Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems

Thesis by
Mark B. Milam

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2003

(Defended May 23, 2003)

To Mom and Dad

Acknowledgements

The work in this thesis would not of been possible without the help of many individuals. I would like to express my gratitude to Kudah Mushambi and Samuel Chang for their contributions to the software package that resulted from this research. I would like to acknowledge Simon Yeung, Jim Radford, Bill Dunbar, Muruhan Rathinam, Michiel van Nieuwstadt, Sudipto Sur, and Francesco Bullo for being generous with their time. Additionally, I would like to thank Youbin Mao for the many invaluable discussions we have had over the years. Ryan Franz was instrumental in getting the Caltech Ducted Fan experiment operational. I thank him for his help on the experiment as well as introducing me to the sport of rock climbing.

Prof. Nicolas Petit greatly influenced the work in this thesis. I would like to thank him for his contributions to the micro-satellite project as well as evaluating the performance of the proposed software algorithm in this thesis. I look forward to collaborating with Nick in the future. My thanks go out to the dynamical systems mavens: Wang Sang Koon and Prof. Jerrold Marsden provided the opportunity to work on the micro-satellite project; Dong-Eui Chang for working with me on the mathematical intricacies of differential flatness. I acknowledge Prof. Joel Burdick for accepting to be a member of my thesis committee. Thanks to Prof. John Hauser for contributing to the receding horizon control chapter of this thesis as well as the design of the ducted fan experiment. I was surprised when we met our goal of implementing receding horizon control on the experiment before I graduated. I would like to reserve special acknowledgment for my mentor and exemplary adviser, Prof. Richard Murray. Richard provided an unique multi-discipline research environment as well as the inspiration, funding, guidance, and motivation for my work.

I am indebted to Charmaine Boyd for handling the endless administrative details of the Caltech ducted fan experiment. Thanks also to Rodney Rojas and John van Deusen for their machining expertise. I will never forget the day when

Rodney and I hoisted the 175 kg Caltech ducted fan experiment on its pedestal. I owe gratitude to Northrop Grumman for providing a PhD fellowship to help support my research.

I would like to thank to my parents, to whom I have dedicated this thesis, for their support throughout the years. My wife, Joy, has always been loving throughout this difficult experience while working full-time and raising our family. She has always been my greatest source of energy to finish this thesis. For this, I am forever grateful.

Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems

by

Mark B. Milam

In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Abstract

With the advent of powerful computing and efficient computational algorithms, real-time solutions to constrained optimal control problems are nearing a reality. In this thesis, we develop a computationally efficient Nonlinear Trajectory Generation (NTG) algorithm and describe its software implementation to solve, in real-time, nonlinear optimal trajectory generation problems for constrained systems. NTG is a nonlinear trajectory generation software package that combines nonlinear control theory, B-spline basis functions, and nonlinear programming. We compare NTG with other numerical optimal control problem solution techniques, such as direct collocation, shooting, adjoints, and differential inclusions.

We demonstrate the performance of NTG on the Caltech Ducted Fan testbed. Aggressive, constrained optimal control problems are solved in real-time for hover-to-hover, forward flight, and terrain avoidance test cases. Real-time trajectory generation results are shown for both the two-degree of freedom and receding horizon control designs. Further experimental demonstration is provided with the station-keeping, reconfiguration, and deconfiguration of micro-satellite formation with complex nonlinear constraints. Successful application of NTG in these cases demonstrates reliable real-time trajectory generation, even for highly nonlinear and non-convex systems. The results are among the first to apply receding horizon control techniques for agile flight in an experimental setting, using representative

dynamics and computation.

Contents

1 Introduction **1**

1.1 Previous and Parallel Work 4

1.2 Overview and Statement of Contributions 6

2 Background and Mathematical Framework **8**

2.1 Optimal Control Problems under Consideration 8

2.2 Necessary Conditions of Optimality for Constrained Systems . . . 9

2.3 Trajectory Generation Using Homotopy 11

2.3.1 Algorithm Description 12

2.3.2 Ducted Fan Example 15

2.4 Nonlinear Programming Techniques 19

2.4.1 Optimality Conditions 20

2.4.2 Techniques for Global Convergence 22

2.4.3 Trust-Region Methods 23

2.4.4 Merit Functions 23

2.4.5 Simple Infeasible Nonlinear Programming Algorithm Using a Line-search 24

2.4.6 Active Set Methods 25

2.5 Numerical Solutions of Optimal Control Problems Using Nonlinear Programming 25

2.5.1 Direct Methods of Solution Using Nonlinear Programming 25

2.5.2 Transcription Techniques from the Optimal Control Problem to the Nonlinear Programming Problem 27

2.6 Trajectory Generation Using Feedback Linearization 30
2.6.1 Mathematical Background 30
2.6.2 Classical Feedback Linearization 31
2.6.3 Devasia-Chen-Paden Non-minimum Phase Zero-Dynamics Algorithm 36
2.7 Differential Flatness 36
2.8 System Design and Differential Flatness 39
2.9 Parameterization of the Output 47
2.10 Summary and Conclusions 51

