequalities (5.29) and (5.30) only enforce obstacle avoidance constraint satisfaction at times corresponding to decision steps $t \in \mathbf{H}$; the ultimate continuous-time guidance trajectory between decision step t and $t+1$ may actually clip constraint faces, especially near corners of polyhedral obstacles. This inter-step violation is a natural consequence of working with a discrete decision step horizon. In general, when solving guidance problem with a MILP framework, it is necessary to artificially expand the obstacle boundaries by some margin-of-safety depending on the sampled system dynamics, sampling time T_s, and obstacle physical dimensions [132]. In many cases, it is possible to use the state and control bounds to bound the inter-decision step constraint violation.

5.2 Application to the 3-DOF Helicopter

Given the basic mixed integer-linear programming tools of the preceding section, it is now possible to consider some of the specific terms for the three degree-of-freedom helicopter. In particular, this section describes the LTI-modes used to plan helicopter motions, the requirement of affine maneuver class state transformations and time durations as seen in Inequalities (5.12) and (5.18), and techniques useful for

generating parametrized maneuvers quickly. Together, these discussions will set up the one and two-dimensional motion planning examples of the following sections.

5.2.1 LTI Models

The most basic element of MILP-based motion planning is the LTI-mode, which approximates a vehicle operating in a closed-loop command-following configuration. For the helicopter, the same control architecture used for nonlinear maneuver tracking, as discussed in Section 4.4, may be used for nominal velocity, position, and elevation control. As will be seen shortly, the helicopter LTI-mode commands are sequences of velocity and elevation setpoints. To apply the methods of Section 4.4 for control of LTI-modes, simply take the MILP-solved state vector sequences, apply corresponding quasi-static collective and cyclic profiles based on Equation (4.22), and apply the feedback loop of Figure 4-2 with $\theta_{cmd} = \dot{\theta}_{cmd} = \dot{z}_{cmd} = 0$. Note that for the helicopter operating under gain-scheduled control, a single LTI-mode ($L = 1$) is sufficient to describe virtually the entire useful flight envelope.

The first motion planning scenario below involves one-dimensional helicopter motion, in which elevation z is assumed equal to 0. As such, flight is restricted to the positive and negative travel directions only. In this case, a suitable planning state vector in the form of Equation (5.2) is

$$\mathbf{x}[t] = \begin{bmatrix} \dot{v}[t] \\ v[t] \\ x[t] \end{bmatrix}, \tag{5.31}$$

where the $\dot{v}$ component allows for easy bounds on vehicle acceleration. With elevation reduced to level flight only, the LTI-mode system input is simply a velocity command:

$$\mathbf{u}[t] = v_{cmd}[t]. \tag{5.32}$$

The inclusion of position x in Equation (5.31) allows for specification of vehicle initial and final position, even if velocity is the only command component. To form a corresponding LTI model of the form $x[t+1] = Ax[t] + Bu[t]$, simply approximate the continuous-time velocity tracking loop as a unit-gain lightly-damped second-order system with an additional integrator for position. That is, let the velocity control loop be approximated as

$$\frac{d}{dt}\begin{bmatrix} \dot{v} \\ v \\ x \end{bmatrix} = \begin{bmatrix} -2\zeta\omega_n & -\omega_n^2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{v} \\ v \\ x \end{bmatrix} + \begin{bmatrix} \omega_n^2 \\ 0 \\ 0 \end{bmatrix} v_{cmd}, \tag{5.33}$$

where ω_n and ζ may be tuned to match the observed system closed-loop performance. In the control design of this thesis, it is suitable to choose $\omega_n = 1.5$ rad/sec and $\zeta = 1.0$. The LTI-mode A and B matrices then follow from a zero-order hold continuous-to-discrete transformation of state equations (5.33) using a $T_s = 1$ sec sample time.

In the second motion planning example, the restriction to level-only elevation is

lifted, allowing for both velocity and elevation commands. In this case, choose the five-dimensional LTI-mode state vector

$$\mathbf{x}[t] = \begin{bmatrix} \dot{v}[t] \\ v[t] \\ x[t] \\ \dot{z}[t] \\ z[t] \end{bmatrix} \tag{5.34}$$

and two-dimensional input command vector

$$\mathbf{u}[t] = \begin{bmatrix} v_{cmd}[t] \\ z_{cmd}[t] \end{bmatrix}. \tag{5.35}$$

Now augment to the continuous-time state-space model of Equation (5.33) to include an additional unit-gain second-order elevation response channel:

$$\frac{d}{dt}\begin{bmatrix} \dot{v} \\ v \\ x \\ \dot{z} \\ z \end{bmatrix} = \begin{bmatrix} -2\zeta_1\omega_{n,1} & -\omega_{n,1}^2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2\zeta_2\omega_{n,2} & -\omega_{n,2}^2 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{v} \\ v \\ x \\ \dot{z} \\ z \end{bmatrix} + \begin{bmatrix} \omega_{n,1}^2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \omega_{n,2}^2 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} v_{cmd} \\ z_{cmd} \end{bmatrix}, \tag{5.36}$$

where the velocity and elevation tracking loops are taken to be decoupled. Set the modeling constants $\omega_{n,1} = 1.5$ rad/sec and $\zeta_1 = 1.0$ as above for the velocity tracking loop. Elevation setpoint tracking occurs somewhat faster but with more overshoot, leading to values of $\omega_{n,2} = 2.0$ rad/sec and $\zeta_2 = 0.3$. Again, obtain LTI-mode matrices A and B by continuous-to-discrete conversion of Equation (5.36) with a sampling time of $T_s = 1$ sec.

5.2.2 Affine Transformation Maneuver Design

Because MILP-based path planners and solution algorithms can only accommodate discrete binary variables and continuous real variables subject to linear constraints, maneuver classes must perform *affine* transformations of the state vector. This fact is evident in Inequalities (5.12), which imply the following affine equality relation when a maneuver from class j is activated at decision step t (i.e. when $m_j[t] = 1$):

$$\mathbf{x}[t+1] = M_{s,j}\mathbf{x}[t] + M_{c,j} + \mathbf{x}[t]. \tag{5.37}$$

In addition, if the maneuver class is considered within a minimum time problem statement, the maneuver duration must also be affine in the initial state, as seen in Inequalities (5.18), which imply that

$$c_m[t] = J_{s,j}\mathbf{x}[t] + J_{c,j}, \tag{5.38}$$

when $m_j[t] = 1$.

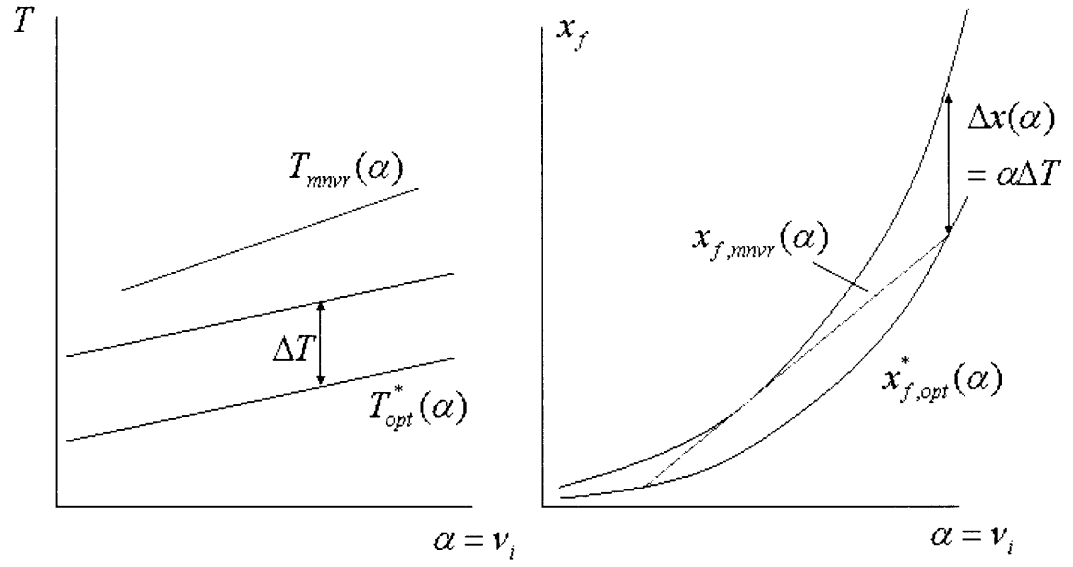


Figure 5-1: Affine maneuver transformations are dynamically feasible, although typically suboptimal.

However, a general parametrized maneuver class $p(\alpha)$ for a general nonlinear system may not necessary obey Equations (5.37) and (5.38) "naturally". That is, given an arbitrary maneuver class $p(\alpha)$ based on prototype trajectories p_1 and p_2, it is not likely that the physical state transformation and motion duration will be affine in the MILP planning vector $\mathbf{x}[t]$. (Note that in a given planning problem, the maneuver class variable α must be some function of the vector $\mathbf{x}[t]$, so that $\mathbf{x}[t]$ at the commencement of the motion uniquely defines the specific element of the maneuver class).

Indeed, the example of Figures 3-1 and 3-2 shows a case where the maneuver time duration T varies as the square root of the initial velocity $v_i = \alpha$ and is clearly not an affine function of the initial state. However, as was discussed in Section 3.1 and illustrated in Figures 3-3 and 3-7, it is possible to use available degrees-of-freedom in p to impose additional user-selected constraints on the trajectory family, achieving some useful property for $p(\alpha)$. In the cases of the figures, an additional equality constraint of the form $T(\alpha) = \hat{f}(\alpha) = 0$ made possible user-selected profiles of maneuver time throughout the entire family. In the case of Equation (5.37) (and Equation (5.38) for minimum time problems), it is a simple matter to impose extra equality constraints to enforce affine state transformations and affine maneuver times.

To show that these extra constraints do not interfere with dynamic feasibility, consider a bounded control quick-stop maneuver, such as that discussed in Section 4.5.1. Bounded control implies that the system can generate only a finite, bounded thrust magnitude to decelerate the system. An optimal minimum-time quick-stop maneuver will employ this maximum thrust regardless of the initial trim velocity $\alpha = \overline{v}_i$. As a first-order approximation, a vehicle decelerating under constant thrust obeys essentially constant acceleration kinematics for sufficiently large $\overline{v}_i$, so that the

optimal stopping distance $x^*_{f,opt}(\alpha)$ is approximately parabolic in α (assume $x_i = 0$ without loss of generality) and the minimizing time $T^*_{opt}(\alpha)$ is approximately linear in α, as illustrated in Figure 5-1. However, by building a constant initial delay ΔT into each maneuver, during which the helicopter cruises at its initial speed $v_i = \alpha$, it is a simple matter to create a suboptimal maneuver class with duration $T(\alpha) = T^*_{opt}(\alpha) + \Delta T$ and with displacement $x_f(\alpha) = x^*_{f,opt}(\alpha) + \Delta x(\alpha) = x^*_{f,opt}(\alpha) + \alpha\Delta T$. Then it is possible to (conservatively) enforce affine constraints to define a further maneuver class with affine duration function $T_{mnvr}(\alpha) = k_1\alpha > T^*_{opt}(\alpha) + \Delta T$ and affine final displacement function $x_{f,mnvr}(\alpha) = k_2\alpha \leq x^*_{f,opt}(\alpha) + \alpha\Delta T$ for some constants $k_1, k_2 > 0$. Note that other important planning variables are easily affine functions of $\overline{v}_i = \alpha$ for the quick-stop maneuver. For example, the final trim velocity $\overline{v}_f$ is identically 0, and is therefore trivially an affine function of α. Further, the final vehicle acceleration is also zero since the final state boundary condition constraints of Section 4.2.4 enforce equilibrium conditions at both beginning and end of maneuvers.

This design exercise illustrates two important points regarding maneuver classes in MILP-based path planners. First, it is a fairly simple matter to create feasible, affine maneuver transformations given the interpolation framework of Chapter 2, the instructive examples of Chapter 3, and the specific trajectory variables of Chapter 4. Secondly, the affinity requirement of MILP-based planners forces the parametrized maneuver class to have suboptimal performance, in general, relative to common cost functions. Such a "degradation" is typically not a major concern though since: (1) the optimal maneuver family can be computed off-line and affine state transformations chosen specifically to minimize the optimality gap; and (2) the performance degradation may be trivial compared to alternative planners which use only a few static maneuver elements with fixed boundary conditions, therefore forcing the vehicle to spend time reaching the required maneuver initial state and thus incurring additional cost. If desired, it is always possible to break a maneuver class up into smaller families to further reduce the overall optimality gap. The consequence then is an increase in the number of maneuver class binary variables in the MILP problem statement.

As a final note, the parametrized maneuver framework of Chapter 2 can be made to suit the requirements of very general motion planners, simply by imposing the necessary constraints on available degrees-of-freedom in p, a luxury made possible by having access to the entire space of feasible maneuvers.

5.2.3 Considerations for Real-Time Maneuver Generation

As shown in Figure 1-3, a reasonable procedure for building parametrized maneuver classes into a combined motion planning and control scheme is as follows. First, define a conceptual maneuver class and decide what constraints are necessary to describe it. For MILP-based solutions, this process may involve introducing affine state transformation constraints, such as those discussed in Section 5.2.2, to the usual vehicle dynamic feasibility and boundary condition constraints. It is of course necessary to generate feasible example maneuvers which form the guidepoints for trajectory interpolation. As discussed in Section 4.3 with regards to the 3-DOF helicopter, these

example motions may come from off-line nonlinear programming or motion capture (or even motion capture methods applied to mathematical "sketches" of desired trajectories). These methods can take many minutes to perform, given that nonlinear programming is a somewhat uncertain process with *no guarantee* of convergence for a given trajectory. Thus it is clearly necessary to generate the examples off-line.

With several known feasible motions in place and constraint functions well-defined, it is then possible to generate a continuous maneuver class by carrying out the trajectory interpolation algorithms of Section 2.3. The expressions of Section 3.2 give bounds on the expected number of elementary operations involved in interpolation but actual computing times are naturally platform and application dependent. For the helicopter maneuvering examples of Section 4.5, once a reasonable update interval (in α) for constraint derivatives is found (from experimentation, thus revealing the degree of nonlinearity in the constraint functions), an entire maneuver class involving only equality constraints can take 10 to 15 seconds to generate; incorporating full inequality constraints (without using any engineering judgment to trim superfluous components of $g(p)$) requires 1 to 3 minutes to fill in the entire maneuver class, using MATLAB® software and a 930 MHz Pentium III processor. Here, generation of the "entire" class implies continuation of the interpolation algorithms across the entire interval $[\alpha_1, \alpha_2]$, thus obtaining a complete function $p(\alpha)$. Note however that this process, which sits in the upper right-hand block of Figure 1-3, generates a family of *many* feasible trajectories in roughly the same time (and often less) than it takes to find a *single* feasible motion with nonlinear programming or motion capture.

Finally, with the entire maneuver family in hand, it is reasonable to store a sequence of maneuvers $\{p(\alpha_i)\}$, where $\{\alpha_i\}$ is some gridding, or mesh, of the interval $[\alpha_1, \alpha_2]$ of the same resolution as the constraint derivative update rate used in obtaining $p(\alpha)$. Then, given the availability $\{p(\alpha_i)\}$, it is possible to generate any new $p(\alpha)$ for any $\alpha \in [\alpha_1, \alpha_2]$ with only a *single* constraint set differentiation, making the trajectory generation process extremely fast and reasonable for real-time applications. For the maneuvers of Section 4.5 and those of the following sections, this process requires 2 to 4 seconds depending on the maneuver type and number of constraints involved. It is this operation which occurs in the bottom center block of Figure 1-3.

5.3 1-D Example: Sudden Direction Reversal and Return to Hover State

It is now possible to integrate the helicopter maneuver parametrizations of Chapter 4 into the MILP-based hybrid system framework of Section 5.1 to create a highly flexible vehicle path planning capability. This section considers a simple but informative case of a helicopter at an initial, steady cruise state traveling in the negative travel direction, but then being required to return, *in minimum time*, to a resting hover state at a position originally behind it. This scenario presents a good opportunity to employ two highly useful helicopter maneuvers, a direction reversal and a flaring quick-stop motion (similar to that seen in Section 4.5.1). The case is similar

to circumstances autonomous helicopters encounter in cluttered urban environments requiring aerobatic direction-reversing motions [133], or rapid retreat scenarios where a vehicle encounters a sudden, forward-positioned threat.

For this example, the helicopter is at a near-level elevation state throughout the entire motion, so employ the 3-component planning state vector of Equation (5.31), the scalar velocity command input of Equation (5.32), and the continuous-time LTI-mode closed-loop system model of Equation (5.33). In light of the gain-scheduled closed-loop servo design of Section 4.4, it is sufficient to consider a single LTI-mode, that is, $L = 1$. Therefore, drop the i subscript notation from Inequalities (5.9) and refer to the discrete state and input matrices simply as A and B, respectively. Note, however, that because there are two available maneuvers ($M = 2$), it is necessary to retain the j subscripts for all maneuver variables.

Consider an initial, steady cruise state defined by the state vector

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ -50 \\ 0 \end{bmatrix}, \tag{5.39}$$

where all angular units are given in degrees: the initial acceleration is $\dot{v}[0] = 0$ deg/sec^2, the initial velocity is $v[0] = -50$ deg/sec, and the initial position is selected to be $x[0] = 0$ deg. The goal state to be reached in minimum time is

$$\mathbf{x}_{goal} = \begin{bmatrix} 0 \\ 0 \\ 500 \end{bmatrix}, \tag{5.40}$$

corresponding to a steady hover state at $x_{goal} = +500$ deg, and thus requiring a change in helicopter direction. Note that since the goal state $\mathbf{x}_{goal}$ involves all three components of $\mathbf{x}$, it follows that $S = \mathbf{X} = \{1, 2, 3\}$ in Inequalities (5.15).

Define the initial velocity command input as $\mathbf{u}_0 = -50$ deg/sec, in accordance with the problem initialization of Equations (5.24), so that the vehicle will continue at a steady velocity to the first decision step of $t = 1$ where $\mathbf{x}[1] = \mathbf{x}[0]$. Additionally, the first maneuver cost term follows as $c_m[0] = 0$, simply verifying that no maneuver occurs at the $t = 0$ (problem initialization) decision step.

Now, since $L = 1$ and $M = 2$, the constraints of Inequalities (5.9) and Equation (5.13) simplify algebraically to

$$\begin{aligned} \mathbf{x}[t+1] - A\mathbf{x}[t] - B\mathbf{u}[t] &\leq K(m_1[t] + m_2[t]) \\ -\mathbf{x}[t+1] + A\mathbf{x}[t] + B\mathbf{u}[t] &\leq K(m_1[t] + m_2[t]), \end{aligned} \tag{5.41}$$

so that if no maneuver occurs at decision step t (i.e. $m_1[t] = m_2[t] = 0$), the planning dynamics follow the LTI-mode equation $\mathbf{x}[t+1] = A\mathbf{x}[t] + B\mathbf{u}[t]$. Otherwise, if a maneuver occurs (either $m_1[t] = 1$ or $m_2[t] = 1$), then Inequalities (5.41) are trivially satisfied thanks to the vector K of large numbers.

To form a well-posed guidance problem, impose 0.5 deg/sec^2 magnitude bounds on the vehicle acceleration $\dot{v}[t]$; in terms of Inequalities (5.25), it follows that $S_{\overline{x}} =$

$S_{\underline{x}} = \{1\}$ and $\overline{\mathbf{x}} = -\underline{\mathbf{x}} = 0.5$ deg/sec^2. In addition, impose 60 deg/sec magnitude bounds in the commanded velocity, giving $S_{\overline{u}} = S_{\underline{u}} = \{1\} \equiv \mathbf{U}$ with $\overline{\mathbf{u}} = -\underline{\mathbf{u}} = 60$ deg/sec, based on Inequalities (5.26).

Now, consider a simple direction reversal maneuver (given index $j = 1$) resulting in a helicopter velocity in the opposite direction but with no *net* physical displacement. That is, the helicopter decelerates to zero velocity while traveling in its initial direction, and then accelerates in the opposing direction, achieving a steady cruise state at the exact maneuver initiation position. Of course in general, there are still several indeterminate boundary condition degrees-of-freedom for the motion, with the maneuver parameter vector α therefore taking the form $\alpha = [\overline{v}_i, \overline{v}_f, T]^T$, where the steady initial and final speeds as well as the maneuver duration can vary. However, the maneuvering transition requirements of Inequalities (5.12) dictate that the vehicle post-maneuver state $\mathbf{x}[t+1]$ depend on the pre-maneuver state $\mathbf{x}[t]$ in an affine manner. As such, it is possible to apply a dimensionality reducing map of the form $\alpha = \gamma(\sigma)$ as discussed in Section 2.3.6, so that the maneuver state transition obeys the form of Inequalities (5.12) as well as the affine cost map of Inequalities (5.18). Therefore, set $\sigma \equiv \overline{v}_i$ and choose

$$\alpha = \gamma(\overline{v}_i) = \gamma(\sigma) = \begin{bmatrix} \sigma \\ -\sigma \\ c_1\sigma + c_2 \end{bmatrix}, \tag{5.42}$$

achieving a one-dimensional reversal maneuver class with final velocity exactly opposite the initial velocity and with duration T affine in the initial velocity. Analysis by the method of Section 5.2.2 shows that reasonable choices for the constants in Equation (5.42) are $c_1 = -14.99$ deg^{-1} and $c_2 = 3.69$ sec, based on user-selected feedforward control bounds of $V_{coll,max} = 2.0$ V, $V_{coll,min} = 1.0$ V and $V_{cyc,max} = -V_{cyc,min} = 0.6$ V. Note that this analysis is based conceptually on the right-hand plot in Figure 5-1, since the physical displacement function for the time-optimal reversal maneuver class is chosen constant with value 0 deg.

Given the map of Equation (5.42), it is a straightforward matter to determine the maneuver state transformation and cost matrices as

$$\begin{aligned} M_{s,1} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ M_{c,1} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ J_{s,1} &= \begin{bmatrix} 0 & c_1 & 0 \end{bmatrix} \\ J_{c,1} &= c_2, \end{aligned} \tag{5.43}$$

giving the state transformation

$$\mathbf{x}[t+1] = \begin{bmatrix} \dot{v}[t+1] \\ v[t+1] \\ x[t+1] \end{bmatrix} = \begin{bmatrix} 0 \\ -v[t] \\ x[t] \end{bmatrix} \tag{5.44}$$

and affine cost function

$$c_m[t] = c_1 v[t] + c_2. \tag{5.45}$$

To create the reversal maneuver class, parametrized by $\sigma \equiv \overline{v}_i$, impose the following nonlinear boundary condition function, based on Equations (2.43) and (4.38), replacing the symbol α by σ to obtain:

$$h_{bc}(p,\sigma) = \begin{bmatrix} v(0;p) - \sigma \\ v(1;p) + \sigma \\ z(0;p) - 0 \\ z(1;p) - 0 \\ \theta(0;p) - \overline{\theta}_i(\sigma) \\ \theta(1;p) - \overline{\theta}_f(-\sigma) \\ V_{coll}(0;p) - \overline{V}_{coll,i}(\sigma) \\ V_{coll}(1;p) - \overline{V}_{coll,f}(-\sigma) \\ V_{cyc}(0;p) - \overline{V}_{cyc,i}(\sigma) \\ V_{cyc}(1;p) - \overline{V}_{cyc,f}(-\sigma) \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \\ T\int_0^1 v(\tau;p)d\tau - 0 \\ T - (c_1\sigma + c_2) \end{bmatrix} = 0. \tag{5.46}$$

Comparison to the boundary condition vectors of Section 4.5 makes obvious the addition of the last two components of Equation (5.46), allowing for the zero net displacement and affine time constraints.