3 The Nonlinear Trajectory Generation (NTG) Algorithm 53
3.1 Transforming the System Dynamics to a Lower Dimensional Space 53
3.1.1 Differentially Flat Systems 54
3.1.2 An Algorithm to Map the System Dynamics to Lower Dimensional Space 56
3.2 Parameterization of the Flat Outputs and "Pseudo" Outputs . . . 59
3.3 Transformation into a Nonlinear Programming Problem 59
3.4 Conclusion 65

4 NTG Performance Comparisons 66
4.1 An Investigation of NTG, Differential Inclusion, and Collocation Integration Accuracy 66
4.2 Comparison of NTG and RIOTS 67
4.3 An Investigation of the Accuracy of NTG vs. Shooting: The Goddard Problem 71
4.4 Variable Space Reduction 72
4.4.1 Example 76
4.5 Conclusion 84

5 Two Degree of Freedom and Receding Horizon Control (RHC) of the Caltech Ducted Fan 86

5.1 Experimental Setup and Mathematical Model 86
5.2 Two-degree-of-Freedom Design for Constrained Systems 91
5.2.1 Optimization Problem Formulation 91
5.2.2 Timing and Trajectory Management 92
5.2.3 Stabilization around Reference Trajectory 94
5.2.4 “Shuttle” Maneuver 96
5.2.5 Terrain Avoidance 96
5.2.6 Forward-Flight Trajectory Generation 100
5.2.7 Conclusions and Future Work 102
5.3 Receding Horizon Control for Constrained Systems 105
5.3.1 Theoretical Background 107
5.3.2 Receding Horizon Control Formulation 112
5.3.3 NTG Setup 113
5.3.4 Results 117
5.3.5 Conclusion 124

6 Micro-satellite Formation Flying 128

6.1 Introduction 128
6.2 Problem Formulation 131
6.2.1 Micro-Satellite Formation Flying Requirements 133
6.2.2 Micro-Satellite Trajectory Generation 133
6.3 Optimal Control Problem Formulation 134
6.3.1 Station-Keeping with Guaranteed Earth Coverage 135
6.3.2 Nonlinear Formation Reconfiguration and Deconfiguration . 137
6.4 Numerical Results 138
6.4.1 Station-Keeping 138
6.4.2 Results to Problems 2 and 3: Nonlinear Reconfiguration and Deconfiguration 141
6.5 Conclusion 142

7 Future Work 147

List of Figures

1.1 Separation of different levels of trajectory generation. 2
1.2 Two-degree-of-freedom-design. 3
1.3 Receding horizon control design. 3

2.1 UCAV coordinate system conventions. 16
2.2 Ducted fan trajectory generation homotopy example: x: $0 \rightarrow -.8$ and z: $0 \rightarrow -.8$ with large time weight. 18
2.3 A flat system has the important property that the states and the inputs can be written in terms of the outputs z and their derivatives. Thus, the behavior of the system is determined by the flat outputs. Note that the map w is bijective. 38
2.4 The views of the "Aeroranger" show the body coordinate frame, direction of rotation of the propellers, the definition of the applied forces due to the ducted fan thrust, and a method used for stabilizing the aircraft in aerodynamic flight without using aerodynamic surfaces. 40
2.5 Two planar VTOL configurations and two six DOF configurations 46
2.6 The B-spline curve has six intervals ($l = 6$), forth order ($k = 4$), and is C^3 at the breakpoints (or smoothness $s = 3$). The nine control points are the decision variables. 49

3.1 In this hypothetical problem, the B-spline curve has six intervals ($l = 6$), fourth order ($k = 4$), and is C^3 at the breakpoints (or smoothness $s = 3$). The constraint on the B-spline curve (to be larger than the constraint in this example) will be enforced at the 21 collocation points. The nine control points are the decision variables. 60

4.1 RIOTS and NTG van der Pol comparison 70
4.2 The different formulations of the optimal control problem for the example. Top: full collocation. Bottom: flatness parametrization. . 78
4.3 Main results. The cpu-time is an exponential decreasing function of the relative degree. Top: full collocation. Bottom: flatness parametrization. 79
4.4 log(Number of variables) versus log(cpu-time). In each case 200 runs were done with random initial guesses. The *variance* of the results is represented by the error bar. The slope of the linear regression of the mean values of cpu-time is 2.80 80
4.5 NTG direct collocation, “semi-flat” and “flat” convergence analysis. The abscissa is the number of convergent test cases out of the 500 for each 6561 test case. The ordinate shows the number of 6561 test cases. 83