Finally, it is necessary to impose initiation bounds, so the helicopter satisfies a reasonable range of initial velocities before executing the maneuver. Choose an initial velocity requirement of $-65 \leq \sigma = \overline{v}_i \leq -5$ deg/sec giving, in the terminology of Inequalities (5.27), $\overline{I}_1 = \underline{I}_1 = \{2\}$ with $\overline{\mathbf{x}}_{1,0} = -5$ deg/sec and $\underline{\mathbf{x}}_{1,0} = -65$ deg/sec.

A similar procedure allows for the design of quick-stop maneuver (given index $j = 2$), similar to that of Section 4.5.1 but now satisfying the requirements of the MILP-based planning framework. The quick-stop final state is defined to be a steady hover condition, leaving available the maneuver degrees-of-freedom given by $\alpha = [\overline{v}_i, x_f, T]^T$. Here, x_f denotes the net maneuver displacement, which is clearly not equal to zero across the maneuver class, unlike the reversal maneuver.

This time, choose a dimensionality reducing map of the form $\alpha = \gamma(\sigma)$, with

$\sigma \equiv \overline{v}_i$ again, but this time given by

$$\alpha = \gamma(\overline{v}_i) = \gamma(\sigma) = \begin{bmatrix} \sigma \\ c_3\sigma + c_4 \\ c_5\sigma + c_6 \end{bmatrix}. \tag{5.47}$$

Note the affine expressions for displacement x_f and duration T in terms of the initial velocity. Again, imposing helicopter control bounds $V_{coll,max} = 2.0$ V, $V_{coll,min} = 1.0$ V, and $V_{cyc,max} = -V_{cyc,min} = 0.6$ V, and then following an analysis identical to that in Section 5.2.2, gives constants for Equation (5.47) as follows: $c_3 = 5.89$ sec, $c_4 = -33.40$ deg, $c_5 = 7.58$ sec^2/deg, and $c_6 = 4.08$ sec.

The maneuver transformation and cost matrices then follow as

$$\begin{aligned} M_{s,2} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & c_3 & 0 \end{bmatrix} \\ M_{c,2} &= \begin{bmatrix} 0 \\ 0 \\ c_4 \end{bmatrix} \\ J_{s,2} &= \begin{bmatrix} 0 & c_5 & 0 \end{bmatrix} \\ J_{c,2} &= c_6, \end{aligned} \tag{5.48}$$

with the resulting state transformation

$$\mathbf{x}[t+1] = \begin{bmatrix} \dot{v}[t+1] \\ v[t+1] \\ x[t+1] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ c_3 v[t] + c_4 \end{bmatrix} \tag{5.49}$$

and cost function

$$c_m[t] = c_5 v[t] + c_6. \tag{5.50}$$

As in Equation (5.46), add position change and maneuver duration components to the standard nonlinear boundary condition constraints to obtain:

$$h_{bc}(p,\sigma) = \begin{bmatrix} v(0;p) - \sigma \\ v(1;p) - 0 \\ z(0;p) - 0 \\ z(1;p) - 0 \\ \theta(0;p) - \overline{\theta}_i(\sigma) \\ \theta(1;p) - \overline{\theta}_{f,hov} \\ V_{coll}(0;p) - \overline{V}_{coll,i}(\sigma) \\ V_{coll}(1;p) - \overline{V}_{coll,hov} \\ V_{cyc}(0;p) - \overline{V}_{cyc,i}(\sigma) \\ V_{cyc}(1;p) - \overline{V}_{cyc,hov} \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \\ T\int_0^1 v(\tau;p)d\tau - (c_3\sigma + c_4) \\ T - (c_5\sigma + c_6) \end{bmatrix} = 0, \tag{5.51}$$

which then defines the quick-stop maneuver class for the trajectory interpolation algorithms.

A reasonable maneuver initial speed range is $10 \leq \sigma \equiv \overline{v}_i \leq 60$ deg/sec so that $\overline{I}_2 = \underline{I}_2 = \{2\}$, $\overline{\mathbf{x}}_{2,0} = 60$ deg/sec, and $\underline{\mathbf{x}}_{2,0} = 10$ deg/sec, in the terminology of Inequalities (5.27).

With the helicopter one-dimensional LTI-mode and two parametrized maneuver classes defined, it is now possible to summarize the formal minimum time problem statement as:

$$\min_{m_1[t],m_2[t],d[t];\mathbf{x}[t],\mathbf{u}[t],c_m[t]} J, \tag{5.52}$$

where Equation (5.17) defines the guidance trajectory time cost function J. The MILP constraints are: the LTI-mode dynamics of Inequalities (5.41); the maneuvering state update relations of Inequalities (5.12); the goal state requirement of Inequalities (5.15); the goal uniqueness requirement of Inequalities (5.16); the maneuvering cost Inequalities of (5.18) and (5.19); the state bounds of Inequalities (5.25); the command bounds of Inequalities (5.26); the maneuver authorization bound of Inequalities (5.27); and finally the (optional) maneuver execution limits of Inequality (5.28), with $N_m = 2$ selected for this problem instance.

Note that a guidance decision horizon of $H = 20$ is sufficient to find a solution, since the goal state is reached at $t = 9$. Figures 5-2 and 5-3 give the resulting optimal reference position and velocity guidance trajectories, respectively. After the required initial one decision step hold (due to the initial control selection of $\mathbf{u}_0 = -50$ deg/sec), the helicopter immediately executes a reversal maneuver with $\overline{v}_i = -50$ deg/sec, then cruises in a slowly-varying LTI-mode until a quick-stop maneuver, executed at $\overline{v}_i = +48.48$ deg/sec, brings the vehicle to an exact rest at $x = +500$ deg.

In the figure, the solid line gives the MILP solution references while the dashed line shows the actual experimental closed-loop tracking performance for the heli-

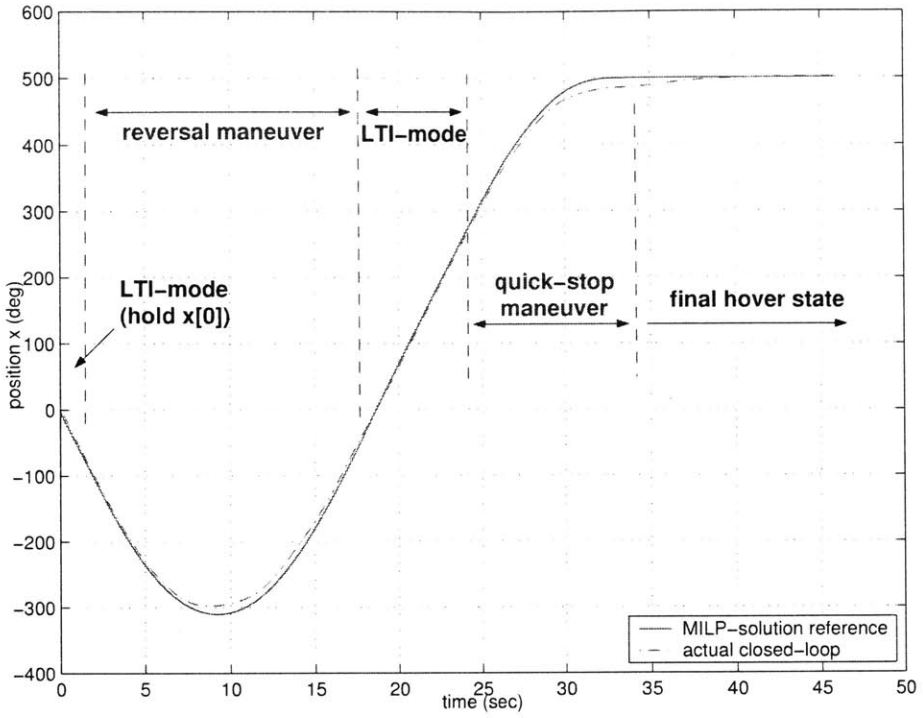


Figure 5-2: Path planning travel position solution for one-dimensional retreat-to-hover scenario.

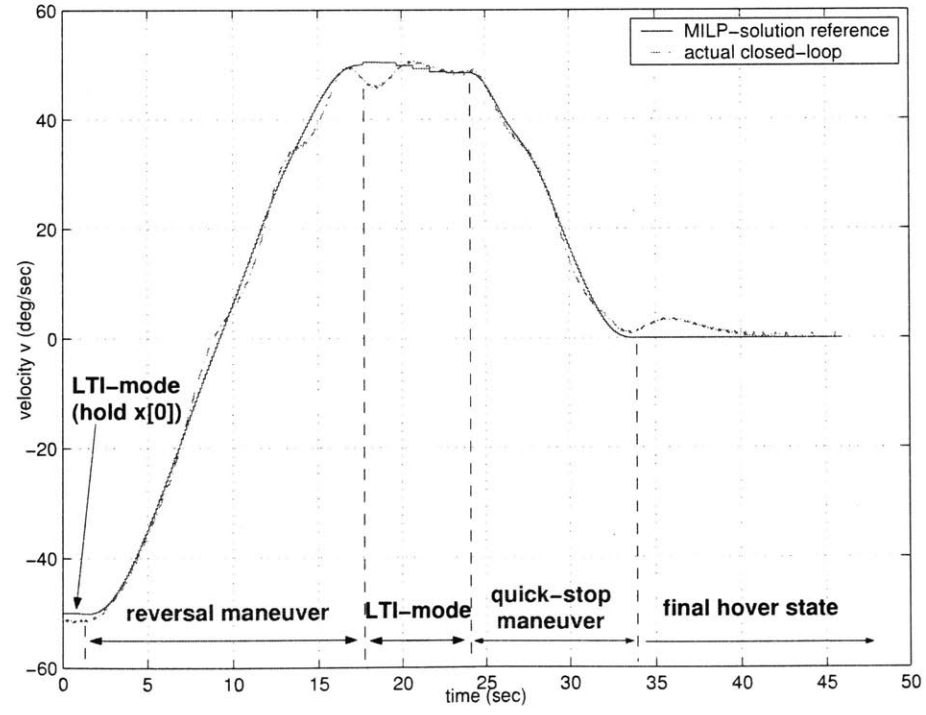


Figure 5-3: Path planning velocity solution for one-dimensional retreat-to-hover scenario.

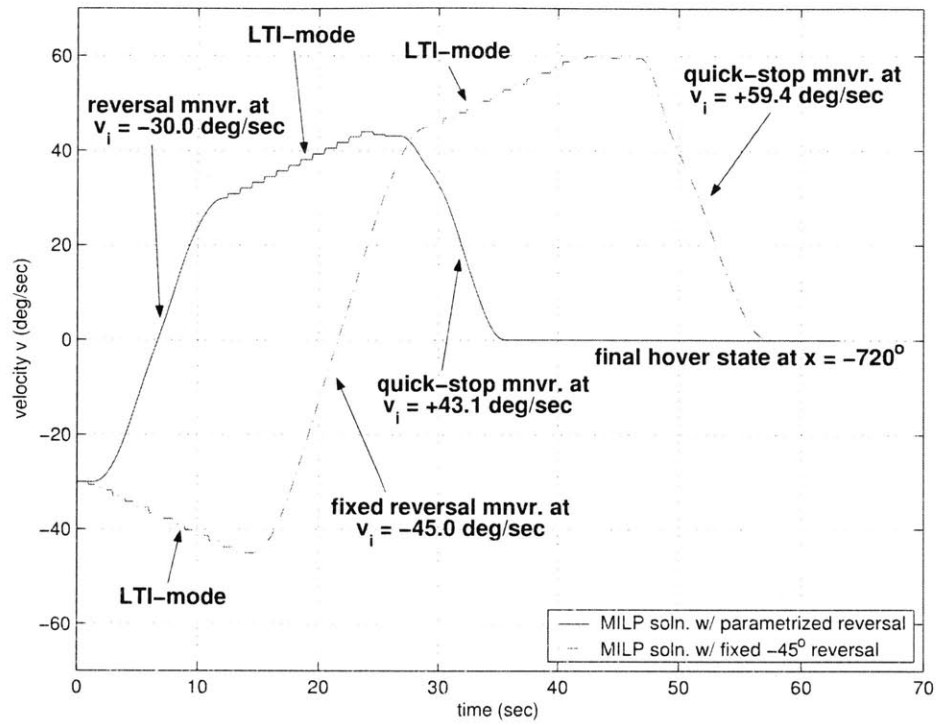


Figure 5-4: Comparison of MILP velocity solutions for parametrized and fixed maneuver classes.

copter. Note that since there is no acceleration tracking loop, LTI-mode velocity values are held constant between decision steps while the position reference follows from linear interpolation of the MILP solutions. For the maneuvers, reference signals follow from continuous-time spline evaluations, as discussed in Chapter 4. As seen in Figures 5-2 and 5-3, the overall tracking performance validates the feasibility of the combined parametrized maneuver-MILP guidance framework. The control system corrects position and velocity errors at the end of each maneuver segment; nonlinear tracking methods superior to the LQ-servo design of Section 4.4 would only lead to better real-world performance.

As mentioned in Section 1.3.2, the addition of parametrized maneuvers to hybrid motion planners allows for increased flexibility in vehicle guidance problems. Specifically, when the preceding one-dimensional scenario is solved for a wide range of initial velocity conditions, the planner repeatedly executes the reversal maneuver at the first available decision step, then cruises in an LTI-mode before executing a final quick-stop maneuver. In several existing hybrid motion planners, such as those of references [51, 100, 133], maneuvers are fixed objects with specific, invariant initial conditions. As such, if the helicopter initial condition were fixed at a constant in the present planner that did not *exactly* match the reversal boundary conditions, then the MILP optimal solution would require an initial LTI-mode segment to reach the reversal velocity requirement, increasing the overall trajectory time and taking the helicopter initially further away from its goal state. Figures 5-4 and 5-5 show the reference velocity and position solutions, respectively, for a comparison for a

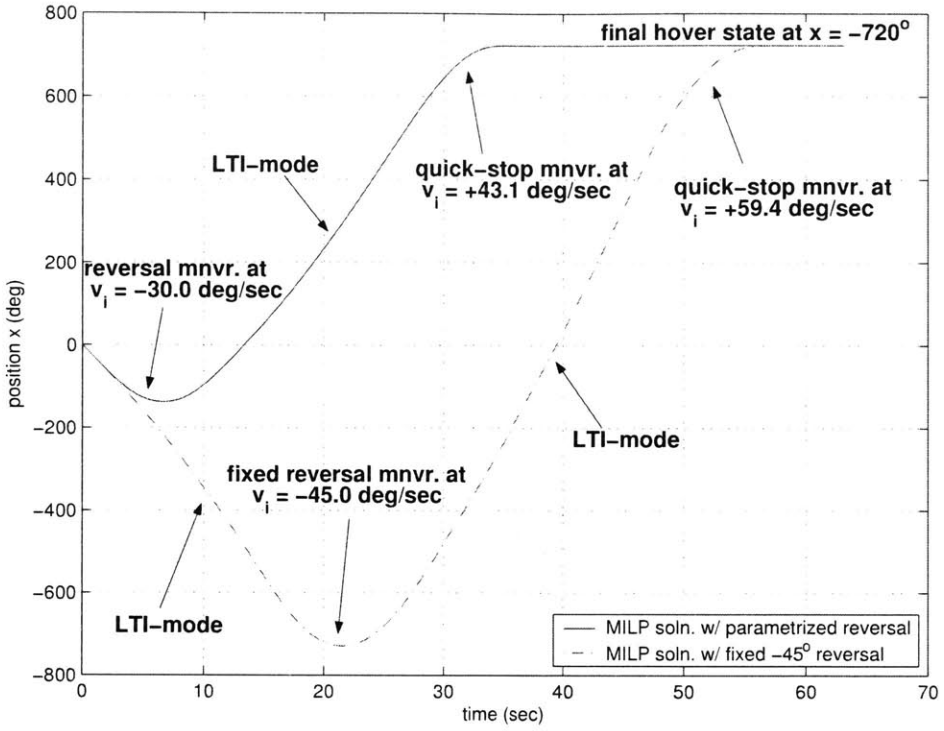


Figure 5-5: Comparison of MILP position solutions for parametrized and fixed maneuver classes.

parametrized reversal family versus a fixed reversal maneuver with a required initial velocity of $\overline{v}_i = -45$ deg/sec. (Note that for the figures, the helicopter initial cruise state is $\mathbf{x}[0] = [0, -30, 0]^T$, the initial command hold is $\mathbf{u}_0 = -30$ deg/sec, the goal state is $\mathbf{x}_{goal} = [0, 0, -720]^T$ and the LTI-mode acceleration bounds are $\overline{x} = -\underline{x} = 1$ deg/sec^2).

Table 5.1 shows the optimal total trajectory time for a series of fixed initial condition reversal maneuvers (based on the problem initial and goal variables discussed above parenthetically). If the invariant maneuver initial velocity $v_{i,fixed}$ agrees exactly with the helicopter initial velocity of $v[0] = -30$ deg/sec, the planning solution exactly matches the parametrized reversal case (solid line solution of Figures 5-4 and 5-5) the minimum planned trajectory time is 36.30 sec. However, when $v_{i,fixed}$ does not match $v[0]$, the trajectory time must increase since the vehicle must first fly in an LTI-mode to reach the reversal maneuver initiation speed.

Of course, for the planners of [51, 100, 133], it would be advisable to include a *set* of fixed reversal motions, not just a single instance, thus placing a bound on the optimal trajectory times in the second column of Table 5.1. However, each additional maneuver requires and extra binary variable at every time step in the cases of [133], or an increase in dynamic program dimensionality in [51, 100].

However, as a caution on a shortcoming of the MILP-based planning method (not necessarily on the parametrized maneuver approach), Table 5.2 shows the MILP solution computing times for various planning horizon lengths H on a Pentium IV processor. For the $v[0] = -30$ deg/sec case with final goal position of $x_{goal} = +720$

Fixed $v_{i,fixed}$ (deg/sec)	Total Trajectory Time (sec)
-10	64.89
-15	58.14
-20	51.41
-25	44.71
-30 = v[0]	36.30
-35	44.52
-40	51.35
-45	58.38
-50	66.07

Table 5.1: Minimizing total trajectory times for fixed initial velocity direction reversals.

Horizon H	Solution Time (sec)
17	1.6
20	2.5
30	17
40	99

Table 5.2: MILP solution times for various planning horizon lengths for the $v[0] = -30$ deg/sec, including the parametrized reversal and quick-stop maneuvers.

deg, the goal position is reached at the seventeenth decision step, i.e. $t = 17$. As seen in the table, if H is set to 17 exactly, the MILP solution time is low, since there are exactly as many binary variables (one per decision step for each of $m_1[t]$, $m_2[t]$, and $d[t]$) as required for a well-posed problem. However, as H increases, the computing time increases essentially exponentially, making clear the need for careful selection of H in general problem instances, most likely as part of a receding horizon guidance implementation as seen in references [132] and [134].

5.4 2-D Example: Stealthy Flight Through an Obstacle Field

For the second vehicle guidance example, consider the full travel and elevation capabilities of the three degree-of-freedom helicopter in an interesting tactical situation. Figure 5-6 illustrates a scenario in which the helicopter emulates an airborne vehicle moving through an obstacle field in a hazardous, adversarial environment. The helicopter must move efficiently from its starting position, noted at the left of the figure by a circle, to a safe goal position, noted right-of-center by a star, while passing over obstacles and maintaining a low profile when exposed to a threat. This threat could be a radar sensing station or weapons emplacement, requiring the helicopter to fly with a very low pitch profile above a certain elevation (shown in the figure as the "threat zone", above the dash-dot boundary line at elevation z_B). In contrast, while in the "safe zone" (below elevation z_B) the vehicle is free to advance rapidly, with large horizontal accelerations and large pitch angles.

This section will illustrate the use of a specially designed "fast-advance" maneuver to help the helicopter reach the goal state efficiently while subject to a velocity limit in the threat zone. This maneuver begins and ends at the same cruise state, parametrized by a velocity-elevation pair, but invokes a rapid acceleration and deceleration to quickly cover the horizontal space between obstacles. This maneuver class can easily accommodate bounds on vehicle elevation and pitch angle, if desired, helping to minimize the helicopter threat exposure. Although this scenario does involve a parametrized maneuvering capability, this time in terms of independent velocity and elevation boundary conditions, the emphasis here is on using the MILP planning framework and specialized maneuvers to address interesting tactical challenges.

Unlike the preceding one-dimensional example, this planner will use the minimum error objective formulation of Equations (5.21) through (5.23). As will be seen, the threat zone thresholding and obstacle avoidance requirements introduce several binary variables at each decision step t and thus increase the MILP optimization solution time. Therefore, it is useful to avoid introducing the goal attainment binary variable $d[t]$ and instead let an accumulated error function drive the helicopter towards the goal state. In addition, unlike the minimum time formulation, which required the maneuver class duration $c_m[t]$ to be an affine function of the initial maneuvering state, the minimum error framework uses only state values at individual decision steps, as seen in the objective function J of Equation (5.21). Therefore, the maneuver

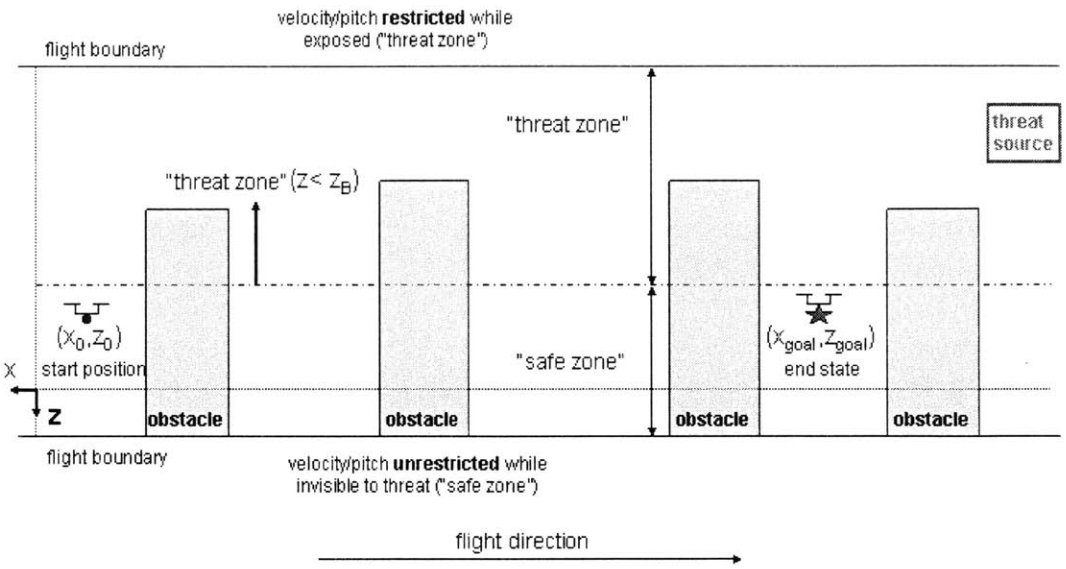


Figure 5-6: Mission scenario for the two-dimensional planning problem.

class time duration may be as low as possible, subject to vehicle control and state bounds, as long as the net state transition is an affine function. Since the fast-advance maneuvers of this section will involve jumps over a fixed horizontal displacement, the affine state transition requirement is easy to enforce.

As with the preceding one-dimensional planning example, it is sufficient to consider a single LTI-mode ($L = 1$), since the system operates under the gain-scheduled closed-loop control architecture. However, the LTI model is now a discrete-time sampled version the five-state planning dynamics of Equation (5.36) with the combined travel-elevation state vector of Equation (5.34) and the velocity-elevation command input vector of Equation (5.35).

There will be only one fast-advance maneuver class, but, as will be seen shortly, the ability to execute the motion in different physical regions of the planning horizon will necessitate two maneuver binary variables, $m_1[t]$ and $m_2[t]$, giving $M = 2$.

Consider the helicopter trajectory beginning at a steady, hover state at the (x, z) coordinate frame origin, so that $\mathbf{x}_0 = [0, 0, 0, 0, 0]^T$. As seen in Figure 5-6, the helicopter mission goal state is given by the two-component position vector $\mathbf{x}_{goal} = [x_{goal}, z_{goal}]^T$, so that $E = \{3, 5\} \subset \mathbf{X}$ in terms of the error state relations of Inequalities (5.22). Note that in the minimum error planning formulation, it is sufficient to specify a final *position* only since, as seen in Equation (5.21), the cost accumulates over the entire planning horizon H, tending to favor solutions in which the helicopter has low velocity in a neighborhood of the goal position.

As discussed above and illustrated in Figure 5-6, it is desirable to keep a low vehicle pitch angle (and speed) when in the threat zone. In terms of the vehicle planning variables of Equation (5.34), a fixed speed limit at high elevations takes the form $|v[t]| \leq v_{bound}$ when $z[t] \leq z_B$, for some suitable "speed limit" v_{bound}. (Recall that the z coordinate is positive *down*wards, so $z[t] \leq z_B$ corresponds to a vehicle

elevation *above* z_B). To build this elevation-dependent velocity bound into the MILP planner, introduce a binary variable $\lambda[t]\ \forall\ t \in \mathbf{H}$, with $\lambda[t] = 1$ if $z[t] \leq z_B$ and $\lambda[t] = 0$ otherwise. The speed bounds then follow easily, $\forall\ t \in \mathbf{H}$, as

$$\begin{aligned} \mathbf{x}_{B_1}[t] - v_{bound} &\leq K(1 - \lambda[t]) \\ -\mathbf{x}_{B_1}[t] - v_{bound} &\leq K(1 - \lambda[t]) \end{aligned} \tag{5.53}$$

with $B_1 = \{2\} \subset \mathbf{X}$ extracting only the velocity component of $\mathbf{x}$. To ensure that $\lambda[t]$ makes the desired switch between 0 and 1 when the vehicle crosses over elevation z_B, employ a constraint set, $\forall\ t \in \mathbf{H}$, of the form

$$\begin{aligned} \mathbf{x}_{B_2}[t] - z_B &\leq K(1 - \lambda[t]) \\ -\mathbf{x}_{B_2}[t] + z_B &\leq K\lambda[t], \end{aligned} \tag{5.54}$$

where $B_2 = \{5\} \subset \mathbf{X}$. This sort of state-dependent binary variable switch was seen earlier for maneuver initiation bounds (Inequalities (5.27)) and is also useful for switching between different LTI models, as in Inequalities (5.9).

Now, considering the fast-advance maneuver design for this scenario, the agile motion involves a rapid acceleration and deceleration from cruise, covering a horizontal distance much faster than could be accomplished in the linearly controlled LTI-mode. In the scenario of Figure 5-6, the motion has the utility of quickly moving (at low elevation) across the horizontal space between obstacles. For an airborne helicopter in a dangerous combat or reconnaissance mission, similar maneuvers might involve rapidly dashing through open, exposed spaces that lie between shielding obstacles (such as across streets or through open glades, for example).

In the present case, the fast-advance maneuver is defined as a rapid motion across a fixed, horizontal displacement Δ_x. The variable boundary conditions then involve the velocity and elevation values at the beginning and end of the motion, so consider a four-dimensional maneuver parameter $\alpha \equiv [\overline{v}_i, \overline{z}_i, \overline{v}_f, \overline{z}_f]^T$. Note that there is no need to explicitly control the maneuver duration in the MILP error minimization framework, so the duration T depends on α implicitly through the trajectory interpolation process.

To enforce an affine maneuvering state transformation as well as reduce interpolation dimensionality, apply a map of the form $\alpha = \gamma(\sigma)$ where the new independent argument has two components equal to the initial velocity and elevation, according to $\sigma = [\sigma_1, \sigma_2]^T \equiv [\overline{v}_i, \overline{z}_i]^T$. Then simply choose the final velocity and elevation to match the corresponding initial quantities according to $\gamma(\sigma) = [\sigma_1, \sigma_2, \sigma_1, \sigma_2]^T$. Interpolation of this maneuver class follows from one of the methods of Section 2.3.6, Figure 2-5, by performing three single-variable interpolations based on four prototype example trajectories generated from minimum-time (i.e. agile motion) nonlinear programming solutions.

In terms of the two components of σ, the parametrized maneuver boundary con-

dition takes the form

$$h_{bc}(p,\sigma) = \begin{bmatrix} v(0;p) - \sigma_1 \\ v(1;p) - \sigma_1 \\ z(0;p) - \sigma_2 \\ z(1;p) - \sigma_2 \\ \theta(0;p) - \overline{\theta}_i(\sigma_1,\sigma_2) \\ \theta(1;p) - \overline{\theta}_f(\sigma_1,\sigma_2) \\ V_{coil}(0;p) - \overline{V}_{coil,i}(\sigma_1,\sigma_2) \\ V_{coil}(1;p) - \overline{V}_{coil,f}(\sigma_1,\sigma_2) \\ V_{cyc}(0;p) - \overline{V}_{cyc,i}(\sigma_1,\sigma_2) \\ V_{cyc}(1;p) - \overline{V}_{cyc}(\sigma_1,\sigma_2) \\ (1/T)z'(0;p) - 0 \\ (1/T)z'(1;p) - 0 \\ (1/T)\theta'(0;p) - 0 \\ (1/T)\theta'(1;p) - 0 \\ T\int_0^1 v(\tau;p)d\tau - \Delta_x \end{bmatrix} = 0, \tag{5.55}$$

where the two-argument dependence of the pitch and voltage boundary conditions follows from Equation (4.22). For the fast-advance motion, the matrices for Inequalities (5.12) take the particularly simple form

$$M_s = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad M_c = \begin{bmatrix} 0 \\ 0 \\ \Delta_x \\ 0 \\ 0 \end{bmatrix}, \tag{5.56}$$

leading to the following state transition between decision steps:

$$\mathbf{x}[t+1] = \begin{bmatrix} \dot{v}[t+1] \\ v[t+1] \\ x[t+1] \\ \dot{z}[t+1] \\ z[t+1] \end{bmatrix} = \begin{bmatrix} 0 \\ v[t] \\ x[t] + \Delta_x \\ 0 \\ z[t] \end{bmatrix}. \tag{5.57}$$

Note that during the interpolation process, it is possible to impose certain state bounds that may be useful in keeping the helicopter concealed. For example, corresponding to the maneuver velocity profiles of Figure 5-7 (which shows the interpolation along the $\sigma_1 = \overline{v}_i$ axis; note the bell-shaped acceleration-deceleration profile), the helicopter pitch may be bounded, if desired, as seen in Figure 5-8, and the elevation signal itself can be bounded below the $z = z_B$ boundary, as seen in Figure 5-9. In that figure, and those that follow, $z_B = -5$ deg, that is, z_B is 5 degrees *above* the "level" elevation.

As seen in Figure 5-6, the helicopter must avoid three obstacles on its way to the goal state, necessitating the collision avoidance methods of Inequalities (5.29) and

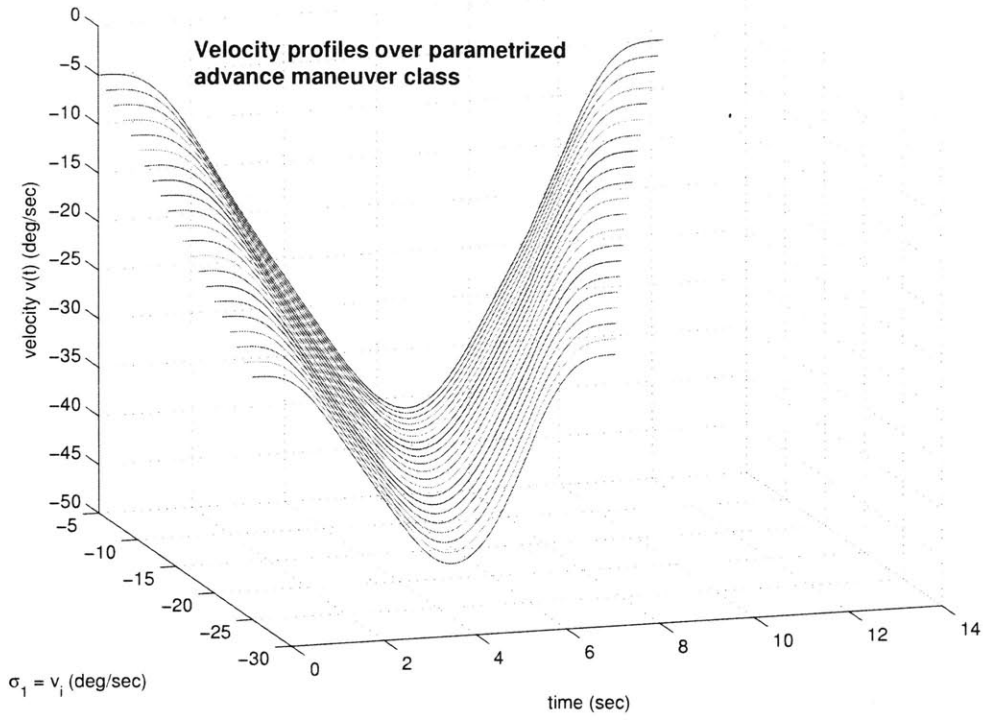


Figure 5-7: Velocity profiles for fast-advance maneuver class.

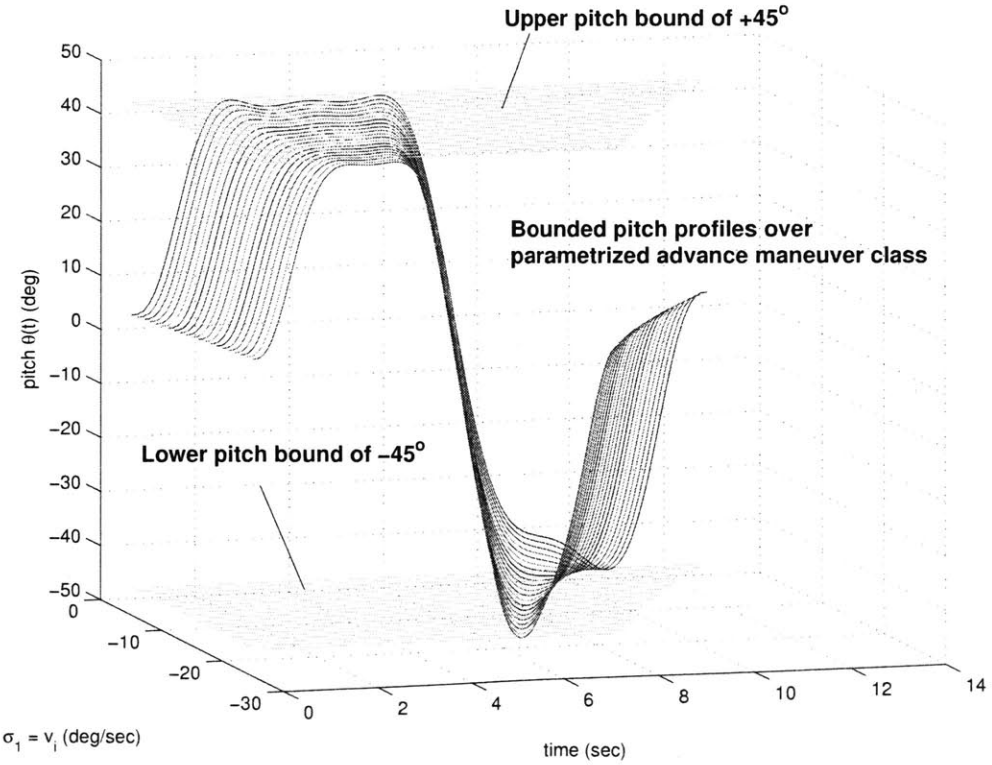


Figure 5-8: Bounded pitch profiles for fast-advance maneuver class.

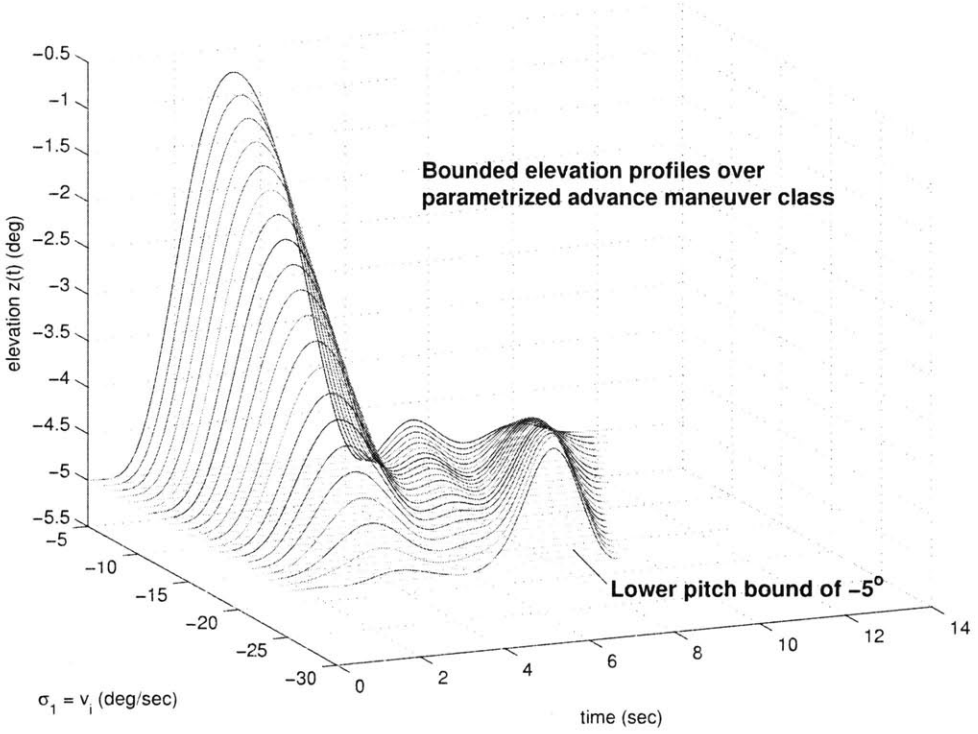


Figure 5-9: Bounded elevation profiles for fast-advance maneuver class.

(5.30). In the present case, the obstacle boundaries align exactly with the planning coordinate directions, eliminating the need for the direction-orienting vectors M_i^q and making the constraint formulation particularly simple.

For example, consider the leftmost obstacle in Figure 5-6 and assign it index $q = 1$. Assume that its left vertical face sits at $x = x_a$; its right vertical face sits at $x = x_b$, and the top face sits at $z = z_c$. (In the present scenario, the obstacles extend into the lower flight limit of $z = +10$ deg, making consideration of a fourth face unnecessary). In terms of Inequalities (5.29), the planning requirement $x[t] \geq x_a$ becomes $-(\mathbf{x}_{S^1_{f_1}}[t] - x_a) \leq Kb^1_{f_1}[t]$, with $S^1_{f_1} = \{3\} \subset \mathbf{X}$. Similarly, $x[t] \leq x_b$ becomes $+(\mathbf{x}_{S^1_{f_2}}[t] - x_b) \leq Kb^1_{f_2}[t]$, with $S^1_{f_2} = \{3\} \subset \mathbf{X}$; and $z[t] \leq z_c$ becomes $+(\mathbf{x}_{S^1_{f_3}}[t] - z_c) \leq Kb^1_{f_3}[t]$, with $S^1_{f_3} = \{5\} \subset \mathbf{X}$.

In any planning situation, it is useful to place maneuver initiation requirements on the state vector $\mathbf{x}[t]$ so that the ensuing state transition makes physical sense and is consistent with the domain of the parametrized maneuver class $p(\alpha)$. In the preceding one-dimensional study, the maneuver authorization bounds required the helicopter to be in a certain speed range. In the present case, speed ranges are also useful, where a minimum initiation speed v_{min} satisfying $|v_{min}| > |v_{bound}|$ implicitly guarantees that the fast advance maneuver will not occur in the "threat zone".

In addition, position requirements $x_{min} \leq x[t] \leq x_{max}$ based on known obstacle locations and the advance displacement value Δ_x ensure that the planner will not use the maneuver to jump "through" the obstacles, clearly a physical impossibility. Such bounds maintain the real-world feasibility of the MILP solution while helping

to reduce the overall optimization search space. In the parlance of Inequalities (5.27), choose $\overline{I}_j = \underline{I}_j = \{2, 3\} \subset \mathbf{X}$ with $\overline{\mathbf{x}}_{j,0} = [v_{max,j}, x_{max,j}]^T$ and $\underline{\mathbf{x}}_{j,0} = [v_{min,j}, x_{min,j}]^T$ as the general fast-advance initiation bounds.

Examining the physical layout of Figure 5-6, it is possible to execute advance maneuvers in the two "safe zones" between obstacles 1, 2, and 3. Since the position bounds $x_{min,j}$ and $x_{max,j}$ for these two zones define a disjoint set, use two maneuver binaries $m_1[t]$ and $m_2[t]$, each with the same state transition of Equation (5.56), to define the two possible fast-advance opportunities. (An alternative is to use a single maneuver binary $m[t]$ to cover both regions, but the "exclusive-or" logic required to define a nonconvex initial position set will itself require another binary variable).

Given the mission scenario of Figure 5-6 and the constraint and maneuver types just discussed, the minimum-error MILP guidance problem statement can be summarized as follows: perform the optimization

$$\min_{m_1[t], m_2[t], \lambda[t]; \mathbf{x}[t], \mathbf{u}[t]} J, \tag{5.58}$$

where Equation (5.21) defines the guidance error cost function (use $W = I_{2x2}$ to weight x and z errors equally). The constraint sets are: Inequalities (5.22) which define the guidance error function; Inequalities (5.41), defining the LTI-mode dynamics; Inequalities (5.12), giving the maneuvering state transitions based on the matrices of Equation (5.56); Inequality (5.28), which limits the number of maneuvers allowed (here, allow only two fast-advances); Inequalities (5.25), used to define upper and lower flight space elevation bounds; Inequalities (5.26), used to enforce bounds on velocity commands; extra linear bounds on input magnitude changes for both velocity and elevation (to prevent erratic motions that might expose the vehicle to the threat as well as limit inter-sample obstacle constraint violation); Inequalities (5.53) and (5.54), which limit the helicopter speed in the "threat zone"; Inequalities (5.29) and (5.30), applied for all three obstacles; and finally, Inequalities (5.27) to enforce the maneuver initiation velocity and position bounds.

Figure 5-10 shows the MILP planner solution for a problem instance where the three obstacles are all 360 degrees apart (1 travel revolution), 40 degrees wide, -30 degrees high (i.e. 30 degrees *above* a level elevation), and the helicopter speed above the $z_B = -5$ deg visibility line is restricted to $v_{bound} = 10$ deg/sec. As is clear from the figure, the helicopter uses a sequence of LTI-modes and advance maneuvers to reach the goal position at $(x_{goal}, z_{goal}) = (-800, 0)$ deg, corresponding to a total travel of 2.2 revolutions. The minimum error objective function, even though not explicitly penalizing trajectory time, induces the vehicle to move as rapidly as possible, subject to the v_{bound} restriction in the threat zone. The MILP solution drives the helicopter velocity v to exactly the 10 deg/sec bound while flying over each of the three obstacles. The large position jumps between the obstacles and at the speed threshold elevation correspond to two executions of the fast-advance maneuver, designed here with $\Delta_x = -270$ deg, or three quarters of a revolution. (In the MILP solution space, these large jumps correspond to activations of the maneuver binaries: $m_1[11] = 1$ and $m_2[21] = 1$).

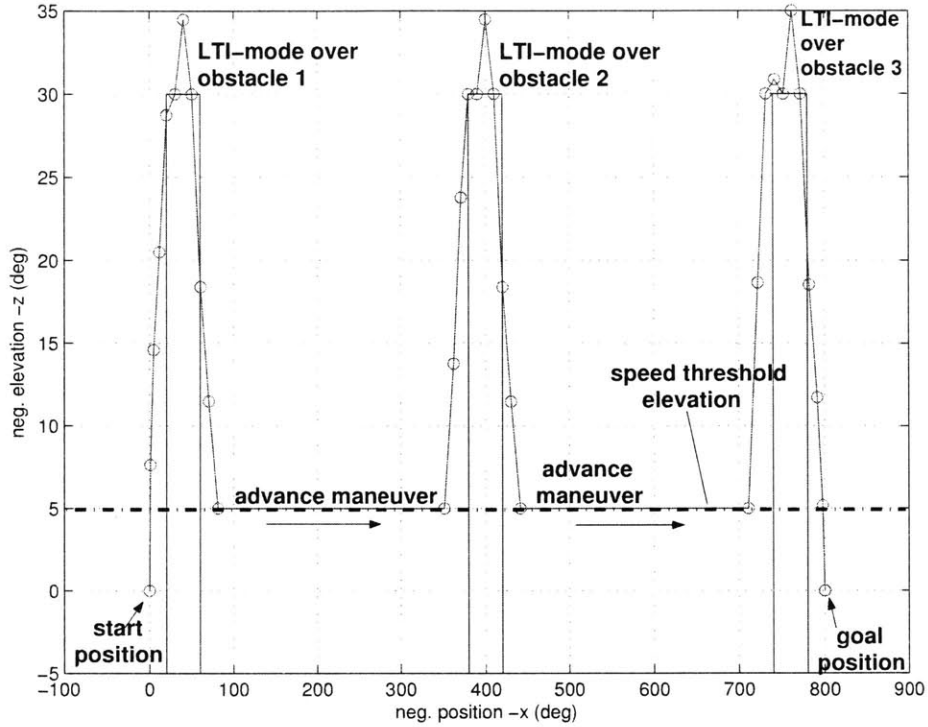


Figure 5-10: MILP planner solution with -30 degree high obstacles and $x_{goal} = -800$ deg.