5.1 Ducted fan testbed 87
5.2 Ducted fan coordinate frames 88
5.3 B-spline curve fits to wind tunnel and flight test results for the $C_L(\alpha)$,$C_D(\alpha)$, and $C_M(\alpha)$ aerodynamic coefficients. 90
5.4 forward-flight mode equilibrium manifold 93
5.5 Sample run showing joystick input and timing concepts. The initial conditions for each run are denoted with IC, and the final conditions with FC. 93

5.6 "Shuttle" maneuver depicting joystick command , NTG reference trajectory, and actual state 97
5.7 "Shuttle" maneuver depicting all the trajectories that are computed and which trajectories are accepted. For each trajectory, a fixed constant is added to the time axis so that all trajectories can be seen. "X" denotes the start of a trajectory. "O" denotes an accepted trajectory that was applied until completion; that is, the system reached a hover equilibrium. 98
5.8 "Shuttle" Maneuver run times 99
5.9 Terrain avoidance position and pitch attitude of the Caltech Ducted Fan. NTG reference trajectory (dashed) and actual (solid). The attitude (θ) is denoted by the orientation of the airfoil. 99
5.10 Terrain avoidance commanded and actual forces 100
5.11 Hover to hover test case: Altitude and x position for two different vertical trajectory constraints. The actual (solid) and NTG reference trajectories (dashed). The attitude (θ) is denoted by the orientation of the airfoil. 101
5.12 Forward-flight test case: θ and $\dot{x}$ desired and actual. 102
5.13 Forward-flight test case: Desired F_{X_b} and F_{Z_b} with bounds. 103
5.14 Forward-flight accepted trajectory errors 103
5.15 Forward-flight test case: Altitude and x position (actual (solid) and desired(dashed)). Airfoil represents actual pitch angle (θ) of the ducted fan. 104
5.16 Forward-flight test case: 60 second run, 30 computed trajectories, t_{sample}= 2 sec. 104
5.17 Illustration of sub-level set of $J_T^*(\cdot)$ to which the state of the true system will converge 112
5.18 Illustration of timing scheme without prediction. 114
5.19 Illustration of timing scheme with prediction. 115

5.20 Plot depicting the actual attitude and position of the ducted fan throughout both step commands. 119
5.21 The commanded forces in the body frame. There is a nonlinear transformation between F_{X_B}, F_{Z_B} and commanded current to the motor and the thrust vector bucket angle δ_τ. 119
5.22 Horizontal position of the ducted fan 120
5.23 The attitude of the ducted fan 121
5.24 The spatial horizontal velocity of the ducted fan 122
5.25 Force constraints and computation times. 123
5.26 Vertical position of the ducted fan 125

6.1 Orbit and inertial coordinate systems. 132
6.2 Micro-satelllite modes of operation: reconfiguration, station-keeping, and deconfiguration 134
6.3 Station keeping with guaranteed earth coverage 136
6.4 Station-keeping for three micro-satellites. Relative distances (m) . 139
6.5 Station-keeping for three micro-satellites. Projected area (m^2). . . 140
6.6 Station-keeping for three micro-satellites. ΔV (m/s) 141
6.7 Projected area for station-keeping with and without control. Projected area (m^2). Without control, the projected area becomes singular. 142
6.8 Reconfiguration. Going into a formation with a projected area of $450m^2$. Top: projected area versus time. Middle: relative distances versus time. Bottom: thrusts in the body frame, where each colunm is a different satellite. 145
6.9 Deconfiguration. Going to a "parking" configuration. Top: projected area versus time. Middle: relative distances versus time. Bottom: thrusts in the body frame, where each column is a different satellite. 146

List of Tables

2.1 Summary of the flatness characteristics of several simple VTOL configurations. Flat in forward flight implies that the mixing of hover actuation and aerodynamic surfaces is possible. 46

4.1 Summary of results of simple cart comparison showing that exploiting differential flatness compares favorably with other transcription techniques. 68

4.2 RIOTS and NTG van der Pol comparison 71

4.3 NTG Goddard example results showing the increase in accuracy in solution as the number of knot intervals is increased 73

5.1 Maximum acceptable periods as determined in simulation 117

5.2 Maximum acceptable periods as determined on the real experiment 118

6.1 Station-keeping. Top: effect of S for a given d. Bottom: effect of e for a given S. 143

6.2 Reconfiguration ΔV for various objectives. 144

Chapter 1

Introduction

A simple one-degree-of-freedom approach to tracking a real-time target using feedback control system is to subtract the current state of the system from the target and feed this error to the controller. It is well known that this approach frequently fails for a large class of nonlinear mechanical systems with constraints, where a global asymptotically stable solution that satisfies the constraints does not exist. Even when a solution is found, it is difficult to achieve high performance in the presence of constraints.