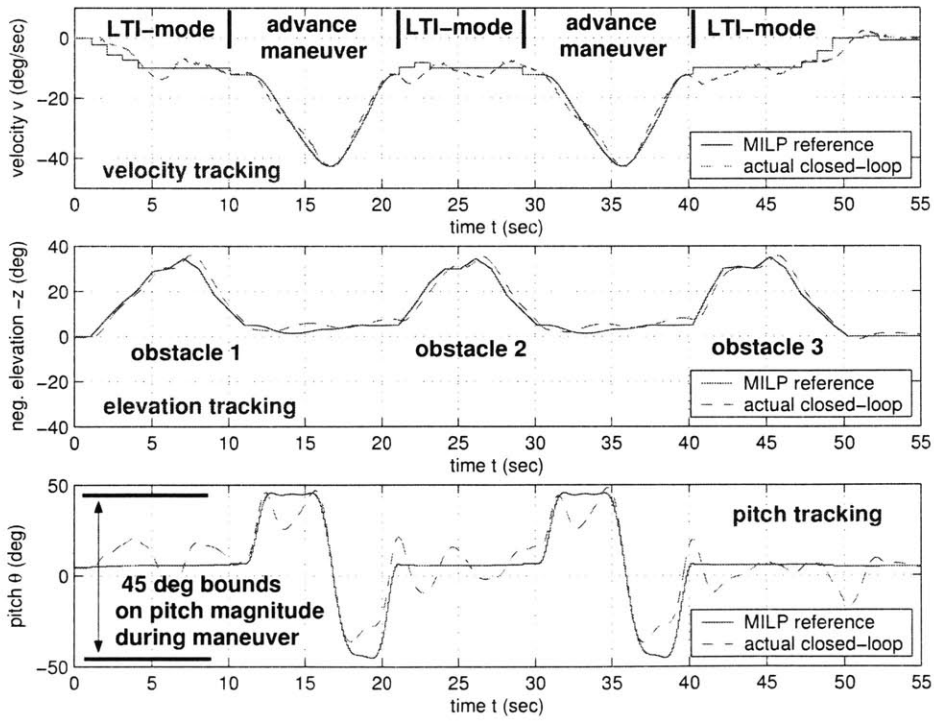


Figure 5-11: Reference and closed-loop tracking of helicopter velocity, elevation, and pitch guidance solutions.

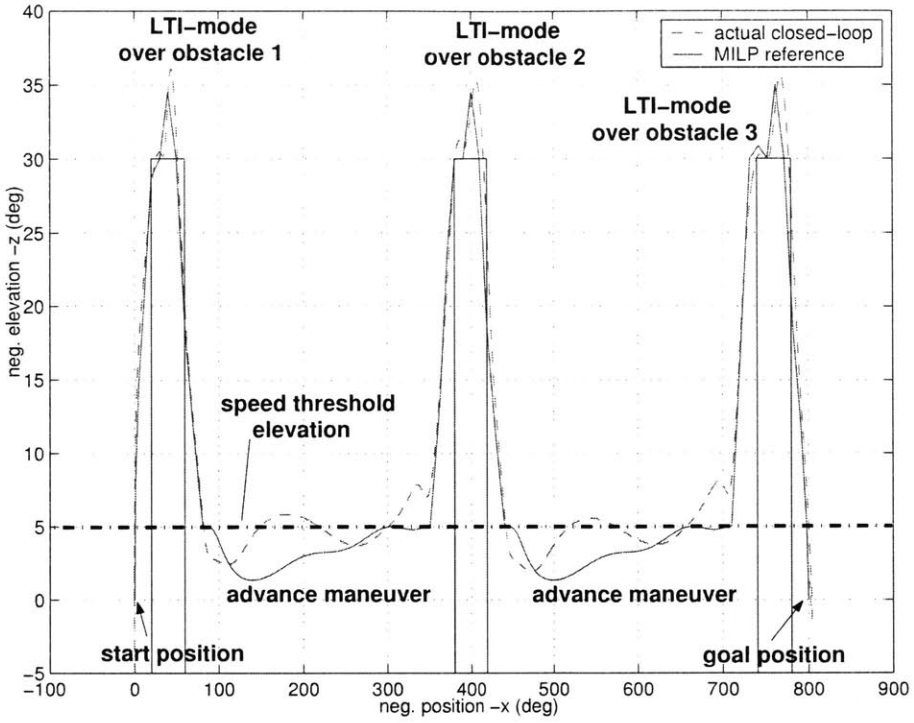


Figure 5-12: Actual closed-loop tracking of MILP reference path in Figure 5-10.

Figure 5-11 fills in the trajectory details, showing the extraction of the discrete decision step MILP solution into continuous-time reference trajectories (solid lines) for each of the velocity, elevation, and pitch signals. In addition, the dashed line shows the actual closed-loop tracking performance of the experimental helicopter. As is clear from the first subplot, the helicopter accelerates rapidly during the advance maneuvers, more than quadrupling its speed to over -40 deg/sec in accordance with the interpolated maneuver class (see Figure 5-7). As seen in the second subplot of Figure 5-11, closed-loop elevation tracking generally falls within ± 4 deg error bounds, with best performance occurring during the three, slower LTI-mode segments. Interestingly, the third plot shows the pitch reference over the entire flight, with the advance maneuvers obeying the ± 45 deg pitch bounds, as required in Figure 5-8. The pitch reference during the LTI-mode segments, as well as input voltage references, follow from a quasi-steady approximation of the overall helicopter state given by Equation (4.22). This approximation leads to fairly conservative pitch references during the LTI-modes, but is highly useful in providing low-magnitude pitch references in light of the stealthiness, low-profile tactical objective when flying above the visibility line. Indeed, the closed-loop pitch angle magnitude never exceeds 20 deg during any LTI-mode, while more than doubling up to 45 deg during the acceleration portion of the advance maneuver, illustrating the aggressiveness of the motion and the need for nonlinear methods in describing these agile maneuvers.

Figure 5-12 combines travel and elevation tracking signals into an experimental closed-loop verification of the MILP solution in Figure 5-10. Overall position tracking quality is high, with the greatest error of around 4 deg in elevation occurring during

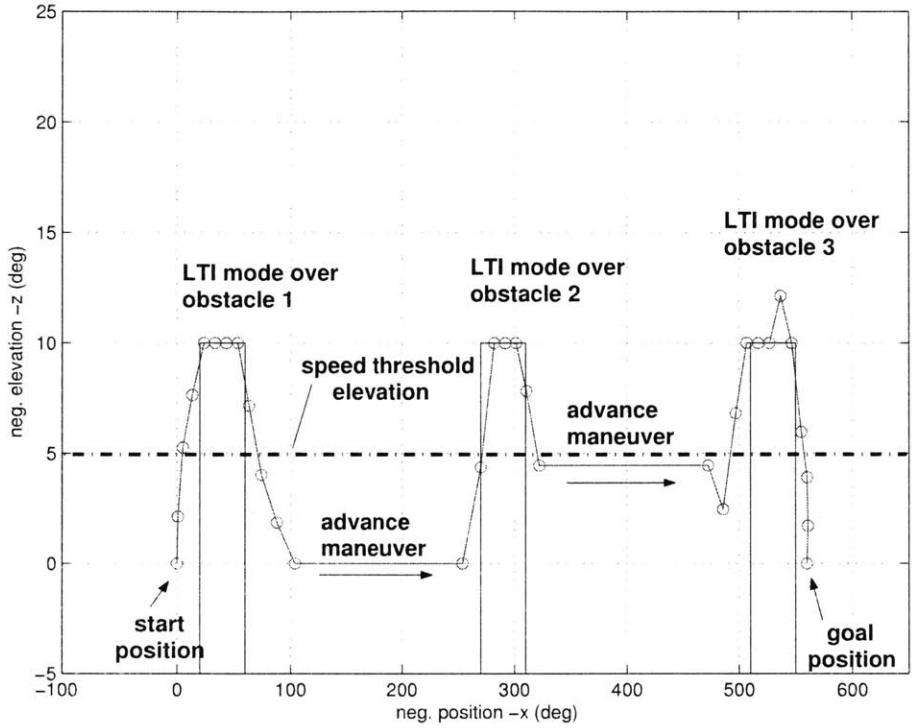


Figure 5-13: MILP planner solution with 10 degree high obstacles and $x_{goal} = -550$ deg.

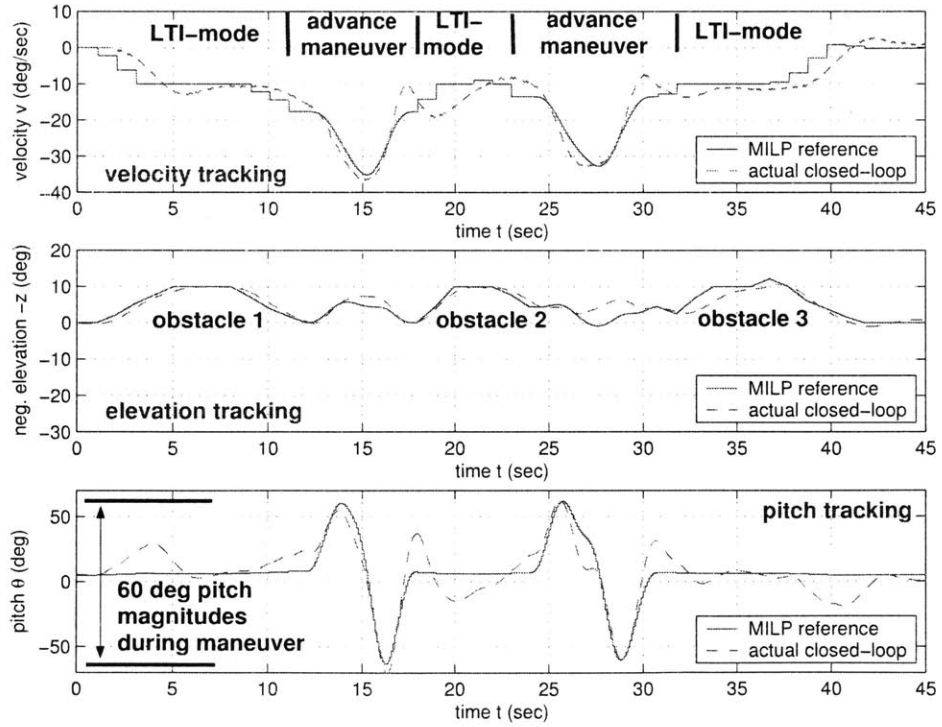


Figure 5-14: Reference and closed-loop tracking of helicopter velocity, elevation, and pitch guidance solutions.

the advance maneuvers. (It is logical that tight tracking performance is most difficult during rapid accelerations as opposed to the slower LTI-modes). However, the overall performance is satisfactory, and the helicopter center of mass effectively stays at or below the visibility line during the maneuver segments.

Figures 5-13 and 5-14 show the MILP solution and the corresponding closed-loop tracking performance for a slightly different scenario, with $x_{goal} = -550$ deg, $\Delta_x = -150$ deg, and substantially lower obstacles (the obstacles in the preceding scenario are near the maximum of the experimental hardware physical flight space). Again the threat zone velocity is bounded by $v_{bound} = 10$ deg/sec, while the MILP guidance solution accelerates the vehicle in the safe region between the first and second obstacles, executing the first advance maneuver at -17.6 deg/sec. As such, the maneuver accelerates the helicopter up to -35 deg/sec using a pitch motion of up to 60 deg in amplitude, which is successfully tracked in closed-loop. Note that pitch flare magnitude is unrestricted in this example since the obstacles are fairly low, increasing the vehicle threat exposure, and making necessary the availability of highly aggressive motions.

If in a given situation the obstacle spacing is highly irregular, it may be desirable to include advance maneuvers of variable travel length (instead of a fixed Δ_x value). To create a fast-advance maneuver class with displacement coupled to initial speed, simply change the last component of the vector constraint in Equation (5.55) to $T \cdot \int_0^1 v(\tau; p) d\tau - f(\sigma_1)$, where f is some desired (differentiable) velocity-to-displacement sensitivity function.

Chapter 6

Conclusions

This thesis motivates, develops, and applies a practical method for parametrizing interesting maneuvers for autonomous vehicles, subject to underlying nonlinear equations of motion and nonlinear constraints on state and control signals. Section 1.4 lists the fundamental contributions of the research; the current chapter summarizes the specific parametrization technique, termed "trajectory interpolation", and draws conclusions about its performance and merits. The chapter then outlines some of the interesting directions for further investigation, listing first the topics that would improve the technical performance of the interpolation method and then considering future applications and theoretical developments.

6.1 Discussion of the Method

A common approach for executing agile autonomous vehicle maneuvers is to apply a feedforward control method. The first step in this approach is to find a feasible reference input-output trajectory satisfying the nonlinear vehicle equations of motion, state and control bounds, and maneuver-defining boundary conditions. The second step is to supply these dynamically feasible references to a tracking controller, which then drives the vehicle with the given inputs while simultaneously regulating errors about the prescribed output signals.

The tracking problem is fairly well understood, given the wide variety of existing linear and nonlinear control methodologies. However, the first problem, frequently referred to as "trajectory generation", is a particularly active research area, given the need to guide agile autonomous vehicles through dynamic environments while satisfying real-world computing limitations. Many modern approaches to vehicle trajectory generation employ efficient specialized representations, often using invertible dynamic models which allow easy computation of feedforward inputs given a candidate output-space trajectory. In many cases, real-time trajectory generation requires embedding these invertible system models into nonlinear programs, casting the output-space system behavior in terms some basis functions, and then attempting to quickly find an optimal trajectory (minimum travel time or output error, for example) that meets motion boundary condition requirements.

Although this method is fundamentally sound from a system dynamics point of view, when considered for real-time applications, it suffers from several classic problems associated with nonlinear optimization. First, nonlinear optimizers are iterative by nature and are not guaranteed to converge to an optimal, or even feasible, solution in real-time. Second, the final trajectory solution typically only corresponds to a local minimum of the objective function and is therefore highly influenced by initial solution guesses. This extreme sensitivity places a large burden on the engineer to provide initial trajectory estimates that both resemble the desired motion yet and dynamically realistic. Third, output space signal basis sets are often overly general, containing many more variables than are needed to describe frequently used motion types. This overparametrization inflates the dimensionality of the nonlinear optimization process, increasing the computation time and preventing iterative algorithms from quickly finding useful solutions. Last, there is no *direct* method of infusing known vehicle behaviors into the optimization process. In many cases, expert human operators can quickly test a vehicle through a human control interface and determine useful operational modes and associated control strategies. However, at present, there are few methods, other than using motion capture samples as initial solution guesses, for incorporating such *a priori* knowledge of vehicle maneuvering characteristics into the nonlinear optimization process.

This thesis directly addresses these drawbacks by combining the aforementioned invertible vehicle models and output-space signal bases into a novel low-dimensional parametric description of the vehicle flight envelope. The method follows from a relaxation of standard nonlinear parametric programming, discarding the objective function while retaining nonlinear equality and inequality constraints. Introduction of a low-dimensional descriptive parameter set into the boundary condition equality constraint vector then creates a highly concise and intuitive description of an entire maneuver class. Further, whereas nonlinear parametric programming traces only solution points that strictly satisfy Karush-Kuhn-Tucker optimality criteria (a high-dimensional process requiring many checks for solution bifurcation and other switching phenomena), the parametrized feasible space approach allows access to the *entire* feasible flight envelope.

While an objective function may be absent from this more general feasible space representation, the engineer can now implicitly specify a maneuvering objective by selecting known feasible motions that define the range and attributes (e.g. agility, control effort, flight path shape) of a given trajectory class. Using techniques similar to the continuation methods of full nonlinear parametric programming, a trajectory interpolation algorithm applies an intuitive feasible projection method and numerical integration process to trace out an entire maneuver class, based on a few user-provided example motions. This procedure completely eliminates the need for on-line iterative optimization and abstracts an entire set of motions into a concise, parametric representation, all while maintaining dynamic feasibility. In this framework, the engineer may choose interesting maneuver instances off-line, exploiting the full resources of nonlinear programming or pilot motion capture while computational resources are abundant. On-line, the trajectory interpolation process then finds feasible vehicle motions maintaining the attributes of the examples but satisfying the

particular boundary conditions dictated by higher-level vehicle motion planners.

Naturally, this method has its limitations, in that the trajectory interpolation process will not discover new strategies for maneuver execution or synthesize feasible maneuver classes from widely varying, dissimilar example motions. However, a key strength of the approach is that *any* method can be used to rigorously find useful example motions while the interpolation process then quickly generalizes them to meet future planning requirements. In addition, the interpolation process is a simple integration procedure, capable of exploring an entire maneuver class in the same time it takes to find a single maneuver instance off-line through nonlinear programming. In on-line practice, trajectory generation follows from a low-dimensional table lookup and integration over a short interval, taking just a few seconds to run, at most.

Overall, the parametrized feasible space and trajectory interpolation method combine the modeling rigor of traditional and modern aerospace techniques with the ability to manipulate and alter sample motions, a key capability in the robotics and computer animation fields. The resulting maneuver classes form something akin to motion primitive elements, often seen in biological motion research. Here, they provide a compact representation of large motion families which can then be combined using a higher-level motion planner to achieve vehicle mission objectives.

The thesis explores several useful attributes of trajectory interpolation, showing that for many useful maneuver classes, the optimality gap induced by discarding the objective function can be directly controlled through user-imposition of additional equality constraints. In some situations, proper selection of the example trajectories leads to zero optimality gap over the maneuver class, so that the integration process produces a set of optimal maneuvers without resorting to iterative methods. In addition, upper and lower bounds on the number of elementary operations for trajectory interpolation illustrate that the process computational load is polynomial in the dimensions of the problem quantities, making it a viable approach for real-time trajectory generation.

Experimental application to a three degree-of-freedom helicopter validates the overall method on a challenging nonlinear system whose equations of motion exhibit the typical nonlinearites encountered in airborne aerobatic helicopter models. Example motions generated from both nonlinear programming and human-piloted motion capture are used to define various example maneuver classes. "Playback" of specific trajectories from these classes under closed-loop control illustrates the combined trajectory generation-feedforward control methodology in practice.

Finally, the thesis incorporates the parametrized trajectories into a mixed integer-linear programming (MILP) motion planning framework. This exercise shows that, with the addition of suitable equality constraints during interpolation, entire dynamically feasible maneuver sets can be easily described by a *single* affine state transformation and binary decision variable. A clearly defined process illustrates how to cast useful vehicle motions into the MILP planning framework, which when combined with existing linear approximations to closed-loop flight control systems, creates a highly flexible guidance framework. Although the MILP planning approach itself relies on on-line optimization, the addition of parametrized maneuver classes immediately generalizes existing hybrid schemes restricted to fixed, discrete equilibrium

and maneuver sets, and can therefore find vehicle guidance solutions with lower cost and over a wider range of initial and final vehicle states.

6.2 Future Directions and Applications

With the experience gained by working with parametrized maneuver classes, applying them to a real nonlinear system, and integrating them into a hybrid system motion planner, it is now useful to list recommendations for continuing work. The first set of ideas pertains to the technical details of creating maneuver classes. The second set deals with future applications and theoretical developments.

Technical improvements

• The trajectory interpolation algorithms follow from a feasible projection of the direct algebraic difference between two feasible trajectory vectors, as seen in Figure 2-4. Intuitively, it seems that the resulting integrated arc takes something near the shortest geometric path between v_1 and v_2 and thus results in a maneuver class with members straying very little from the "style" of the given examples. However, it is an open question of how close this solution arc comes to approximating a true geodesic path [110] in the high-dimensional parameter space. Perhaps more important, it is relevant to ask why choosing such a path leads to maneuver families with similar attributes. Answering these questions will lead to better projection methods and better selection of the example motions.

• Algorithms 1 and 2 of Chapter 2 both interpolate trajectories when no inequality constraints are present. They differ in that the first algorithm uses continuous numerical integration methods while the latter uses large-jump predictor-corrector methods. It is not entirely clear when one is preferable to the other. It would be useful to quantify the relative complexity to the two methods, giving a clear indication over what interval lengths in α it is more efficient to use predictor-corrector instead of integral methods. Further, as yet, there is no predictor-corrector analog of Algorithm 3 that easily handles changes in the active inequality constraint set. Such an algorithm could be highly useful for speeding up the interpolation process as well as developing better methods of detecting changes in the active set J_0.

• Along similar lines, it would useful to employ higher-order feasible arc projections steps in both the integral and predictor-corrector methods [91, 126]. Although such methods would likely involve constraint derivatives of second-order and higher, the corresponding payoff could improve interpolation fidelity and lead to a better understanding of the underlying nonlinear feasible space.

• Constraint derivative computation is the most costly operation in the interpolation process. In many situations it is not necessary, or even practical, to update the derivative matrices at every time step. In this thesis, derivatives are updated over some regular, user-selected interval in α. However, intelligent updating rules exist in the literature [14, 17] and could be applied here to reduce computational load, perhaps even performing low-rank updates of the Jacobian matrices. In addition,

it would be beneficial to study the local curvature of the feasible projection arc for insights into the integration process and feasible space nonlinearity.

• All helicopter maneuver classes in this thesis use the same B-spline output representation with identical knot sequences as given by Equations (4.4) and (4.5). However, these choices can be specialized further based on a particular maneuver type. For instance, if a given maneuver involves significant control or state activity over a particular portion of the trajectory time horizon, it is possible to reduce the knot sequence density in other areas, reducing the dimensionality of p, and thereby improving interpolation speed, as quantified in Section 3.2.

• At present, Algorithm 3 has no provision for preventing a maneuver class from activating more inequality constraints than there are degrees-of-freedom in the trajectory parameter p (given by the dimension of p minus the number of components in $h(p)$). In all of the examples of Chapters 4 and 5, the number of active constraints is fairly small. However, in the optimality gap example of Section 3.1, Algorithm 3 activates as many inequalities as there are available degrees-of-freedom. An important improvement would be to intelligently handle such cases and prevent the projection operation from driving the feasible arc further towards constraint violation, especially for general nonlinear system models. There is likely much to learn on the matter from detailed nonlinear programming methods [14].

• A logical algorithmic contribution would be to directly apply the parametric programming methods of Section 2.2 and references [44, 45, 61, 92] to create a true nonlinear parametric programming method of vehicle trajectories. Such an algorithm would, of course, be concerned only with strictly optimal trajectories but would be useful as a finite parameter space analog of neighboring extremal control [26].

Future applications and theory

• Of clear practical benefit is the application of parametrized maneuver classes beyond the three degree-of-freedom helicopter and into the world of airborne autonomous vehicles. An obvious candidate application is helicopter aerobatics, with a goal of creating flexible agile motions, especially threat evasion, urban warfare, and air-to-air combat maneuvers for military vehicles. Existing work based on analysis of expert human operator control strategies has investigated maneuver parametrizations useful for describing hammerhead, split-S, and aileron roll maneuvers [54, 56]. The mathematical variables in these parametrizations, typically analytical descriptions of fuselage angular rate reference trajectories, are obvious candidates for components of p. In addition, this basis set would involve working not with the bare airframe equations of motion, but instead with a stability-augmented *closed-loop* system model, thus demonstrating maneuver classes on a different system representation.

• Of related interest is the creation of partitioned, or segmented, maneuver classes. These motions would have multiple reduced-dimension parametrizations α_i corresponding to different, parametrized phases of a larger compound motion. For example, such methods could describe a highly complex aerobatic motion, such a helicopter using an articulated hammerhead motion to rapidly turn corners in dense, urban environments. In addition, segmented motions could involve different physical models

over different flight phases, such as a helicopter transitioning from a cruise state to an autorotative descent and then using the autorotation to land at a particular location. Such techniques would also be highly useful for creating parametrized computer animations, adding another capability to the existing methods of Section 1.2.3. In animation applications, systems are often dynamic models of humanoid characters with many degrees-of-freedom or even simpler rigid-body systems experiencing multiple collisions with surrounding objects. By designing coupled, segmented maneuver classes that treat successive phases of larger compound motions, it would be possible to describe entire feasible dynamic simulations with a small set of parameters.

- A fascinating theoretical contribution would be an analysis of general "trajectory interpolability" conditions for useful classes of nonlinear systems. As mentioned in Section 4.5.3, a key assumption underlying the creation of parametrized maneuver classes is that example motions are similar in some manner, reflecting different instances of the same basic motion type but with numerically different boundary conditions. This assumption is critical, allowing the integration process to span the full space "between" the given examples, creating a well defined trajectory class. Discovery of interpolability conditions for arbitrary trajectories would likely involve some sort of topological and/or differential geometric view of the general feasible motion space. A potential payoff includes an analytical method for efficiently partitioning the vehicle flight envelope, allowing creation of a minimal number of generalized maneuver classes.

- A related endeavor that instead treats the flight envelope from an experimental viewpoint is a systematic identification and maneuver-based segmentation of flight data sets. In this scheme, an expert human pilot, or even a stable, randomized control algorithm, explores the vehicle flight envelope unburdened of an overarching mission objective. Afterwards, an automated analysis algorithm examines the recorded flight data and searches for interesting motions that can then define maneuver classes. Similar to the above "interpolability" study, a key outcome is a breakdown of the flight envelope into essential maneuver pieces (or even more basic atomic motion primitives), that can then be used to construct trajectories to meet future motion planning needs. Existing work [37] has explored the basic problem of finding and classifying maneuvering elements by type in a given flight data record. The maneuver class viewpoint of this thesis provides tools for the next phase of the general flight data analysis problem.

- Finally, a clear and practical research area is the continuing integration of maneuver classes into both MILP-based and other hybrid system motion planners. In the domain of MILP-based methods, ongoing work on objective function selection, nonlinear constraint approximation, and efficient methods for solving minimum time problems provides great promise that the combination of LTI-modes and parametrized maneuver classes can become a robust and certifiable method for real-time, on-line motion planning. In the realm of non-MILP planners, integration of continuous maneuver sets into dynamic programming-based methods, such as the maneuver automaton of references [51, 53], can provide a useful method of generating off-line cost maps that enable highly efficient on-line guidance laws and randomized search algorithms.

Appendix A

Helicopter Modeling

The experimental platform for the trajectory generation ideas presented in this thesis is the three degree-of-freedom helicopter, manufactured by Quanser Consulting [122]. This nonlinear system, actuated by a pair of twin rotors, emulates the longitudinal-vertical dynamics of an airborne helicopter and presents a challenging application for continuous maneuver parametrization and real-time control. This appendix describes the main dynamic features of the helicopter, subsequently referred to as the "quanser", and the efforts taken to obtain descriptive linear and nonlinear models.

A.1 System Description

The quanser helicopter is a tabletop-mounted rotorcraft designed for guidance and control systems research. Depicted in Figure A-1, it consists of a twin-rotor-equipped main helicopter body, a counterweight, and three essentially rigid links each of which rotates about a single axis.

The first rotation, called "travel", occurs about a vertical axis perpendicular to the tabletop. The travel angle x gives the rotational position of the elevation arm when viewed from above. This circular degree-of-freedom, with angular velocity $v = \dot{x}$, is measured positive in the clockwise direction when viewed from above. The position origin may be chosen arbitrarily since the system dynamics are essentially invariant with respect to x.

The elevation arm is an angled rod with the helicopter body at one end and a heavy counterweight at the other. The counterweight's position may be adjusted along the arm to modify the helicopter's equilibrium elevation position. The elevation angle is roughly analogous to helicopter altitude and is taken as zero when the helicopter end of the arm is horizontally level. The variable y gives the helicopter's angular elevation, measured positive upward. Equivalently, the variable $z = -y$ gives the elevation measured positive downwards and is similar to the downward translational position coordinate common in helicopter and fixed-wing body frames.

The final degree-of-freedom is the helicopter pitch angle θ, a rotation about the elevation arm. A zero pitch angle occurs when the helicopter body is horizontally aligned, placing the two rotors at equal height above the tabletop. Figure A-1 shows

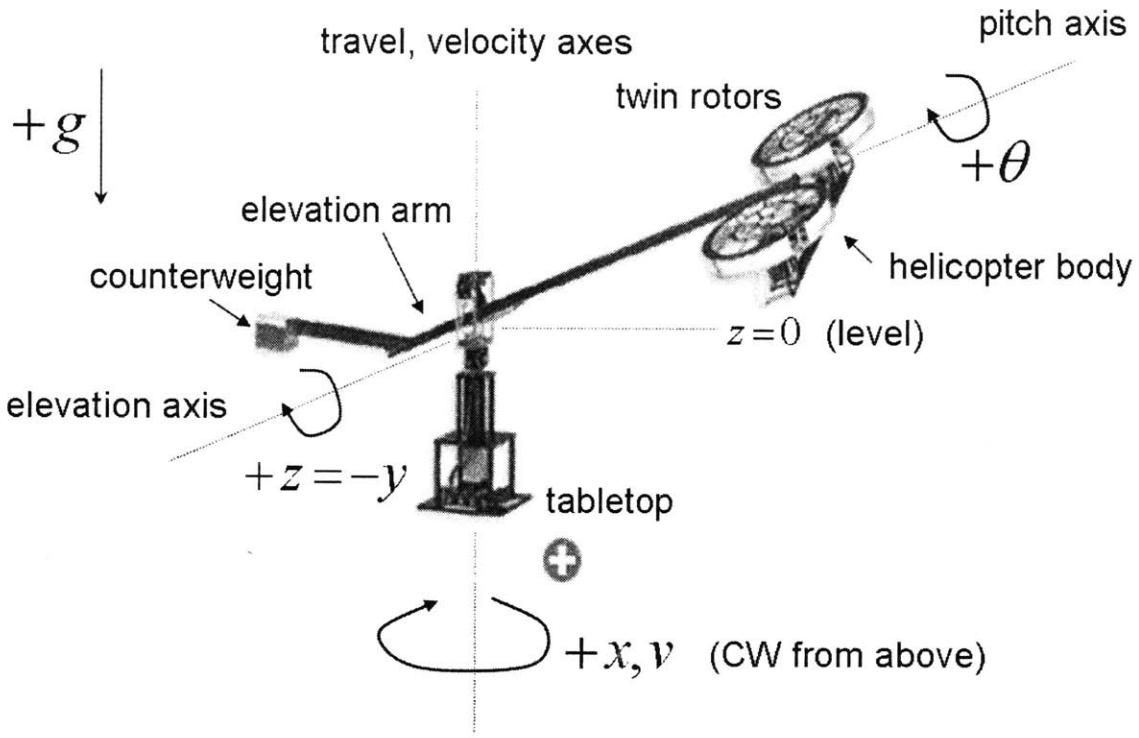


Figure A-1: Three degree-of-freedom helicopter from Quanser Consulting. The instantaneous vehicle configuration is given by the travel angle x, pitch angle θ, and elevation angle z (positive downward), or alternatively, $y = -z$ (positive upward).

the sign convention for θ: positive when the helicopter rotates clockwise when viewed radially inward. The pitching behavior is directly analogous to the longitudinal pitch of a full-scale helicopter.

Each of the twin rotors is a fixed-blade pitch propeller attached to a voltage-driven DC motor. The "forward" motor is at the front the helicopter (to the right when viewed radially outward from the central vertical travel axis) and provides a positive pitching moment. Alternatively, the "rear" motor is at the aft end of the helicopter (to the left when viewed from the travel axis) and provides a negative pitching moment. The two motor voltage channels are organized into a collective voltage V_{coll} and a cyclic voltage V_{cyc}. The collective voltage provides a common voltage signal to the two motors, developing the same rotor speed, and thus generating a thrust that lifts the helicopter without pitching it. The cyclic voltage is a differential voltage signal superimposed on the collective but with opposite signs for each motor, tending to pitch the vehicle. The input channel relations are expressed in equation form as

$$\begin{aligned} V_{front} &= \frac{1}{2}(V_{coll} + V_{cyc}) \\ V_{rear} &= \frac{1}{2}(V_{coll} - V_{cyc}). \end{aligned} \tag{A.1}$$

The input channel terminology mimics that of full-scale helicopters: the collective input governs the amount of lift developed, while the cyclic input is a perturbation

to the collective intended to pitch the vehicle, and thus rotate the thrust vector to achieve translational accelerations. Note that as the quanser system has only three rigid-body degrees-of-freedom (compared to six for a full-scale helicopter), there is only one channel of cyclic input, a reduction from the two-channel longitudinal and lateral cyclics present in standard helicopters.

For experimental operation, the quanser real-time software is driven from a special MATLAB® Simulink® model environment [96] that specifies input collective and cyclic voltages (from user-chosen signal generators, closed-loop control designs, or two-channel pilot joystick inputs), records travel, elevation, and pitch angular positions with data encoders mounted on the rotation arms, and allows the design of automatic feedback and feedforward control systems.

A.2 Linear Modeling and Identification

Mathematical modeling of the quanser helicopter begins with the experimental identification of a linear hover model. This model serves both as a stepping stone to more detailed nonlinear models and as an aid to linear control designs. The vehicle is in a basic hover state when the travel velocity $v = \dot{x}$ is zero and the elevation arm is level and at rest, expressed as $y = 0, \dot{y} = 0$. Note that during the experiments related to this research, the counterweight sits at a radial position such that the hover state is obtained by the steady input voltage pair $(V_{coll}, V_{cyc}) = (1.64\text{ V}, 0\text{ V})$. In the hover state, steady pitch angle is approximately 4.7 degrees. This slight offset from zero is necessary since the combined angular momentum of the twin-rotors (which spin in the same direction) creates a small travel moment on the overall system. If the pitch angle were exactly zero, the helicopter would precess with a small nonzero travel velocity.

About the steady hover condition, one may identify the local system linear dynamics, where the inputs are perturbations from trim equilibrium voltage inputs and the system states are perturbations from their corresponding steady values. For the inputs, define $\delta V_{coll} = V_{coll} - 1.64$ and $\delta V_{cyc} = V_{cyc} - 0$. For the system states, which are taken as the primary three degrees-of-freedom v, y, and θ (and their time derivatives), the linear perturbations are $\delta v = v - 0$, $\delta y = y - 0$, and $\delta\theta = \theta - 4.7^o$. Throughout this section, the "δ" prefixes are dropped since all discussion is understood be in terms of the linear model.

A combination of lumped-parameter modeling, experimental observation, and ad-hoc experimentation reveals that around hover, the system follows three primary dynamic modes. First, the helicopter elevation acts as a lightly damped oscillator with a period of around ten seconds. Perturbations in the y direction lasted for several cycles while slowly decaying in amplitude. Second, the pitching motion also behaves as a lightly-damped oscillator with a period of around five seconds. Lastly, when the helicopter pitches, the thrust vector tilts, resulting in a small component that accelerates the vehicle. This travel component of thrust is proportional to $sin(\theta)$, which is approximately θ for small pitch angles. Recalling that DC motor angular rates typically behave as first-order systems in response to voltage inputs [108], some

first-order actuation behavior is expected in the collective and cyclic input channels. Taking in all of these considerations, the linear system equations of motion about hover are as follows:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -a_1 v - a_2 \theta \\ \ddot{\theta} &= -b_1 \dot{\theta} - b_2 \theta + b_3 \tau_{cyc} \\ \dot{\tau}_{cyc} &= -c_1 \tau_{cyc} + c_2 V_{cyc} \\ \ddot{y} &= -d_1 \dot{y} - d_2 y + d_3 \tau_{coll} \\ \dot{\tau}_{coll} &= -e_1 \tau_{coll} + e_2 V_{coll}. \end{aligned} \tag{A.2}$$

The first two lines of Equations (A.2) express the travel dynamics as a damped double integrator angular system with driving torque proportional to the horizontal thrust component, which is itself proportional to θ. The coefficient a_1 represents the helicopter travel damping and the a_2 coefficient has a negative sign as a consequence of the sign conventions for v and θ (see Figure A-1).

The third and fourth lines of Equations (A.2) give the pitch dynamics as a standard second-order system with input torque τ_{cyc}, governed by the aforementioned first-order actuation dynamics in response to V_{cyc}. The last two lines provide the same general dynamics for the elevation motion: a second-order mechanical system with input torque τ_{coll} and first-order actuation dynamics between V_{coll} and τ_{coll}. Note that a key feature of Equations (A.2) is the separation of the dynamics into two channels: the first being the combined pitch-travel dynamics governed solely by V_{cyc}, and second being the elevation motion governed solely by V_{coll}. This decoupling is convenient and intuitive for operation near the hover state but will be an unreasonable model for more general nonlinear behavior.

With the form of Equations (A.2) in place, the task shifts to finding the numerical coefficient values. To this end, the well-established rotorcraft industry system identification package CIFER® provides the desired tools. CIFER® (Comprehensive Identification from FrEquency Responses) is a two-phase linear identification package used for model identification of full-scale helicopters [145, 146]. First, sample system responses to increasing-frequency chirp inputs are time-frequency windowed and transformed into nonparametric frequency response curves. Second, the software fits user-provided parametric linear models (in either state-space or transfer function form) to the frequency responses using nonlinear optimization methods.

Given the decoupling of linear dynamics, the quanser's elevation channel is identifiable separately from the pitch and travel behavior. The damping and period of the elevation motion were estimated from simple initial condition time responses, providing an estimate of the frequency-domain response range of the collective-to-elevation transfer function:

$$\frac{y(s)}{V_{coll}(s)} = \frac{d_3}{s^2 + d_1 s + d_2} \cdot \frac{e_2}{s + e_1}. \tag{A.3}$$

This transfer function follows directly from the last two lines of Equations (A.2). From there, sets of increasing-frequency chirp inputs (both joystick-driven and signal-

generated) over the expected response frequency range produced the magnitude and phase profiles seen as the solid line plots in Figures A-2 and A-3. The experimental data verifies the lightly damped second-order characteristics.

Once a set of satisfactory response data (wide frequency range, high coherence) were in hand, the linear model in Equation (A.3) provided the template for a parametric data fit. The resulting nonlinear optimization-based match produced the numerical values tabulated in the last five rows of Table A.1. These coefficients apply to Equations (A.2) with y in units of radians, τ_{coll} in units of Newton-meters, and V_{coll} in Volts. The coefficient e_2 was manually set to 1 during fitting, since Equation (A.3) shows that e_2 cannot be identified independently of coefficient d_3. It follows from this data that the elevation damping ratio is 0.0789, corresponding to a lightly damped system, and the oscillation period is 8.85 seconds, in agreement with casual observation. The first-order actuation pole is at $e_1 = 6.16$, giving a time constant of about 0.16 seconds, indicating a relatively fast thrust response. Careful examination of the highest decade in Figure A-2 reveals the presence of this pole.

Note that Table A.1 also gives statistical measures of confidence in the model parameter estimates. The Cramer-Rao (CR) bound gives the minimum possible variance in estimating a parameter taken over all possible unbiased estimators [111]. The table reports the CR-bounds as standard deviations (square root of variance) and as percentages of the parameter estimates. The rightmost column gives the insensitivity of the estimation cost function in terms of single parameter variations, taking into account any correlations with other parameters [145]. A low insensitivity means that the cost function is in fact sensitive to the parameter estimate, indicating that even small parameter variations will cause large degradations in fitting accuracy.

As previously mentioned, the decoupling of linear system dynamics allowed the identification of the pitch-travel modes separate from the elevation behavior. In fact, the second-order pitch dynamics were identifiable in exact analogy to the elevation dynamics, while the pitch-to-travel first-order dynamics (second line of Equation (A.2)) came last. Following the same technique as above, the second-order pitch frequency response range was evaluated from rough estimates of the damping ratio and period, then multiple pilot-flown and signal generated sweeps in V_{cyc} used to drive the system and collect time-domain response data of θ.

Figures A-4 and A-5 show the resulting frequency-domain-transformed response magnitude and phase, respectively, as solid line plots. Once again, the second-order oscillator behavior with first-order actuation was verified and then numerically fit with the transfer function

$$\frac{\theta(s)}{V_{cyc}(s)} = \frac{b_3}{s^2 + b_1 s + b_2} \cdot \frac{c_2}{s + c_1}. \tag{A.4}$$

Finally, the remaining first-order pitch-to-travel transfer function was identified by collecting time-domain response data with V_{cyc} as input and v as output. The low-frequency content of the input signals had to be significantly increased since the travel damping coefficient was known to be quite low (corresponding to a long time constant). Figures A-6 and A-7 give the empirical frequency response of velocity

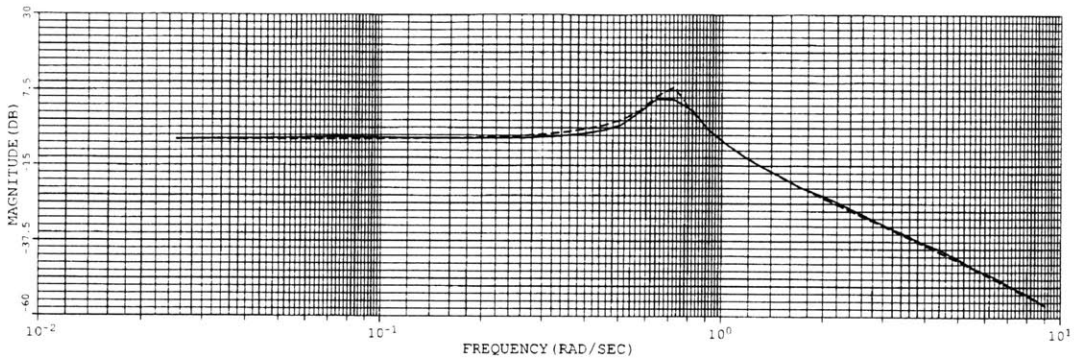


Figure A-2: Collective-to-elevation magnitude frequency response. Solid line is experimental response shape; dashed line is identification fit.

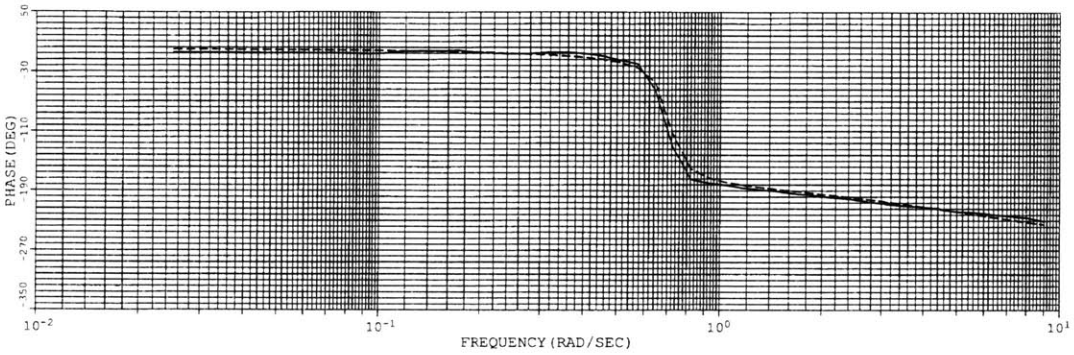


Figure A-3: Collective-to-elevation phase frequency response. Solid line is experimental response shape; dashed line is identification fit.

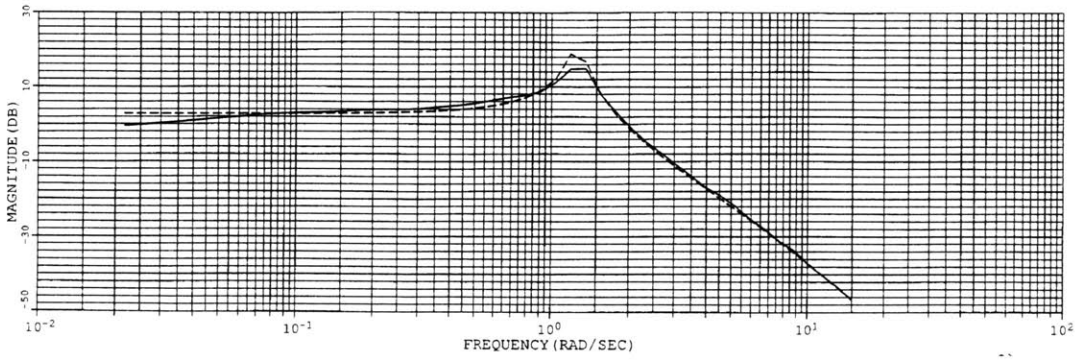


Figure A-4: Cyclic-to-pitch magnitude frequency response. Solid line is experimental response shape; dashed line is identification fit.

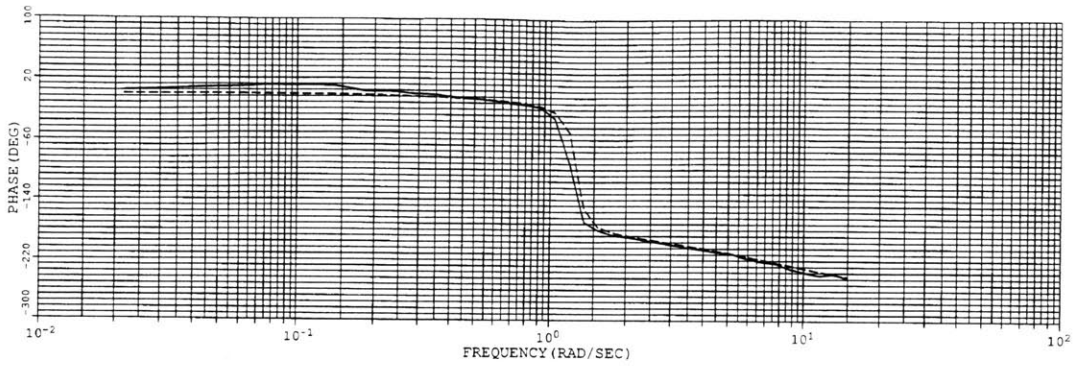


Figure A-5: Cyclic-to-pitch phase frequency response. Solid line is experimental response shape; dashed line is identification fit.

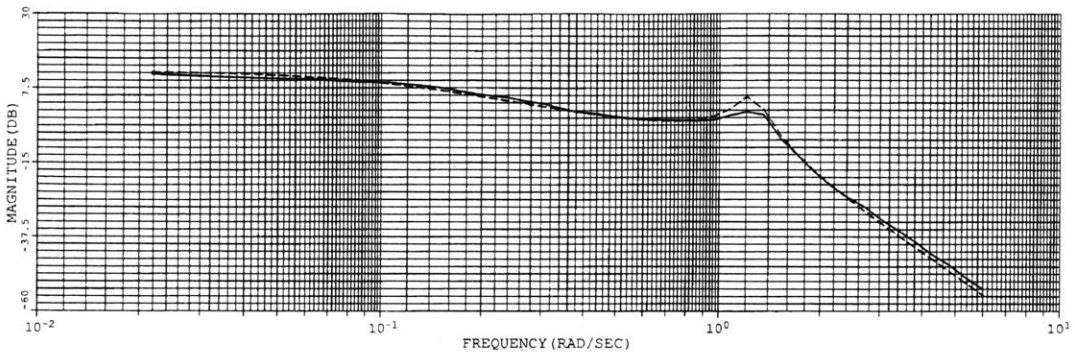


Figure A-6: Cyclic-to-velocity magnitude frequency response. Solid line is experimental response shape; dashed line is identification fit.