When used in conjunction with a stabilization technique, generating a trajectory that satisfies the system dynamics has been proven effective in mitigating the deficiencies of the one-degree of freedom design. Typical applications of trajectory generation include obstacle avoidance by a robotic vehicle, minimum time missile interception of an agile target, formation flight of microsatellites with coverage constraints, and a rapid change of attitude for an unmanned flight vehicle to evade a dynamic threat. References [1] and [68] articulate the need for advanced control techniques to accomplish these missions.

Depending on the system setup, there may be more than one level of trajectory generation, as shown in Figure 1.1. At the mission level, an objective along with a set of mission constraints are derived from high level mission inputs, with which a desired trajectory is generated. The time-scale at this level of trajectory generation is on the order of minutes to hours. At the vehicle level, a local trajectory

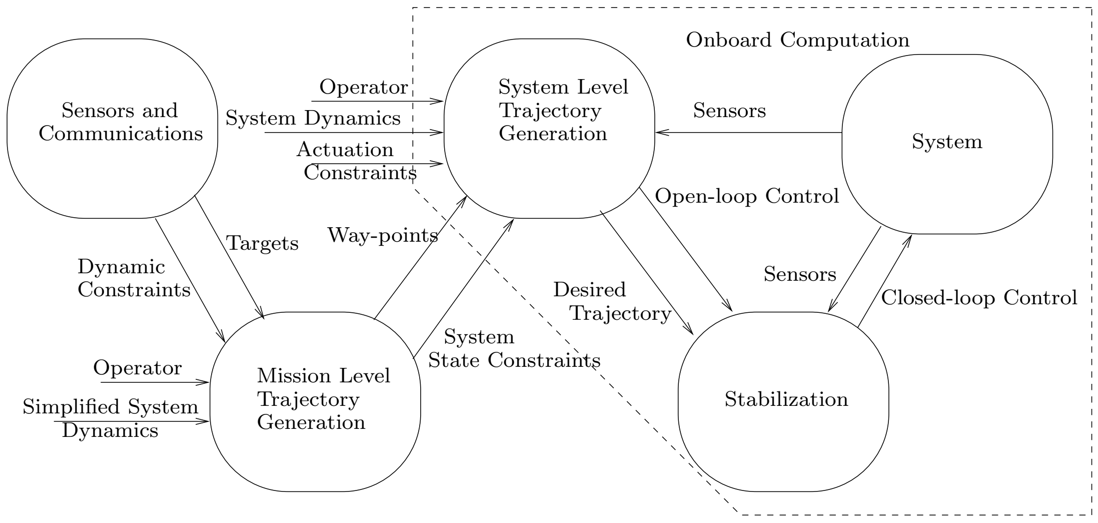

Figure 1.1: Separation of different levels of trajectory generation.

generation algorithm autonomously computes an open loop reference trajectory given the full system dynamics, actuation and boundary constraints, and the cost objective. System level trajectory generation may have a time-scale on the order of tens of milliseconds to minutes, depending on the application. The operator may also directly provide a reference trajectory at the system level.

There are two different approaches and combinations of the two, for system level trajectory generation. One is called the two-degree-of-freedom design, depicted in Figure 1.2, and the other is receding horizon control design, depicted in Figure 1.3.

The two-degree of freedom design consists of a trajectory generator and a linear feedback controller. The trajectory generator provides a feasible feed-forward reference trajectory that satisfies system and actuation constraints. A gain-scheduled feedback controller then stabilizes around and tracks the reference trajectory. The advantage of this approach is that the system is tracking a feasible trajectory along which the system can be stabilized. Furthermore, the reference trajectory can be as aggressive as allowed by the model.

In receding horizon control, an open-loop trajectory is found by solving a finite-horizon constrained optimal control problem starting from the current state. The

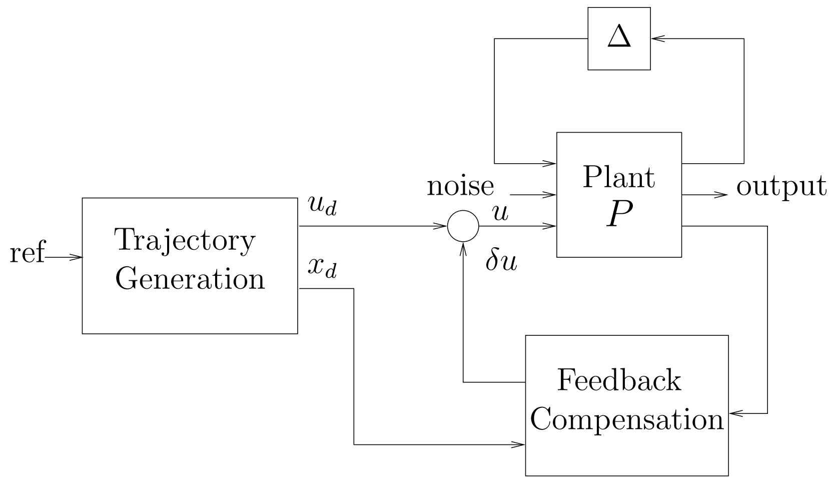

Figure 1.2: Two-degree-of-freedom-design.

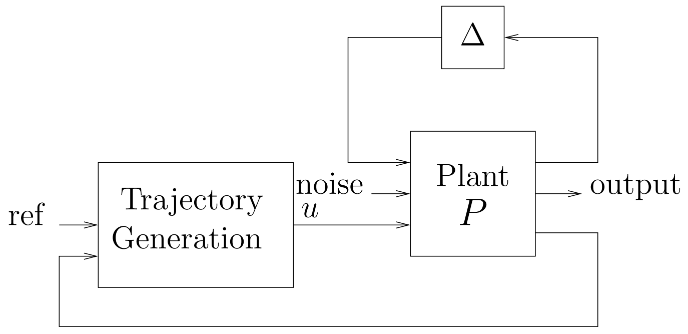

Figure 1.3: Receding horizon control design.

controls of this trajectory are then applied for a certain fraction of the horizon length, after which the process is repeated. It has been shown theoretically that receding horizon control is stabilizing if an appropriate cost function is chosen and the trajectory can be computed quickly.

It is possible to combine the two-degree-of-freedom and receding horizon control designs. For example, one could generate a feasible trajectory using the two-degree-of-freedom design and stabilize with the receding horizon control design.

A prime example, where a two-degree-of-freedom or receding horizon control design would be applied, is an unmanned flight vehicle. The desired objective of the unmanned flight vehicle could be commanded by the operator or pre-programmed without further operator intervention. The desired objective may be to go to a certain point, pass through several way-points, or track a target. For some missions, unmanned flight vehicles will autonomously fly highly aggressive maneuvers, frequently on the fringe of the flight envelope. The idea of directly commanding the unmanned flight vehicle to track a target would be impractical, considering the fast dynamics and constraints of a typical unmanned flight vehicle. Some future unmanned flight vehicles could autonomously launch an attack and return to base or land on aircraft carriers. These vehicles would provide more tactical flexibility than a cruise missile because they would be able to loiter in the area and search for a moving target, which it could then strike with its weapons. Future unmanned flight vehicles will have the ability to respond quickly to a wide range of unforeseeable circumstances in search and rescue, border patrol, and counter-drug operations. A key feature of these unmanned flight vehicles will be their ability to autonomously *plan their own trajectories*. Therefore, the important goal of optimal trajectory generation is to construct, in real time, a solution that optimizes the system objective while satisfying system dynamics, as well as state and actuation constraints.

Hence, it is the objective of this thesis to develop an efficient computational algorithm for real time optimal trajectory generation of constrained systems and demonstrates its effectiveness on an experiment testbed that represents a real-world application.

1.1 Previous and Parallel Work

Most early numerical methods of solution to constrained optimal trajectory generation problems relied on either indirect or direct methods of solution. The indirect method relies on finding a solution to the Pontryagin's maximum principle [83]. This results in finding a numerical solution to a two-point boundary value problem, if no closed form solution can be found. The multiple shooting method, used to solve two-point boundary value problems, is discussed in Pesch [77, 78] and von Stryk [108]. The direct method obtains solutions by direct minimization of the objective function, subject to the constraints of the optimal control problem. An introduction to direct methods can be found in Hargraves and Paris [41] and vonStryk [109]. Techniques were developed to optimally interpolate a set of stored trajectories on-line by Chen and Allgower in [18] and [3], respectively. A technique for generating real time trajectories by searching and interpolating over a large trajectory database in real time can be found in Atkeson [4].

Several randomized trajectory generation techniques have recently been reported and have promising potential for real-time applications. For example, see LaValle [55], Frazzoli [36], and Hsu [44]. These techniques are most applicable to the mission level trajectory generation shown in Figure 1.1.