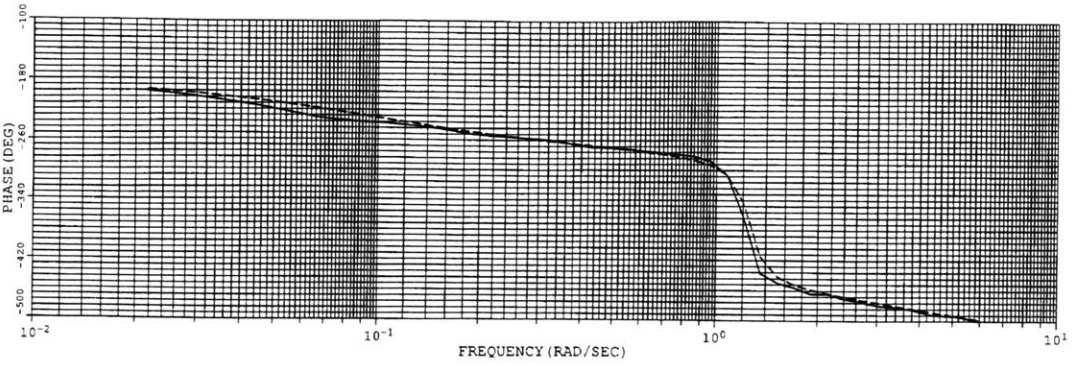


Figure A-7: Cyclic-to-velocity phase frequency response. Solid line is experimental response shape; dashed line is identification fit.

Parameter	Estimate	CR Bound	CR%	Insensitivity%
a_1	0.0839	0.00828	9.87	4.03
a_2	0.257	0.0164	6.41	2.06
b_1	0.163	0.0504	30.9	14.8
b_2	1.58	0.0528	3.35	1.24
b_3	16.2	1.98	12.2	1.43
c_1	7.32	0.993	13.6	1.48
c_2	1†	-	-	-
d_1	0.112	0.0300	26.8	12.9
d_2	0.504	0.0188	3.72	1.61
d_3	1.34	0.242	18.1	2.03
e_1	6.16	1.20	19.4	2.11
e_2	1†	-	-	-

Table A.1: Identified parameters for the three degree-of-freedom helicopter linear model. †These values set identically to 1.

to frequency sweeps in V_{cyc}. Note that the pitch dynamics are necessarily included in these response shapes meaning that all the coefficients from the first six lines of Equation (A.2) are identifiable simultaneously.

To this end, the pitch transfer function of Equation (A.4) was cascaded with the pitch-to-travel transfer function

$$\frac{v(s)}{\theta(s)} = \frac{-a_2}{s + a_1} \tag{A.5}$$

and numerically fit to the responses of Figures A-6 and A-7. The numerical values produced by a preliminary independent cyclic-to-pitch fitting gave the initial estimates for the coefficients b_1, b_2, b_3, c_1, c_2. Note that again, c_2 was set equal to 1 since it was not identifiable independent of b_3. The first seven rows of Table A.1 give the resulting final coefficient values for the pitch-travel dynamics. Similar to the elevation channel, θ is in radians, v is in radians/sec, τ_{cyc} is in Newton-meters, and V_{cyc} is in Volts.

Note that the pitch damping ratio is 0.0648, indicating a lightly damped mode and the oscillation period is 4.99 seconds, in agreement with casual observation. The velocity damping coefficient corresponds to a (comparatively) longer time constant of around 11.2 seconds.

The model form of Equations (A.2) and the experimentally identified coefficient set of Table A.1, provide a working system model around hover. This model is useful of system analysis and control design as well as providing a point-of-departure for the nonlinear modeling necessary for aggressive trajectory generation.

A.3 Nonlinear Modeling and Identification

With a baseline linear system model in place, one may now modify and augment it to obtain a more widely descriptive nonlinear model. As will be shown in the following subsections, the dominant nonlinear effects in the helicopter dynamics come from rotor thrust generation, thrust vector-tilt kinematics, static equilibria offsets, and aerodynamic lift and damping. Once the vehicle model contains the appropriate mathematical expressions for these effects, realistic aggressive trajectory generation is possible.

A.3.1 Actuation and Elevation Pendulum Effects

Given that the earlier system model of Equations (A.2) applies only to flight around the steady hover condition, it is reasonable to assume linear actuation terms. Here, "actuation" refers to the relationship between the input voltage signals, V_{coll} and V_{cyc}, and the rotor thrust forces and torques that drive the system mechanically. For operating conditions away from hover, it is necessary to account for nonlinearities associated with the combined DC motor-propeller rotor design. In addition, it is generally preferable to work with "absolute", as opposed to the perturbed "δ" inputs required for linear models.

To begin, note that the linear actuator model poles correspond to very short time constants compared to the overall system dynamics. The collective actuation pole in Table A.1 is $e_1 = 6.16$ (time constant of 0.16 seconds) and the cyclic pole is $c_1 = 7.32$ (time constant of 0.14 seconds). Since these time scales are small, the dynamics of thrust generation may be neglected for the nonlinear model, and the thrust values may be approximated by their steady state values. However, the nonlinear relationship between input voltage and thrust magnitude *is* important.

Recall that at steady-state, a voltage-driven DC motor's shaft speed is proportional to its input voltage [108], so that the two quanser motors obey

$$\begin{aligned} \Omega_{front} &\propto V_{front} \\ \Omega_{rear} &\propto V_{rear} \end{aligned} \tag{A.6}$$

where Ω denotes the motor shaft angular speed. However, propellers generally exhibit a quadratic relationship between speed and thrust [88], giving rise to the relations

$$\begin{aligned} T_{front} &\propto \Omega_{front}|\Omega_{front}| \\ &\propto V_{front}|V_{front}| \\ T_{rear} &\propto \Omega_{rear}|\Omega_{rear}| \\ &\propto V_{rear}|V_{rear}|, \end{aligned} \tag{A.7}$$

where the T's are the rotor assembly thrusts, taken as positive if upward and negative if downward in Figure A-1. The absolute values in this equation provide an upward thrust vector for positive voltages and downward vector for negative voltages.

Now, assuming a pitch angle $\theta = 0$, the collective torque tending to rotate the

quanser about its elevation axis is proportional to the sum of the two rotor thrusts. Recall that the helicopter is in a hover equilibrium state at $(V_{coll}, V_{cyc}) = (1.64\text{ V}, 0\text{ V})$. The useful input range of the vehicle is generally within the bounds $0.9\text{ V} \leq V_{coll} \leq 2.0\text{ V}$ and $-0.6\text{ V} \leq V_{cyc} \leq 0.6\text{ V}$ so that the following simplification of collective torque is possible *over these voltage ranges*:

$$\begin{aligned} \tau_{coll} &\propto T_{front} + T_{rear} \\ &\propto (V_{coll} + V_{cyc})|V_{coll} + V_{cyc}| + (V_{coll} - V_{cyc})|V_{coll} - V_{cyc}| \\ &= (V_{coll} + V_{cyc})^2 + (V_{coll} - V_{cyc})^2 \\ &\propto V_{coll}^2 + V_{cyc}^2 \\ &\approx V_{coll}^2. \end{aligned} \tag{A.8}$$

The last approximation is reasonable since V_{coll} is usually larger than V_{cyc}, especially for trajectories of interest in this thesis. This input simplification also helps avoid tricky solution logic when inverting the final vehicle model yet still accurately represents the dominant quadratic nonlinearity in thrust generation. This quadratic relationship between thrust magnitude and collective voltage can also be seen in Quanser product literature plots [122].

A similar set of steps gives an expression for the cyclic torque over the above input voltage ranges:

$$\begin{aligned} \tau_{cyc} &\propto T_{front} - T_{rear} \\ &\propto (V_{coll} + V_{cyc})|V_{coll} + V_{cyc}| - (V_{coll} - V_{cyc})|V_{coll} - V_{cyc}| \\ &= (V_{coll} + V_{cyc})^2 - (V_{coll} - V_{cyc})^2 \\ &\propto V_{coll} V_{cyc}. \end{aligned} \tag{A.9}$$

In this case, there is no reasonable way to separate V_{coll} and V_{cyc}, so both of these input channels will be present in pitch motion channel.

The quadratic collective actuation term of Equation (A.8) can be verified and the linear restoring term of Equation (A.2, line 5) generalized simultaneously by performing a static measurement test. While maintaining a near zero pitch angle with $V_{cyc} = 0$, the collective voltage V_{coll} was set to various steady values between 0 and 2.2 Volts. Measurement of the steady-state elevation angle (circle-line plot in Figure A-8) shows a nonlinear relationship between voltage and deflection. Note that in the sequel, the elevation angle is coordinatized as $z = -y$ (that is, z positive *downwards*) to be more consistent with standard rotorcraft body frame notation. The quadratic collective-to-thrust relationship is not sufficient to fit the data set of the figure, mainly since the linear restoring force for the hover model is not valid over large angle excursions. Therefore, generalize the restoring force from a linear term in z to a $-cos(z) + sin(z)$ trigonometric expression. The sine term is common for nonlinear pendulums; the cosine term accounts for the offset counterweight at the far end of the elevation arm. With these modifications, the steady-state elevation behavior follows:

$$0 = d_2 cos(z_{ss}) - d_3 sin(z_{ss}) - d_4 V_{coll,ss}^2, \tag{A.10}$$

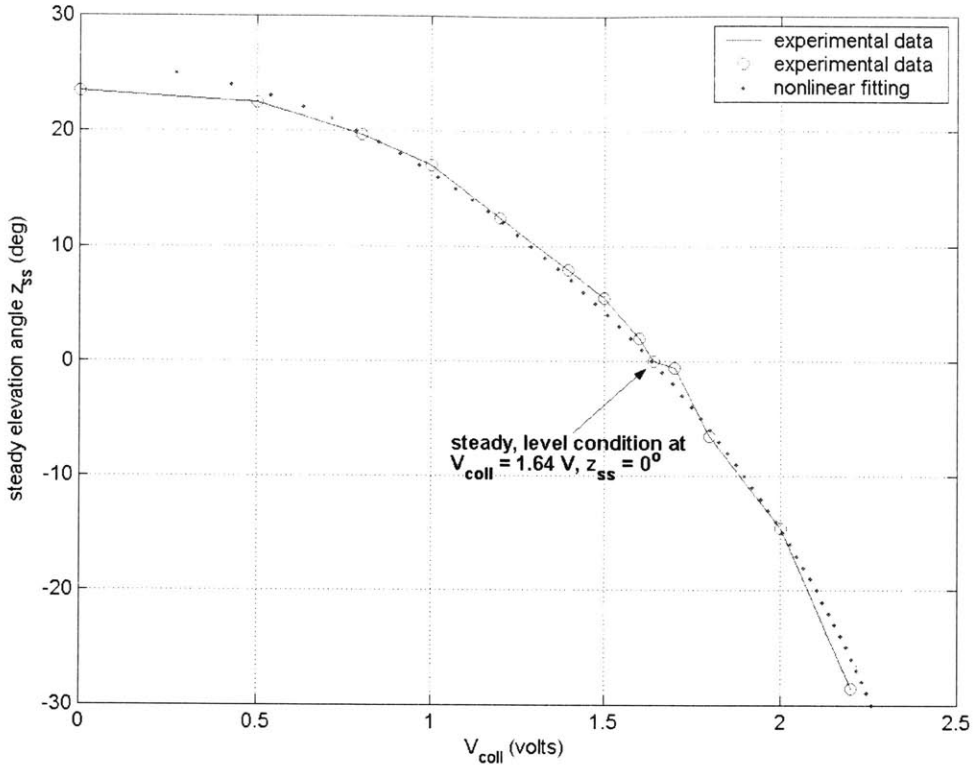


Figure A-8: Fit of steady collective-elevation coefficients from experimental data.

where z_{ss} denotes the steady angle and $V_{coll,ss}$ denotes the steady collective input. A nonlinear regression fit for the coefficients (d_2, d_3, d_4) gave the numerical values listed in Table A.2. Reinsertion of the acceleration and velocity damping terms gives a (temporary) equation of motion:

$$\ddot{z} = -d_1\dot{z} + d_2 cos(z) - d_3 sin(z) - d_4 V_{coll}^2. \tag{A.11}$$

As seen in the table, the d_1 coefficient value remains the same since the voltage-deflection test did not measure any dynamic effects.

A.3.2 Thrust Vector Tilting

Similar to the trigonometric nonlinearities added to the elevation equation, sine and cosine terms are necessary to account for thrust vector-tilting. When the helicopter pitch angle assumes large values, the thrust vector resulting from the twin rotors rotates, contributing a larger component to travel acceleration and smaller component to elevation.

Recall that line 2 of Equation (A.2) assumes a constant hover thrust and small pitch angles θ. Modification for nonlinear actuation leads to

$$\dot{v} = -a_1 v - a_2 V_{coll}^2 \theta \tag{A.12}$$

with the following alteration for large pitch angles:

$$\dot{v} = -a_1 v - a_2 V_{coll}^{'2} sin(\theta). \tag{A.13}$$

Estimation of a_1 and a_2 for the nonlinear model follows shortly.

Additionally, Equation (A.11) requires a cosine factor to account for the reduced lift effect, bringing that equation to its final form:

$$\ddot{z} = -d_1 \dot{z} + d_2 cos(z) - d_3 sin(z) - d_4 V_{coll}^{'2} cos(\theta). \tag{A.14}$$

A.3.3 Static Offsets

Recalling that the nonlinear model is in terms of absolute coordinates, whereas the linear model uses perturbed "δ" variables, it is possible to explicitly include the small precessing effect from the twin rotor's combined spin rate. The θ variable is offset by a small $\theta_a = 4.74^o = 0.0827$ rad constant:

$$\dot{v} = -a_1 v - a_2 V_{coll}^{'2} sin(\theta - \theta_a) \tag{A.15}$$

Further, the helicopter itself is slightly imbalanced, with a static offset of around 5 degrees when no inputs are applied. Note that this imbalance is essentially identical to the above rotor spin offset. It is therefore necessary to add a small constant b_0 to the pitch equation of motion:

$$\ddot{\theta} = -b_1 \dot{\theta} - b_2 sin(\theta) + b_0 + b_4 V_{coll} V_{cyc}. \tag{A.16}$$

The nonlinear cyclic actuation effect replaces the previously linearized simple τ_{cyc} term of Equation (A.2, line 3). Reidentification of the b_2 and b_4 coefficients follows in the next subsection.

A.3.4 Aero Damping and Lift

The final nonlinear effects to include involve aerodynamics of forward and reverse flight. Taken together, Equations (A.15) and (A.16), comprise a complete model of the pitch-travel dynamics. By considering a sequence of steady (V_{coll}, V_{cyc}) commands, a comparison can be made between the predicted and observed steady-state velocity and pitch angles. Casting Equations (A.15) and (A.16) into a regression format, it is possible, assuming a value for a_1, to estimate the coefficient set (a_2, b_2, b_0, b_4) with linear regression. For various data sets, the predicted pitch angle was consistently too large for a given speed, indicating the presence of a speed-damping term in the θ dynamics. By hypothesizing and then testing various forms of the damping expression, it was possible to identify an effect of the form $v|v|$ and then introduce it into the pitch equation of motion as follows:

$$\ddot{\theta} = -b_1 \dot{\theta} - b_2 sin(\theta) + b_0 + b_3 v|v| + b_4 V_{coll} V_{cyc}. \tag{A.17}$$

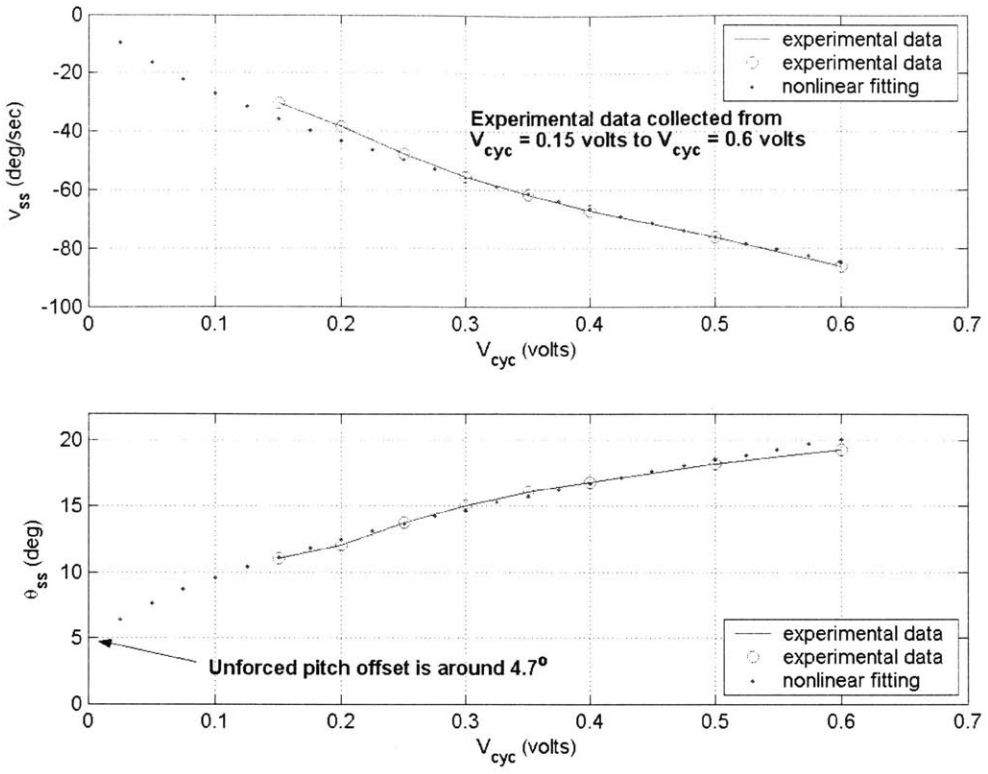


Figure A-9: Fit of steady cyclic-pitch-travel coefficients from experimental data.

Once the form of the term was finalized, a linear regression analysis gave estimates of an expanded coefficient set $(a_2, b_2, b_0, b_3, b_4)$. Figure A-9 shows the corresponding steady-state velocity and pitch angle fits for the negative velocity travel direction (corresponding to $V_{cyc,ss} > 0$). Table A.2 gives the resulting coefficient estimates. Due to asymmetries in the velocity dynamics, possibly related to the static pitch offset, the resulting fit did a poor job of describing the pitch angle in the positive velocity direction (where $V_{cyc,ss} < 0$). Therefore the form of the equations was maintained but the coefficients resolved using positive-velocity data only. In addition, a combined-direction identification fit produced a coefficient set with reasonable fit for both travel directions, although with slightly more steady-state error than the highly-tuned one-sided cases. Table A.2 gives the numerical coefficients for each case. Note that only a_2 and b_3 need to be modified in each setting, since they account for the pitch damping effect more than any other terms. The final value of a_1 was adjusted manually to give a good fit of the dynamic velocity response to a staircase-like V_{cyc} input profile. See Figure A-10 for the time domain verification. Since b_1 cannot be identified from the steady-state regression, the previous linear identification value was assumed.

Finally, a translational lifting effect was observed during steady forward and reverse flight. At a given speed, the steady elevation could be as much as 5 degrees higher the predicted value. A v^2 term in the equations of motion provides a bidirectional lift:

$$\ddot{z} = -d_1\dot{z} + d_2 cos(z) - d_3 sin(z) - d_5 v^2 - d_4 V_{coll}^2 cos(\theta). \tag{A.18}$$

Note that other nonlinear forms such as $|v|$ or $\sqrt{|v|}$ would give a better fit to

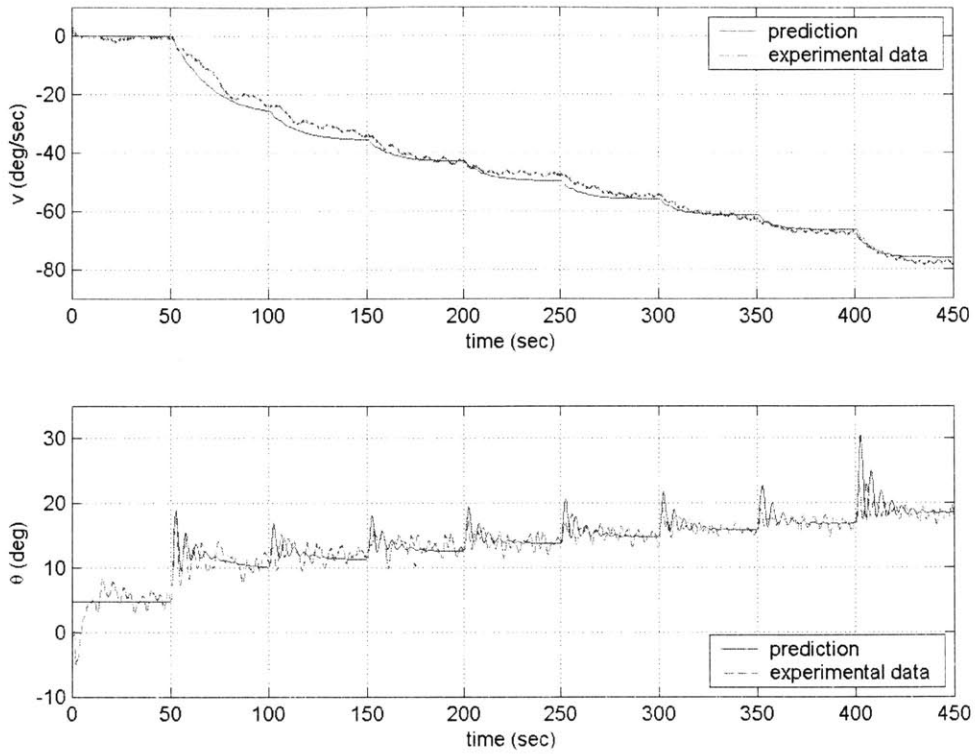


Figure A-10: Time domain validation of velocity and pitch nonlinear system responses. Input is an increasing staircase of V_{cyc} values from 0.05 V to 0.6 V.

the observed lifting effect, especially since it tends to level off with increasing speed. However, for the purposes of retaining a differentiable set of system equations, the v^2 term can be suitably tuned over the speed range of interest. An alternative is to simply add a constant lift terms, corresponding to a mean offset slightly less than 5 degrees, although it would overpredict the steady elevation near the hover condition.

A.3.5 Equations of Motion

Combination of all the nonlinear effects into one set of equations gives:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -a_1 v - a_2 V_{coll}^2 sin(\theta - \theta_a) \\ \ddot{\theta} &= -b_1 \dot{\theta} - b_2 sin(\theta) + b_0 + b_3 v|v| + b_4 V_{coll} V_{cyc} \\ \ddot{z} &= -d_1 \dot{z} + d_2 cos(z) - d_3 sin(z) - d_5 v^2 - d_4 V_{coll}^2 cos(\theta) \end{aligned} \tag{A.19}$$

These equations of motion, with numerical coefficients in Table A.2, represent a general flight model of the three degree-of-freedom helicopter, and are the basis for nonlinear trajectory generation. Position x is in radians, velocity v is in rad/sec, pitch angle θ and elevation angle z are in radians. Collective and cyclic inputs are in Volts. The system model has two desirable properties. First, given time histories of $v(t)$, $\theta(t)$, and $z(t)$, it is possible to invert the model and explicitly solve for $V_{coll}(t)$ and $V_{cyc}(t)$. Second, the system model is differentiable in terms of its states, making

Parameter	$v < 0$ Estimate	$v > 0$ Estimate	General Estimate
a_1	0.0252	same	same
a_2	0.0525	0.0408	0.0439
θ_a	0.0827	same	same
b_0	0.131	same	same
b_1	0.163	same	same
b_2	1.58	same	same
b_3	0.449	0.188	0.259
b_4	1.42	same	same
d_1	0.112	same	same
d_2	0.243	same	same
d_3	0.504	same	same
d_4	0.0905	same	same
d_5	0.0400	same	same

Table A.2: Estimated parameters for the three degree-of-freedom helicopter nonlinear model. Due to system asymmetries, model is broken into two operating regions: $v < 0$ travel and $v > 0$ travel. Parameters a_2 and b_3 can be specifically tuned for either case or set to a general bidirectional value. Units correspond to a system with angles in radians and inputs in Volts.

it amenable to trajectory interpolation.