Another approach to solving the trajectory generation problem would be to use nonlinear geometric control techniques. These techniques attempt to find ad hoc outputs such that the complete differential behavior of the system can be represented in terms of these outputs and their derivatives. The theory of existence and generation of these so-called flat outputs are subjects of Isidori [46], Rathinam [87], Charlet *et al.* [16], Fliess [32, 33], Chetverikov, [19], and [62, 65]. Unfortunately, there are many classes of systems for which these outputs cannot be found, even if they can be proven to exist. However, it is usually not very difficult to find some outputs that will characterize at least part of the system behavior. In this case, attention must be paid to the stability of the resulting zero dynamics which could lead to unbounded controls and states. Techniques were developed in Devasia and Chen [25], Devasia [24], and Verma and Junkins [107] to circumvent this problem. Some methods of real-time trajectory generation without constraints for differentially flat systems are illustrated in van Nieuwstadt [103]. An approach to find feasible trajectories for constrained differentially flat systems by approximating constraints with linear functions is given in Faiz and Agrawal [29]. Agrawal and Faiz in [2] investigated higher-order variation methods to solve optimization problems for feedback linearizable systems without constraints.

An example of work more related to the approach in this thesis can be found in Steinbach [98]. Steinbach shows that combining inverse dynamics with sequential quadratic programming can drastically reduce computation times in robotic motion planning. The work of Oldenburg *et al.* [75] and [76] expands on Steinbach's approach by including the case when the system is not feedback linearizable. Oldenburg relies on the work of van der Schaft in [102], which provides a method to represent a nonlinear state space system as a set of higher-order differential equations in the inputs and outputs. Mahadevan *et al.* in [60] and [61] use Oldenburg's work and apply it to chemical processes. Veeraklaew *et al.* in [106] combines the concepts of differential flatness and sequential quadratic programming. However, his choice of basis functions representing the outputs requires additional continuity constraints to the resulting optimization problem. Bulirsch *et al.* [37] discusses the use of sequential quadratic programming methods to solve trajectory optimization problems.

1.2 Overview and Statement of Contributions

A brief summary and thesis contributions by chapter:

- *Chapter 2:* In this chapter we derive a novel homotopy method, which when used in conjunction with a stored database of trajectories, will find nearby optimal trajectories. Additionally, we discuss the relationships between Vertical Takeoff and Landing (VTOL) design and differential flatness. Finally, we present a promising new VTOL design and demonstrate that it is differentially flat.

- *Chapter 3:* In this chapter we first propose a generic algorithm to reduce the dimension of the system dynamics to facilitate real-time computation. Second, we develop the Nonlinear Trajectory Generation (NTG) algorithm and describe the software implementation intended to solve nonlinear, optimal, real-time, constrained trajectory generation problems. NTG is a software package that combines techniques of nonlinear control, B-splines, and nonlinear programming.

- *Chapter 4:* In this chapter, we compare and contrast NTG with the optimal control problem solution techniques of direct collocation, shooting, adjoints, and differential inclusions.

- *Chapter 5:* In this chapter, we investigate the performance of NTG on the Caltech Ducted Fan experiment. Results are presented for both the two-degree-of-freedom and receding horizon control design. For the two-degree of freedom design, aggressive constrained optimization problems are solved in real-time for hover-to-hover, forward flight, and terrain avoidance test cases. The results confirm the applicability of real-time, nonlinear, constrained receding horizon control. They are among the first to demonstrate the use of receding horizon control for agile flight in an experimental setting, using representative dynamics and computation.

- *Chapter 6:* In this chapter, we provide another example of complex, real-time nonlinear constrained trajectory generation for the station-keeping, reconfiguration, and deconfiguration of micro-satellites.

Chapter 2

Background and Mathematical Framework

In this chapter we will survey the techniques that are currently used for constrained trajectory generation, followed by background material that motivates this research.

2.1 Optimal Control Problems under Consideration

Let $x : [t_0, t_f] \to \mathbb{R}^n$ denote the state of the system and $u : [t_0, t_f] \to \mathbb{R}^m$ the input of the system

$$\dot{x} = f(x, u), \tag{2.1}$$

where all vector fields and functions are real-analytic. It is desired to find a trajectory of (2.1) in $[t_0, t_f]$ that minimizes the performance index functional

$$J := \int_{t_0}^{t_f} L(x, u, t)dt + \phi_f(x_f, u_f, t_f) \tag{2.2}$$

$J : \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}$ subject to a vector of N_0 initial time, N_f final time, and N_t trajectory constraints,

$$\begin{array}{rcl} lb_0 & \leq & \psi_0(x_0, u_0) \leq ub_0, \\ lb_f & \leq & \psi_f(x_f, u_f) \leq ub_f, \\ lb_t & \leq & S(x, u, t) \leq ub_t. \end{array} \tag{2.3}$$

The functions $\psi_0 : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{N_0}$, $\psi_f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{N_f}$, $S : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}^{N_t}$ are assumed to be as least C^2 on appropriate dense open sets. The final time t_f could be either fixed or free.

2.2 Necessary Conditions of Optimality for Constrained Systems

Bryson and Ho [12] and Lewis and Syrmos [58] derive the necessary conditions using the calculus of variations. Pontryagin *et al.* [83] show that finding an extremal solution is equivalent to the requirement that the variational Hamiltonian be minimized, with respect to all admissible inputs.

Without loss of generality, we will assume that there is one control ($m = 1$) and one state inequality constraint ($N_t = 1, S(x,t) \leq 0$) and that the trajectory constraint is active on the time interval $t \in [t_1, t_2] \subset [t_0, t_f]$. An active constraint is known as a constrained arc.

Defining the Hamiltonian H and the auxiliary functions Ξ and Φ,

$$\begin{aligned} H(x,u,\lambda,\mu,t) \quad &:= L(x,u,t) + \lambda^T f(x,u) + \mu^T S^{(q)}(x,u,t) \\ \Xi(x_0,u_0,t_0,\nu_0) \quad &:= \phi_0(x_0,u_0,t_0) + \nu_0^T \psi_0(x_0,u_0,t_0) \\ \Phi(x_f,u_f,t_f,\nu_f) \quad &:= \phi_f(x_f,u_f,t_f) + \nu_f^T \psi_f(x_f,u_f,t_f), \end{aligned} \tag{2.4}$$

where the $\lambda : [t_0, t_f] \to \mathbb{R}^n$, $\nu_0 \in \mathbb{R}^{N_0}$, $\nu_f \in \mathbb{R}^{N_f}$ and $\mu : [t_0, t_f] \to \mathbb{R}$ are Lagrange multipliers. The number of time derivatives q of S such that u explicitly appears and $S_u^{(q)} \neq 0$ is denoted by $S^{(q)}(x,u,t)$. In order that $S^{(q)}(x,u,t) = 0$ and $S(x,u,t) = 0$ on $[t_1, t_2]$ we also require that the entry conditions

$$N^T(x(t_1), t_1) = [S(x(t_1), t_1), S^{(1)}(x(t_1), t_1), \ldots, S^{(q-1)}(x(t_1), t_1)] = 0 \tag{2.5}$$

be satisfied. See Bryson, Denham and Dreyfus [11] for more details. The Lagrange multipliers $\pi \in \mathbb{R}^q$ will be associated with the constraints in equation (2.5).

An optimal solution of (2.2) and (2.3) must satisfy the the necessary conditions

$$\dot{x} = H_\lambda \tag{2.6}$$

$$\dot{\lambda}^T = -H_x \tag{2.7}$$

$$H_u = 0 \tag{2.8}$$

$$\lambda^T(t_0) = \Xi_x|_{t=t_0} \tag{2.9}$$

$$\lambda^T(t_1^-) = \lambda^T(t_1^+) + \pi^T N_x|_{t=t_1} \tag{2.10}$$

$$H(t_1^-) = H(t_1^+) - \pi^T N_t|_{t=t_1} \tag{2.11}$$

$$\lambda^T(t_f) = \Phi_x|_{t=t_f} \tag{2.12}$$

$$\mu = 0 \ \text{ if } \ S^{(q)}(x,u,t) < 0 \tag{2.13}$$

$$\mu \geq 0 \ \text{ if } \ S^{(q)}(x,u,t) = 0 \tag{2.14}$$

and if the final time t_f is not specified we must include

$$(\Phi_t + H)|_{t_f} = 0. \tag{2.15}$$

Note: The partial derivatives H_x, Ξ_x, and Φ_x are considered row vectors, i.e., $H_x = (\frac{\partial H}{\partial x_1}, \ldots, \frac{\partial H}{\partial x_n})$ and the transpose of the column vector (.) is denoted by $(.)^T$

The necessary conditions result in a multi-point boundary value problem at times t_0, t_1 and t_f involving the differential equations (2.6) and (2.7). At each boundary there will be effectively n constraints derived from the total number of terminal equations minus the free variables.

Utilizing (2.11) and (2.15), we can determine t_1 and t_f. The input on the constrained arc can be found from $S^{(q)}(x,u,t) = 0$. Otherwise we can use equation (2.8). The Lagrange multiplier μ is found from equation (2.13) off the constrained arc and equation (2.8) when on the constrained arc.

Generally, closed form solutions are difficult to find. Numerical methods based on gradient descent or Newton's method are usually invoked to find solutions to multi-point boundary value problems. It is up to the user to provide an initial guess to the free-variables $\nu_0, \nu_f, \pi, t_1,$ and t_f sufficiently close to a solution to guarantee convergence of the method.