The model of Equations (A.19) is reasonably complete for the purposes of this thesis, however there remain several unmodeled nonlinear effects which could become relevant in other contexts. Among them are: first-order rotor inflow dynamics, variable rotational inertia of the helicopter depending on elevation angle, ground effect of the rotors near the tabletop, vibrational modes of the long, slender elevation arm, and a variable restoring moment on the pitch angle as a function of z. An early attempt at nonlinear modeling through a complete Lagrangian formulation quickly produced many complex high-order terms, and was used for insight purposes only.

Note that there are alternative methods of nonlinear identification, providing a more systematic approaches method than the ad-hoc vehicle-specific procedure given here. Examples include nonlinear black box, regression, and fuzzy techniques [90] as well as Volterra series [39].

Appendix B

Splines in B-form

This thesis expresses system output and state signals in terms of B-spline bases, since linear spline combinations form a very natural trajectory parametrization. B-splines are by no means the only option for casting time domain signals in a compact form. Other basis options include polynomials and piecewise polynomials [63, 68], wavelets [94], exponential functions [41], and even nonlinear subsystem output signals [70]. However, B-splines have certain properties well-suited for the systems of present interest and have been shown quite useful for trajectory generation for numerous nonlinear systems [30, 102, 113]. For example, B-splines combine the generality of polynomials with the ability to capture varying levels of detail over localized regions, similar to multiresolution formats. In addition, higher-order continuity throughout the interval of definition is easy to guarantee, eliminating the need for large numbers of interior continuity conditions that sometimes hinder piecewise polynomial bases. Finally, recursive evaluation of higher-order splines in terms of lower-order ones provides numerical stability and computational speed during evaluation [34], making them attractive in applications as diverse as geometric design, computer graphics, data fitting and storage, and even typography [33]. Reference [34] gives a comprehensive discussion of spline properties and algorithms, as well as tensor spline extensions to multivariate data sets.

The following discussion will be concerned only with univariate splines as applied to time domain signals. The first section will highlight elementary properties and follow notation similar to that of references [34, 35]. The second section examines the signal dependence on basis coefficients as independent variables, deriving explicit formulae for analytic differentiation, a frequently used computation in trajectory interpolation and optimization.

B.1 Basic Construction

Consider a univariate function f with independent variable x expressed in terms of a B-spline basis expansion:

$$f(x) = \sum_{i=1}^{n} a_i B_{i,k}(x). \tag{B.1}$$

Here, f is called a "spline in B-form" while the n basis functions $B_{i,k}$ are the actual B-splines, each of order k. The a_i are combination coefficients and may be adjusted to change the shape of the function f. A B-spline $B_{i,k}$ of order k is actually a locally defined polynomial of order $k-1$. That is, a first-order spline is a locally defined constant, a second-order spline is a locally defined linear function, and so on.

The overall function f is zero everywhere outside of a basic interval $[a, b]$. Essential to the designation of a spline in B-form is the knot sequence

$$S = \{x_1, \ldots, x_m\} \tag{B.2}$$

which is a nondecreasing sequence of m scalars, called "knotpoints", in the interval $[a, b]$ satisfying the properties

$$\begin{aligned} x_1 &= a, \\ x_m &= b > a, \\ x_i &\leq x_{i+1}. \end{aligned} \tag{B.3}$$

Note that knotpoints may be repeated at a general point $x \in [a, b]$ but only a finite number of times *and* satisfying the upper bound

$$\text{mult}(x) \leq k, \tag{B.4}$$

where $\text{mult}(x)$ is the "multiplicity" of knotpoints at x, i.e. the number of repetitions of x in S. The ith individual B-spline is defined only on the interval $[x_i, x_{i+k}]$ and is therefore often denoted by

$$B_{i.k}(x) = B_{i.k}(x \,|\, x_i : x_{i+k}) \tag{B.5}$$

for every $i \in \{1, \ldots, n\}$. Since this interval of definition involves $k+1$ knotpoints, it follows from Equations (B.3) and (B.4) that $x_i < x_{i+k}$, meaning $B_{i,k}$ is supported on a interval of positive measure, that is,

$$\text{supp}\, B_{i,k}(x \,|\, x_i : x_{i+k}) = [x_i, x_{i+k}]. \tag{B.6}$$

Since there are m knotpoints and each of the n B-splines $B_{i,k}$ "covers" $k+1$ of them, it follows that

$$m = n + k. \tag{B.7}$$

A feature of the B-spline format is that two neighboring splines $B_{i,k}$ and $B_{i+1,k}$ overlap, both being supported on the common interval $[x_{i+1}, x_{i+k}]$. Alternatively, only one these pair is supported on each of the intervals $[x_i, x_{i+1}]$ and $[x_{i+k}, x_{i+k+1}]$. In addition, $B_{i,k}$ will overlap $B_{i+2,k}$ on the interval $[x_{i+2}, x_{i+k}]$, and so forth. This property allows the collective splines to richly cover the entire basic interval $[a, b]$ while maintaining linear independence.

Further, since each kth-order spline is actually a $(k-1)$th-order polynomial in its interval of definition, f will be infinitely differentiable away from knotpoints (that is, $f \in C^\infty$ if $x \notin S$). At knotpoints $x \in S$, where neighboring splines overlap, the

multiplicity of x will govern the continuity of f. Specifically,

$$f \in C^{k-\text{mult}(x)-1} \text{ at } x \text{ if } x \in S \tag{B.8}$$

where $f \in C^p$ at x denotes a function with p continuous derivatives at x. Here, $f \in C^0$ at x indicates a function that is merely continuous at x but not differentiable while $f \in C^{-1}$ at x indicates a discontinuous function at x. These continuity conditions allow the user to pick a knot sequence S to meet function approximation needs throughout the basic interval $[a, b]$. For example, if a function f is required to assume a nonzero value at a, then take $\text{mult}(a) = k$. (Recall that f is defined to be 0 outside of $[a, b]$). Similarly, an interior discontinuity at a point $a < x < b$ is accommodated by taking $\text{mult}(x) = k$. Lesser degrees of discontinuity are handled by taking $\text{mult}(x) < k$. The simple choice $\text{mult}(x) = 1$ implies that $f \in C^{k-2}$ at x, since f at that knotpoint is an overlap of two $(k-1)$th-order polynomials.

Of occasional use is the finite *partition of unity* property of B-splines. Recall that a partition of unity (finite or infinite) is a set of functions who sum identically to 1 over some superset that includes the support sets of each [104]. Given a knot sequence S and an order k, setting $a_i = 1$ for every $i \in \{1, \ldots, n\}$ has the consequence that

$$f(x) = 1 \text{ at every } x \in [x_k, x_{m-k}] = [x_k, x_n] \subset [a, b].$$

By picking $\text{mult}(a) = k$ and $\text{mult}(b) = k$, f will obey

$$f(x) \equiv 1 \text{ at every } x \in [a, b]$$

when all $a_i = 1$.

B.2 Differentiation of Parametrized Signals

With the fundamentals of basic construction in place, it is assumed that a knot sequence S and order k have been chosen and that the function f is a *time domain* signal with $x = t$. Such signals expressed in terms of B-spline bases are the fundamental objects for trajectory interpolation in this thesis and are evaluated at times t but parametrized by the coefficient set $\{a_i\}$. As such, f is a function of $n + 1$ total variables:

$$f(t) = f(t; a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i B_{i,k}(t). \tag{B.9}$$

Partial differentiation of f with respect to t follows by differentiating each of the individual B-splines with respect to t. Of course, the derivative of a kth-order spline (a $(k-1)$th-order polynomial) gives a resultant $(k-1)$th-order spline (a $(k-2)$th-order polynomial) as shown by

$$\frac{\partial f}{\partial t} = \sum_{i=1}^{n} a_i \frac{\partial B_{i,k}}{\partial t}(t) = \sum_{i=1}^{n'} a'_i B'_{i,k-1}(t). \tag{B.10}$$

That is, $\partial f/\partial t$ is just another function expressed in terms of a *lower*-order spline basis. As a spline in B-form, $\partial f/\partial t$ must still obey the algebraic relation of Equation (B.7) between the number of knotpoints, number of elements, and spline order. That is,

$$m' = n' + k'$$

but now with

$$k' = k - 1$$

and

$$m' = m - N_k.$$

Here, N_k is number of original knotpoints in S with multiplicity k. Since the $B'_{i,k-1}$ are $(k-1)$th-order splines and the multiplicity constraint of Inequality (B.4) must hold, the knot sequence S' of $\partial f/\partial t$ is the same as S, but with any knotpoints of original multiplicity k reduced to multiplicity $k-1$. It follows from the above relations that $n' = m - N_k - k + 1$.

Considerably less obvious is the relationship between the coefficient sets $\{a'_i\}$ and $\{a_i\}$. Fortunately, reference [34] provides the following recursive difference relationship:

$$a'_i = (k-1)\frac{a_i - a_{i-1}}{x_{i+k-1} - x_i} \tag{B.11}$$

where $a_0, a_{n+1} \equiv 0$. Such easy and numerically stable relationships between bases of differing order are one the primary reasons for the wide popularity of B-splines [33].

Just as important for the purposes of continuous trajectory parametrization is the differentiation of $f(t; a_1, \ldots, a_n)$ with respect to the coefficient set $\{a_i\}$. Since the spline coefficients directly parametrize the motion of a system, it is important to understand the first-order behavior of f in terms of any given a_i. What follows is a simple and progressive discussion of the differentiation of f with respect to any given a_i. The derived analytic relationships eliminate the need for numerical differentiation of trajectories, thus reducing computational burden and improving numerical accuracy.

First, it is easy to see that the partial derivative of f with respect to the ith coefficient a_i evaluated at a point t_0 is equal to the value of the ith B-spline at t_0:

$$\begin{aligned} \frac{\partial f}{\partial a_i}(t_0) &= B_{i,k}(t_0) \\ &= f(t_0; 0, \ldots, 1_i, \ldots, 0) \\ &= f(t_0; e_i) \\ &= f(t; e_i)|_{t=t_0} \end{aligned} \tag{B.12}$$

where e_i is the i-th unit vector in R^n. The latter two lines of Equation (B.12) illustrate that the evaluation of the partial derivative of f at t_0 is nothing other than a simple evaluation of a reduced form of f at that point. The "reduced form" is merely the ith B-spline component of f. Further, letting $f^{(m)}$ denote the m-th *time* derivative

of f. that is. with the definition

$$f^{(m)}(t; a_1, \ldots, a_n) \equiv \frac{\partial^m f}{\partial t^m}(t; a_1, \ldots, a_n), \tag{B.13}$$

then the partial derivative of $f^{(m)}$ with respect some a_i is given by first replacing $f(t; a_1, \ldots, a_n)$ by its ith component $f(t; e_i)$, differentiating with respect to time, and then evaluating at the desired point t_0:

$$\begin{aligned} \frac{\partial f^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) &= \left(\frac{\partial^m}{\partial t^m}(f(t; e_i)) \right)_{t=t_0} \\ &= f^{(m)}(t; e_i)|_{t=t_0}. \end{aligned} \tag{B.14}$$

The $t = t_0$ outside of the time differentiation indicates that the time evaluation comes last. Note that $f^{(m)}$ is indeed an explicit function of the $\{a_i\}$ because of Equation (B.11).

Naturally, linear combinations of spline functions of $\{a_i\}$ obey the usually linearity of differentiation. That is, if f_1 and f_2 are two spline functions of the same parameter set $\{a_i\}$, as in

$$g(t; a_1, \ldots, a_n) = f_1(t; a_1, \ldots, a_n) + f_2(t; a_1, \ldots, a_n), \tag{B.15}$$

then

$$\begin{aligned} \frac{\partial g^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) &= \frac{\partial f_1^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) + \frac{\partial f_2^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) \\ &= f_1^{(m)}(t; e_i)|_{t=t_0} + f_2^{(m)}(t; e_i)|_{t=t_0}. \end{aligned} \tag{B.16}$$

This result follows directly from linearity and Equation (B.14). A simpler case occurs when g is a function of two different parameter sets $\{a_i\}$ and $\{b_i\}$, by virtue of being a linear combination of two *independent* splines, as in

$$g(t; a_1, \ldots, a_n; b_1, \ldots, b_r) = f_1(t; a_1, \ldots, a_n) + f_2(t; b_1, \ldots, b_r). \tag{B.17}$$

Then the following relations hold:

$$\begin{aligned} \frac{\partial g^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n; b_1, \ldots, b_r) &= \frac{\partial f_1^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) \\ &= f_1^{(m)}(t; e_i)|_{t=t_0} \end{aligned} \tag{B.18}$$

and

$$\begin{aligned} \frac{\partial g^{(m)}}{\partial b_j}(t_0; a_1, \ldots, a_n; b_1, \ldots, b_r) &= \frac{\partial f_2^{(m)}}{\partial b_j}(t_0; b_1, \ldots, b_r) \\ &= f_2^{(m)}(t; e_j)|_{t=t_0}. \end{aligned} \tag{B.19}$$

In situations where $g(t)$ is actually some nonlinear function h of the values of the mth *time* derivative of a spline f, then g is explicitly a function of $\{a_i\}$ as well:

$$g(t; a_1, \ldots, a_n) = h(f^{(m)}(t; a_1, \ldots, a_n)). \tag{B.20}$$

In such cases, the partials of g with respect to a given a_i follow from an application of the standard multivariable chain rule [104]:

$$\begin{aligned} \frac{\partial g}{\partial a_i}(t_0; a_1, \ldots, a_n) &= \frac{\partial h}{\partial f^{(m)}}(f^{(m)}(t_0; a_1, \ldots, a_n)) \cdot \frac{\partial f^{(m)}}{\partial a_i}(t_0; a_1, \ldots, a_n) \\ &= \frac{\partial h}{\partial f^{(m)}}(f^{(m)}(t_0; a_1, \ldots, a_n)) \cdot f^{(m)}(t; e_i)|_{t=t_0}. \end{aligned} \tag{B.21}$$

It is important to note that the evaluation of $\partial h/\partial f^{(m)}$ involves the full spline $f^{(m)}$ while the evaluation of $\partial f^{(m)}/\partial a_i$ involves only its ith spline component.

Finally, expressions may arise where g depends on the product of two related spline-dependent functions, each similar in form to that of Equation (B.20). Partial differentiation then follows the usual product rule, given the chain rule of Equation (B.21), so that

$$g(t; a_1, \ldots, a_n) = g_1(f_1^{(m_1)}(t; a_1, \ldots, a_n)) \cdot g_2(f_2^{(m_2)}(t; a_1, \ldots, a_n)) \tag{B.22}$$

has partial derivatives

$$\begin{aligned} \frac{\partial g}{\partial a_i}(t_0; a) &= g_1(f_1^{(m_1)}(t_0; a)) \cdot \frac{\partial g_2}{\partial f_2^{(m_2)}}(f_2^{(m_2)}(t_0; a)) \cdot f_2^{(m_2)}(t; e_i)|_{t=t_0} + \\ &\quad \frac{\partial g_1}{\partial f_1^{(m_1)}}(f_1^{(m_1)}(t_0; a)) \cdot f_1^{(m_1)}(t; e_i)|_{t=t_0} \cdot g_2(f_2^{(m_2)}(t_0; a)), \end{aligned} \tag{B.23}$$

where a denotes the general n-tuple set $\{a_1, \ldots, a_n\}$. The case where f_1 and f_2 are splines dependent on *different* parametrizations a and b is easier (with b denoting a general r-tuple set $\{b_1, \ldots, b_r\}$). Here, g takes the form

$$g(t; a; b) = g_1(f_1^{(m_1)}(t; a)) \cdot g_2(f_2^{(m_2)}(t; b)) \tag{B.24}$$

with partial derivatives given by

$$\frac{\partial g}{\partial a_i}(t_0; a; b) = \frac{\partial g_1}{\partial f_1^{(m_1)}}(f_1^{(m_1)}(t_0; a)) \cdot f_1^{(m_1)}(t; e_i)|_{t=t_0} \cdot g_2(f_2^{(m_2)}(t_0; b)) \tag{B.25}$$

and

$$\frac{\partial g}{\partial b_j}(t_0; a; b) = g_1(f_1^{(m_1)}(t_0; a)) \cdot \frac{\partial g_2}{\partial f_2^{(m_2)}}(f_2^{(m_2)}(t_0; b)) \cdot f_2^{(m_2)}(t; e_j)|_{t=t_0}. \tag{B.26}$$

Bibliography

[1] Albus, James S. "Outline for a Theory of Intelligence", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.21, No.3, May/June 1991, pp.473-509.

[2] Albus, J., A. Meystel, and S. Uzzaman, "Multiscale Motion Planning", *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997, pp.19-26.

[3] Allgower, Eugene L., and Kurt Georg, *Numerical Continuation Methods, An Introduction*, Springer-Verlag, Berlin, 1990.

[4] Amit, Ramesh, and Maja J. Matarić, "Parametric Primitives for Motor Representation and Control", *Proceedings of the International Conference on Robotics and Automation (ICRA-2002)*, Washington DC, May, 2002, pp.863-868.

[5] Amit, Ramesh, and Maja J. Matarić, "Learning Movement Sequences from Demonstration", *Proceedings of the International Conference on Development and Learning (ICDL-2002)*, Cambridge, MA, June, 2002, pp.302-306.

[6] Arikan, Okan, and D.A. Forsyth, "Interactive Motion Generation from Examples", *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, ACM SIGGRAPH, San Antonio, TX, July, 2002, pp.483-490.

[7] Atkeson, Christopher G., "Using Local Trajectory Optimizers To Speed Up Global Optimization in Dynamic Programming", *Advances in Neural Information Processing Systems*, edited by Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Vol.6, Morgan Kaufmann Publishers, Inc., 1994, pp.663-670.

[8] Bellingham, John, Arthur Richards, and Jonathan P. How, "Receding Horizon Control of Autonomous Aerial Vehicles", *Proceedings of the American Control Conference*, Anchorage, AK, May, 2002.

[9] Bemporad, A., N.A. Bozinis, V. Dua, M. Morari, and E.N. Pistikopouls, "Model Predictive Control: A Multi-Parametric Programming Approach", *European Symposium on Computer Aided Process Engineering*. Florence, Italy, May, 2000.

[10] Bemporad, A. and C. Fillipi, "Suboptimal Explicit Receding Horizon Control via Approximate Multiparametric Quadratic Programming", *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, FL, December, 2001, pp.4851-4856.

[11] Bemporad, Alberto, and Manfred Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints", *Automatica*, Vol.35, 1999, pp.407-427.

[12] Bemporad, A., T.J. Tarn, and N. Xi, "Predictive Path Parameterization for Constrained Robot Control", *IEEE Transactions on Control Systems Technology*, Vol.7, No.6, November, 1999, pp.648-656.

[13] Berry, D.T., "National Aerospace Plane Longitudinal Long-Period Dynamics", *Journal of Guidance, Control, and Dynamics*, Vol.14, No.1, January-February, 1991, pp.205-206.

[14] Bertsekas, Dimitri P., *Nonlinear Programming*, Second Edition, Athena Scientific, Belmont, MA, 1999.

[15] Bertsimas, Dimitris and John N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997.

[16] Betts, John T., "Survey of Numerical Methods for Trajectory Optimization", *Journal of Guidance, Control, and Dynamics*, Vol.21, No.2, March-April, 1998, pp.193-207.

[17] Betts, John T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[18] Betts, John T., and William P. Huffman, "Path-Constrained Trajectory Optimization Using Sparse Sequential Quadratic Programming", *Journal of Guidance, Control, and Dynamics*, Vol.16, No.1, January-February, 1993, pp.59-68.

[19] Bigelow, James H., and Norman Z. Shapiro, "Implicit Function Theorems for Mathematical Programming and for Systems of Inequalities", *Mathematical Programming*, Vol.6, North-Holland Publishing Company, 1974, pp.141-156.

[20] Borrelli, Francesco, *Constrained Optimal Control of Linear and Hybrid Systems*, Springer-Verlag, 2003.

[21] Borrelli, Francesco, Mato Baotic, Alberto Bemporad, and Manfred Morari, "Efficient On-Line Computation of Constrained Optimal Control", *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, FL, December, 2001, pp.1187-1192.

[22] Brockett, Roger W., "Hybrid Models for Motion Control Systems", *Essays on Control: Perspectives in the Theory and its Applications*, edited by H.L. Trentelman and J.C. Willems, Birkhäuser, Boston, 1993, pp.29-53.

[23] Brotman, Lynne Shapiro, and Arun N. Netravali, "Motion Interpolation by Optimal Control", *Proceedings of ACM SIGGRAPH 1988*, ACM SIGGRAPH, Vol.22, No.4, August, 1988, pp.309-315.

[24] Bruderlin, Armin, and Lance Williams, "Motion Signal Processing", *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1995)*, ACM SIGGRAPH, Los Angeles, CA, August, 1995, pp.97-104.

[25] Bryson, Arthur E., *Dynamic Optimization*, Addison Wesley Longman, Menlo Park, CA, 1999.

[26] Bryson, Arthur E., and Yu-Chi Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Revised Printing, Taylor & Francis, New York, NY, 1975.

[27] Bullo, Francesco, and W. Todd Cerven, "On Trajectory Optimization for Polynomial Systems via Series Expansions", *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December, 2000, pp.772-777.

[28] Bullo, F., and N.E. Leonard, "Motion Primitives for Stabilization and Control of Underactuated Vehicles", *Nonlinear Control Systems Design Symposium*, IFAC, Vol.1, July, 1998, pp.133-138.

[29] Bullo, Francesco, and Kevin M. Lynch, "Kinematic Controllability and Decoupled Trajectory Planning for Underactuated Mechanical Systems", *IEEE Conference in Robotics and Automation*, Seoul, Korea, April, 2001, pp.3300-3307.

[30] Chauvin, Jonathan, Laure Sinegre, and Richard M. Murray, "Nonlinear Trajectory Generation for the Caltech Multi-Vehicle Wireless Testbed", *2003 European Control Conference*, Submitted, 2003, *http://www.cds.caltech.edu/ murray/papers.*

[31] Crandall, Stephen H., Dean C. Karnopp, Edward F. Kurtz, Jr., and David C. Pridmore-Brown, *Dynamics of Mechanical and Electromechanical Systems*, Krieger Publishing Company, Malabar, Florida, 1968.

[32] Dasgupta, Anirvan, and Yoshihiko Nakamura, "Making Feasible Walking Motion of Humanoid Robots from Human Motion Capture Data", *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, MI, May, 1999, pp.1044-1049.

[33] Davis, Paul, "B-Splines and Geometric Design", *SIAM News*, Vol.29, No.5, June, 1996.

[34] de Boor, Carl, *A Practical Guide to Splines*, Revised Edition, Applied Mathematical Sciences, Vol.27, edited by J.E. Marsden and L. Sirovich, Springer, New York, NY, 2001.

[35] de Boor, Carl, *Spline Toolbox User's Guide*, The MathWorks, Inc., Natick, MA, 2003.

[36] Del Vecchio, Domitilla, Richard M. Murray, and Pietro Perona, "Primitives for Human Motion: A Dynamical Approach", *IFAC World Congress*, CDS Technical Report 01-009, 2002.

[37] Dever, Chris, Bernard Mettler, Marc McConley, and Eric Feron, "Singular Direction Characterization of Vehicle Maneuvers", C.S. Draper Laboratory, CSDL-C-6540, Cambridge, MA, April, 2004.

[38] Donald, Bruce Randall, Patrick G. Xavier, John F. Canny, and John H. Reif, "Kinodynamic Motion Planning", *Journal of the Association for Computing Machinery*, Vol.40, No.5, 1993, pp.1048-1066.

[39] Doyle, Francis J., Ronald K. Pearson, and Babatunde A. Oqunnaike, *Identification and Control Using Volterra Models*, Springer, London, 2002.

[40] Earl, Matthew G. and Raffaelo D'Andrea, "Modeling and Control of a Multi-Agent System Using Mixed Integer Linear Programming", *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December, 2002, pp.107-111.

[41] Egerstedt, M., T.J. Koo, F. Hoffman, and S. Sastry, "Path Planning and Flight Controller Scheduling for an Autonomous Helicopter", *Lecture Notes in Computer Science 1569*, edited by J. H. van Schuppen and F. W. Vaandrager, Springer-Verlag, Germany, 1999, pp.91-102.

[42] Faiz, Nadeem, Sunil K. Agrawal, and Richard M. Murray, "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities", *Journal of Guidance, Control, and Dynamics*, Vol.24, No.2, March-April, 2001, pp.219-227.

[43] Faloutsos, Petros, Michiel van de Panne, and Demetri Terzopoulos, "Composable Controllers for Physics-Based Character Animation", *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*, ACM SIGGRAPH, Los Angeles, CA, August, 2001, pp.251-260.

[44] Fiacco, Anthony V., *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Mathematics in Science and Engineering, Volume 165, Academic Press, New York, NY, 1983.

[45] Fiacco, Anthony V., Ed., *Sensitivity, Stability, and Parametric Analysis*, Mathematical Programming Study 21, The Mathematical Programming Society, Amsterdam, The Netherlands, 1984.

[46] Fletcher, Roger, *Practical Methods of Optimization, Volume 1: Unconstrained Optimization*, Wiley, Chichester, England, 1987.

[47] Fletcher, Roger, *Practical Methods of Optimization, Volume 2: Constrained Optimization*, Wiley, Chichester, England, 1987.

[48] Fliess, Michel, Jean Lévine, Philippe Martin, and Pierre Rouchon, "Flatness and Defect of Non-Linear Systems; Introductory Theory and Examples", *International Journal of Control*, Vol.61, No.6, 1995, pp.1327-1361.

[49] Fod, Ajo, Maja J. Matarić, and Odest Chadwicke Jenkins, "Automated Derivation of Primitives for Movement Classification", *Autonomous Robots*, Vol.12, No.1, January, 2002, pp.39-54.

[50] Fourer, Robert, David M. Gay, and Brian W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Second Edition, Duxbury Press, Brooks/Cole Publishing Company, 2003.

[51] Frazzoli, Emilio "Robust Hybrid Control for Autonomous Vehicle Motion Planning", Doctoral Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[52] Frazzoli, Emilio, "Explicit Solutions for Optimal Maneuver-Based Motion Planning", invited paper, IEEE Conference on Decision and Control, 2003.

[53] Frazzoli, Emilio, Munther A. Dahleh, and Eric Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles", *Journal of Guidance, Control, and Dynamics*, Vol.25, No.1, January-February, 2002, pp.116-129.

[54] Gavrilets, Vladislav, "Autonomous Aerobatic Maneuvering of Miniature Helicopters", Doctoral Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[55] Gavrilets, Vladislav, Emilio Frazzoli, Bernard Mettler, Mike Piedmonte, and Eric Feron, "Aggressive Maneuvering of Small Autonomous Helicopters: A Human-Centered Approach", *International Journal of Robotics Research*, Vol.20, No.10, pp.795-807, 2001.

[56] Gavrilets, Vladislav, Ioannis Martinos, Bernard Mettler, and Eric Feron, "Control Logic for Automatic Aerobatic Flight of a Miniature Helicopter", AIAA Guidance, Navigation, and Control Conference, Monterey, CA, August, 2002.

[57] Gavrilets, Vladislav, Ioannis Martinos, Bernard Mettler, and Eric Feron, "Flight Test and Simulation Results for an Autonomous Aerobatic Helicopter", AIAA/IEEE Digital Avionics Systems Conference, Irvine, CA, October, 2002.

[58] Gavrilets, Vladislav, Bernard Mettler, and Eric Feron, "Nonlinear Model for a Small-Size Acrobatic Helicopter", *AIAA Guidance, Navigation, and Control Conference*, Montréal, Canada, August, 2001.

[59] Gleicher, Michael, "Motion Editing with Spacetime Constraints", *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, RI, 1997, pp.139-148.

[60] Golub, Gene H., and Charles F. Van Loan, *Matrix Computations*, Third Edition, The John Hopkins University Press, Baltimore, MD, 1996.

[61] Guddat, Jürgen, F. Guerra Vazquez, and H. Th. Jongen, *Parametric Optimization: Singularities, Pathfollowing and Jumps*, John Wiley & Sons, Ltd., West Sussex, England, 1990.

[62] Guillemin, Victor, and Alan Pollack, *Differential Topology*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[63] Hargraves, C.R., and S.W. Paris, "Direct Trajectory Optimization Using Nonlinear Programming and Collocation", *AIAA Journal of Guidance*, Vol.10, No.4, 1987, pp.338-342.

[64] Hauser, J., and A. Jadbabaie, "Aggressive Maneuvering of a Thrust Vectored Flying Wing: A Receding Horizon Approach", *Proceedings of the IEEE Conference on Decision and Control*, Sydney, Australia, 2000.

[65] Hauser, J., and D. Meyer, "Trajectory Morphing for Nonlinear Systems", *Proceedings of the American Control Conference*, June, 1998.

[66] Hodgins, Jessica K., Wayne L. Wooten, David C. Brogan, James F. O'Brien, "Animating Human Athletics", *Computer Graphics*, Vol.29, 1995, pp.71-78.

[67] Hollerbach, John M., "Dynamic Scaling of Manipulator Trajectories", *ASME Journal of Dynamic Systems, Measurement and Control*, Vol.106, 1984, pp.102-106.

[68] Hull, David G., "Conversion of Optimal Control Problems into Parameter Optimization Problems", *Journal of Guidance, Control, and Dynamics*, Vol.20, No.1, January-February, 1997, pp.57-60.

[69] ILOG, Inc., *ILOG CPLEX 7.0 Reference Manual*, URL: `www.ilog.com`, ILOG, Inc., 2000.

[70] Ijspeert, Auke Jan, Jun Nakanishi, and Stefan Schaal, "Trajectory Formation for Imitation with Nonlinear Dynamical Systems", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2001)*, 2001, pp.752-757.

[71] Jacobson, David H. and David Q. Mayne, *Differential Dynamic Programming*, American Elsevier Publishing Company, New York, NY, 1970.

[72] Jadbabaie, Ali, "Receding Horizon Control of Nonlinear Systems: A Control Lyapunov Function Approach", Doctoral Thesis, California Institute of Technology, Pasadena, California, 2000.

[73] Jadbabaie, Ali, and John Hauser, "Relaxing the Optimality Condition in Receding Horizon Control", *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December, 2000.

[74] Jenkins, Odest Chadwicke, and Maja J. Matarić, "Deriving Action and Behavior Primitives from Human Motion Data", *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002)*, Lausanne, Switzerland, September, 2002, pp.2551-2556.

[75] Johansen, Tor. A., "On Multi-Parametric Nonlinear Programming and Explicit Nonlinear Model Predictive Control", *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, 2002.

[76] Jolliffe, I.T., *Principal Component Analysis*, Second Edition, Springer-Verlag, New York, 2002.

[77] Jongen, Hubertus Th., and Gerhard-W. Weber, "On Parametric Nonlinear Programming", *Annals of Operations Research*, Vol.27, 1990, pp.253-284.

[78] Kato, Osamu, and Ichiro Sugiura, "An Interpretation of Airplane General Motion and Control as Inverse Problem", *Journal of Guidance*, Vol.9, No.2, 1985, pp.198-204.

[79] Khalil, Hassan K., *Nonlinear Systems*, Second Edition, Prentice Hall, Upper Saddle River, NY, 1996.

[80] Kim, S.K., and D. Tilbury, "Trajectory Generation for a Class of Nonlinear Systems with Input and State Constraints", ACC01-AIAA1043, 2001.

[81] Ko, Hyeongseok, and Norman I. Badler, "Animating Human Locomotion with Inverse Dynamics", *IEEE Computer Graphics and Applications*, Vol.16, No.2, 1996, pp.50-59.

[82] Kojima, Masakazu, and Ryuichi Hirabayashi, "Continuous Deformation of Nonlinear Programs", *Mathematical Programming Study*, Vol.21, North-Holland Publishing Company, 1984, pp.150-198.

[83] Koo, T. John and Shankar Sastry, "Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization", *Proceedings of the 37th IEEE Conference on Decision & Control*, Tampa, FL, December, 1998, pp.3635-3640.

[84] Kovar, Lucas, Michael Gleicher, and Frédéric Pighin, "Motion Graphs", *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, ACM SIGGRAPH, San Antonio, TX, July, 2002, pp.473-482.

[85] Laszlo, Joseph, Michiel van de Panne, and Eugene Fiume, "Interactive Control for Physically-Based Animation", *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, ACM SIGGRAPH, New Orleans, LA, July, 2000, pp.201-208.

[86] Lee, Jehee, and Sung Yong Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-like Figures", *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1999)*, edited by Alyn Rockwood, ACM SIGGRAPH, Los Angeles, CA, August, 1999, 39-48.

[87] Lee, Suchang, and Youdan Kim, "Solution of the Inverse Simulation Problem by Optimization Techniques and Its Applications to Aircraft Nonlinear Large Angle Maneuvers", AIAA-96-3701-CP, 1996.

[88] Leishman, J. Gordon, *Principles of Helicopter Aerodynamics*, Cambridge University Press, Cambridge, UK, 2000.

[89] Li, Yan, Tianshu Wang, and Heung-Yeung Shum, "Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis", *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, ACM SIGGRAPH, San Antonio, TX, July, 2002, pp.465-472.

[90] Ljung, Lennart, *System Identification: Theory for the User*, Second Edition, Prentice Hall PTR, Upper Saddle River, NY, 1999.

[91] Lundberg, Bruce N., and Aubrey B. Poore, "Variable Order Adams-Bashforth Predictors with an Error-Stepsize Control for Continuation Methods", *SIAM Journal on Scientific and Statistical Computing*, Vol.12, No.3, May, 1991, pp.695-723.

[92] Lundberg, Bruce N., and Aubrey B. Poore, "Numerical Continuation and Singularity Detection Methods for Parametric Nonlinear Programming", *SIAM Journal on Optimization*, Vol.3, No.1, February, 1993, pp.134-154.

[93] Maciejowski, Jan Marian, *Multivariable Feedback Design*, Addison-Wesley, Wokingham, England, 1989.

[94] Mallat, Stéphane, *A Wavelet Tour of Signal Processing*, Second Edition, Academic Press, San Diego, CA, 1999.

[95] Martinos, Ioannis, Tom Schouwenaars, Jan De Mot, and Eric Feron, "Hierarchical Cooperative Multi-Agent Navigation Using Mathematical Programming", *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October, 2003.

[96] The MathWorks, Inc. *Using Simulink, Model-Based and System-Based Design*, User's Guide, Version 5, The MathWorks Inc., Natick, MA, 2003.

[97] The MathWorks, Inc. *Optimization Toolbox For Use with Matlab*, User's Guide, Version 3, The MathWorks Inc., Natick, MA, 2004.

[98] McGovern, Lawrence K., "Computational Analysis of Real-Time Convex Optimization for Control Systems", Doctoral Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[99] Mettler, Bernard, *Identification Modeling and Characteristics of Miniature Rotorcraft*, Kluwer Academic Publishers, Norwell, MA, 2003.

[100] Mettler, Bernard, Mario Valenti, Tom Schouwenaars, Emilio Frazzoli, and Eric Feron, "Rotorcraft Motion Planning for Agile Maneuvering", *Proceedings of the 58th Forum of the American Helicopter Society*, Montréal, Canada, June, 2002.

[101] Milam, Mark B., Ryan Franz, and Richard M. Murray, "Real-Time Constrained Trajectory Generation Applied to a Flight Control Experiment", *IFAC Conference*, 2002.

[102] Milam, M., K. Mushambi, and R.M. Murray, "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems", *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol.1, 2000, pp.845-851.

[103] Monteiro, Renato D.C. and Ilan Adler, "Interior Path Following Primal-Dual Algorithms, Part II: Convex Quadratic Programming", *Mathematical Programming*, Vol.44, No.1, pp.43-66.

[104] Munkres, James R., *Analysis on Manifolds*, Westview Press, 1991.

[105] Murray, Richard M., Joel W. Burdick, Scott D. Kelly, and James Radford, "Trajectory Generation for Mechanical Systems with Application to Robotic Locomotion", *Proceedings of the 3rd International Workshop on Algorithmic Foundations of Robotics*, Houston, TX, 1998.

[106] Murray, Richard M., Muruham Rathinam, and Willem Sluis, "Differential Flatness of Mechanical Control Systems - A Catalog of Prototype Systems", *ASME International Mechanical Engineering Congress and Exposition*, 1995.

[107] Mussa-Ivaldi, F., and E. Bizzi, "Motor Learning Through the Combination of Primitives", *Philosophical Transactions for the Royal Society of London*, Vol.355, 2000, pp.1755-1769.

[108] Nise, Norman S., *Control Systems Engineering*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1995.

[109] O'Neill, Barret, *Semi-Riemannian Geometry with Applications to Relativity*, Academic Press, San Diego, CA, 1983.

[110] O'Neill, Barrett, *Elementary Differential Geometry*, Second Edition, Academic Press, San Diego, CA, 1997.

[111] Papoulis, Athanasios, and S. Unnikrishna Pillai, *Probability, Random Variables and Stochastic Processes*, Fourth Edition, McGraw-Hill Series in Electrical and Computer Engineering, McGraw-Hill, Boston, MA, 2002.

[112] Peterson, David W., "A Review of Constraint Qualifications in Finite-Dimensional Spaces", *SIAM Review*, Vol.15, No.3, July, 1973, pp.639-654.

[113] Petit, N., M. Milam, and R. Murray, "Inversion Based Constrained Trajectory Optimization", *5th IFAC Symposium on Nonlinear Control Systems*, 2001.

[114] Piedmonte, Mike, and Eric Feron, "Aggressive Maneuvering of Aerial Vehicles: A Human-Centered Approach", International Symposium on Robotics Research, Snowbird, UT, October, 1999.

[115] Pierre, Donald A., *Optimization Theory with Applications*, Dover Publications, Inc., New York, NY, 1986.

[116] Pinch, Enid R., *Optimal Control and the Calculus of Variations*, Oxford University Press, Oxford, England, 1993.

[117] Pistikopoulos, Efstratios N., Vivek Dua, Nikolaos A. Bozinis, Alberto Bemporad, and Manfred Morari, "On-Line Optimization via Off-Line Parametric Optimization Tools", *Computers and Chemical Engineering*, Vol.26, 2002, pp.175-185.

[118] Poore, A.B. and C.A. Tiarht, "Bifurcation Problems in Nonlinear Parametric Programming", *Mathematical Programming*, Vol.39, 1987, pp.189-205.

[119] Popović, Jovan, Steven M. Seitz, and Michael Erdmann, "Motion Sketching for Control of Rigid Body Simulations" ACM Transactions on Graphics, Vol.22, No.4, 2003, pp.1034-1054.

[120] Popović, Jovan, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin, "Interactive Manipulation of Rigid Body Simulations", *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, ACM SIGGRAPH, New Orleans, LA, July, 2000, pp.209-217.

[121] Popović, Zoran, and Andrew Witkin, "Physically Based Motion Transformation", *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1999)*, ACM SIGGRAPH, Los Angeles, CA, August, 1999, pp.11-20.

[122] Quanser Consulting, "3D Helicopter System (with Active Disturbance)", User's Manual, Quanser Consulting, Markham, Ontario, 2003.

[123] Rakowska, Joanna, Robert T. Haftka, and Layne T. Watson, "An Active Set Algorithm for Tracing Parametrized Optima", *Structural Optimization*, Vol.3, 1991, pp.29-44.

[124] Rheinboldt, Werner C., *Numerical Analysis of Parametrized Nonlinear Equations*, University of Arkansas Lecture Notes in the Mathematical Sciences, Volume 7, John Wiley & Sons, New York, NY, 1986.

[125] Richards, Arthur, and Jonathan P. How, "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming", *Proceedings of the American Control Conference*, Anchorage, AK, May, 2002.

[126] Robinson, Stephen M., "Local Structure of Feasible Sets in Nonlinear Programming, Part I: Regularity", *Numerical Methods*, edited by V. Pereyra and A. Reinoza, Springer-Verlag, Berlin, 1983, pp.240-251.

[127] Roweis, Sam, and Lawrence Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding", *Science*, Vol.290, No.5500, December, 2000, pp.2323-2326.

[128] Rudin, Walter, *Principles of Mathematical Analysis*, Third Edition, McGraw-Hill, Inc., New York, NY, 1964.

[129] Schaal, Stefan, "Learning from Demonstration", *Advances in Neural Information Processing Systems*, edited by Michael C. Mozer, Micheal I. Jordan, and Thomas Petsche, Vol.9, MIT Press, Cambridge, MA, 1997, pp.1040-1046.

[130] Schaal, Stefan, Shinya Kotosaka, and Dagmar Sternad, "Nonlinear Dynamic Systems as Movement Primitives", *CD-Proceedings of Humanoids2000, First IEEE-RAS International Conference on Humanoid Robots*, Cambridge, MA, September, 2000.

[131] Schouwenaars, Tom, Bart De Moor, Eric Feron, and Jonathan How, "Mixed Integer Programming for Multi-Vehicle Path Planning", European Control Conference, Porto, Portugal, 2001.

[132] Schouwenaars, Tom, Eric Feron, and Jonathan How, "Safe Receding Horizon Path Planning for Autonomous Vehicles", 40th Allerton Conference on Communication, Control and Computing, October, 2002.

[133] Schouwenaars, Tom, Bernard Mettler, Eric Feron, and Jonathan How, "Hybrid Architecture for Full-Envelope Autonomous Rotorcraft Guidance", American Helicopter Society 59th Annual Forum, Phoenix, AZ, 2003.

[134] Schouwenaars, Tom, Bernard Mettler, Eric Feron, and Jonathan How, "Hybrid Architecture for Receding Horizon Guidance of Agile Autonomous Rotorcraft", 16th IFAC Symposium on Automatic Control in Aerospace, St. Petersburg, Russia, June, 2004.

[135] Seferlis, Panagiotis, "Collocation Models for Distillation Units and Sensitivity Analysis Studies in Optimization", Doctoral Thesis, McMaster University, Hamilton, Ontario, Canada, April, 1995.

[136] Seywald, Hans, "Trajectory Optimization Based on Differential Inclusion", *Journal of Guidance, Control, and Dynamics*, Vol.17, No.3, May-June, 1994, pp.480-487.

[137] Shim, H., T.J. Koo, F. Hoffmann, and S. Sastry, "A Comprehensive Study on Control Design of Autonomous Helicopter", *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, December, 1998, pp.3653-3658.

[138] Slotine, Jean-Jacques E. and Weiping Li, *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[139] Spivak, Michael, *Calculus on Manifolds*, Perseus Books Publishing, 1965.

[140] Sternad, Dagmar, and Stefan Schaal, "Segmentation of Endpoint Trajectories Does Not Imply Segmented Control", *Experimental Brain Research*, Vol.124, No.1, pp.118-136.

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

[142] Tenenbaum, J.B., V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science*, Vol.290, No.5500, December, 2000, pp.2319-2323.

[143] Thomson, Douglas G., and Roy Bradley, "The Mathematical Definition of Helicopter Maneuvers", *Journal of the American Helicopter Society*, October, 1997, pp.307-309.

[144] Tiahrt, C.A. and A.B. Poore, "A Bifurcation Analysis of the Nonlinear Parametric Programming Problem", *Mathematical Programming*, Vol.47, 1990, pp.117-141.

[145] Tischler, M.B., and M.G. Cauffman, "Comprehensive Identification from Frequency Responses: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics", *Journal of the American Helicopter Society*, Vol.37, No.3, 1992, pp.3-17.

[146] Tischler, M.B., and M.G. Cauffman, "Comprehensive Identification from Frequency Responses: An interactive facility for system identification and verification, Class Notes and User's Manual", NASA Ames Research Center, Moffett Field, CA, September, 1994.

[147] Tøndel, Petter, Tor Arne Johansen, and Alberto Bemporad, "An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions", *Automatica*, Vol.39, No.3, 2003, pp.489-497.

[148] Vanderberghe, Lieven, and Stephen Boyd, "Semidefinite Programming", *SIAM Review*, Vol.38, No.1, 1996, pp.49-95.

[149] van Nieuwstadt, M., "Trajectory Generation for Nonlinear Control Systems", Doctoral Thesis, California Institute of Technology, 1997.

[150] van Nieuwstadt, M., and R. Murray, "Outer Flatness: Trajectory Generation for a Model Helicopter", *Proceedings of the European Control Conference*, 1997.

[151] van Nieuwstadt, M., M. Rathinam, and R.M. Murray "Differential Flatness and Absolute Equivalence of Nonlinear Control Systems", *SIAM Journal on Control and Optimization*, Vol.36, No.4, 1998, pp.1225-1239.

[152] Verma, A.J., and J.L. Junkins, "Trajectory Generation for Transition from VTOL to Wing-Bourne Flight Using Inverse Dynamics", *38th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January, 2000.

[153] Walker, D.J., and I. Postlethwaite, "Advanced Helicopter Flight Control Using Two-Degree-of-Freedom H^{∞} Optimization", *Journal of Guidance, Control, and Dynamics*, Vol.19, No.2, March-April, 1996, pp.461-468.

[154] Witkin, Andrew, and Michael Kass, "Spacetime Constraints", *Proceedings of ACM SIGGRAPH 1988*, ACM SIGGRAPH, Vol.22, No.4, August, 1988, pp.159-168.

[155] Witken, Andrew, and Zoran Popović, "Motion Warping", *Proceedings of 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1995)*, edited by Robert Cooke, ACM SIGGRAPH, Los Angeles, CA, August, 1995, pp.105-108.

[156] Wright, Stephen J., *Primal-Dual Interior Point Methods*, SIAM, Philadelphia, PA, 1997.

[157] Wu, Jia-chi, and Zoran Popović, "Realistic Modeling of Bird Flight Animations", *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2003)*, ACM SIGGRAPH, San Diego, CA, July, 2003, pp.888-895.

[158] Zwillinger, Daniel, Editor-in-Chief, *CRC Standard Mathematical Tables and Formulae*, Thirtieth Edition, CRC Press, Boca Raton, FL, 1996.