The multiple shooting numerical method is advocated by Pesch in [77] and [78]. A thorough description of shooting techniques is provided by Stoer and Burlisch [99]. One advantage of using shooting is that very accurate solutions are obtainable. Potential disadvantages of shooting include a high sensitivity to the initial guess, as well as the need for robust integration techniques over large time intervals since equations (2.6) and (2.7) will likely have both unstable and stable components. An alternative to the shooting method, which is less sensitive to the initial guess, is to replace the ordinary differential equations in equations (2.6) and (2.7) by their finite difference approximations. Press *et al.* discuss so called *relaxation methods* solution methods in [86]. Due to the undesirable convergence rates and computation times, neither of these numerical methods are practical for real-time implementation. In the next section, we will assume that a solution to an optimal control problem can be determined off-line using the above mentioned numerical techniques and stored in a database for on-line use. We will use a trajectory in the database as the initial guess to solve a neighboring optimal control problem with high confidence in real-time.

2.3 Trajectory Generation Using Homotopy

Homotopy can best be shown by a simple example. Suppose that one wishes to obtain the solution of a system of N nonlinear equations in N variables,

$$W(x) = 0, \tag{2.16}$$

where $W : \mathbb{R}^N \to \mathbb{R}^N$ is a smooth mapping. Without a good initial guess $\bar{x}$, an iterative solution to equation (2.16) will often fail. As a possible remedy, one defines a homotopy $V : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ such that

$$V(x, 1) = U(x) \qquad V(x, 0) = W(x),$$

where $U : \mathbb{R}^N \to \mathbb{R}^N$ is a smooth map having known solutions. Typically, one may choose a convex homotopy such that

$$W(x, \epsilon) := \epsilon U(x) + (1 - \epsilon)V(x),$$

and continuously trace an implicitly defined curve from a starting point $(x_1, 1)$ to a solution $(\bar{x}, 0)$. Burlisch in [13] and [99] warns that morphing a simple system not related to the problem to a complex system may not succeed in critical cases.

In the following section, a homotopy method is presented without applying a homotopy to the system dynamics. Chen *et al.* [18] also took a similar approach to singular optimal control problems.

2.3.1 Algorithm Description

It will be assumed that the desired objective is to move the system from a known initial state to a known final state while minimizing a prescribed cost function. The proposed homotopy algorithm will find an optimal trajectory connecting the initial and final state. The central idea of the algorithm is to decompose the operating envelope into regions. For each region a trajectory is computed off-line and the initial and final states and costates are saved into a database for on-line use. The boundaries of each region are determined such that for each trajectory in the region, there exists a homotopy to any other trajectory in the region. The advantages to this algorithm is that only a small subset of all trajectories need to be stored on-board, and every trajectory is optimal with respect to a prescribed cost function.

We tacitly assume that a solution has been found to the trajectory generation problem for some specified system (2.1), with associated cost function in equation (2.2), an initial state constraint $\psi_0(x_0) = 0$, and the desired state $\psi_f(x_f) = 0$. A solution to this problem implies that we have also found ν_0 and ν_f in equation (2.4). Now suppose that we would like to determine the solution with the new boundary conditions $\tilde{x}(t_0)$ and $\tilde{\psi}(x(t_f)) = 0$, where $\tilde{x}(t_0)$ and $\tilde{\psi}(x(t_f))$ are "sufficiently close" to $x(t_0)$ and $\psi(x(t_f)) = 0$, respectively. By "sufficiently close" we mean that there exists a homotopy between $x(t_0)$ and $\psi(x(t_f)) = 0$ and $\tilde{x}(t_0)$ and $\tilde{\psi}(x(t_f))$. The implicit function theorem will dictate whether or not a homotopy exists. To obtain a solution with the new desired goal $\tilde{\psi}(x(t_f))$ and initial state $\tilde{x}(t_0)$, we will embed the known solution into a family of perturbed solutions through the parameter ϵ using a convex homotopy. Define

$$\begin{aligned} \hat{x}(t_0, \epsilon) &:= x(t_0)\epsilon + \tilde{x}(t_0)(1-\epsilon) && (2.17) \\ \hat{\psi}(x(t_f), \epsilon) &:= \psi(x(t_f))\epsilon + \tilde{\psi}(x(t_f))(1-\epsilon), && (2.18) \end{aligned}$$

where $\epsilon \in [0, 1]$ is a perturbation parameter, $\hat{x}(t_0, \epsilon) : \mathbb{R}^n \times [0, 1] \to \mathbb{R}^n$, $\hat{\psi} : \mathbb{R}^n \times [0, 1] \to \mathbb{R}^p$. By making ϵ go to zero, we obtain the solution to the trajectory generation problem of interest.

In principle, we know that the solutions of the differential equations (2.6) and (2.7) are determined by $x(t_0)$ and $\lambda(t_0)$. That is, if we alter ψ_0 and ν_0, we change x and λ for all t. Thus, we may write at the final time t_f:

$$x(t_f) = x(\psi_0, \nu_0, t_f) \qquad \lambda(t_f) = \lambda(\psi_0, \nu_0, t_f). \tag{2.19}$$

For ease of notation, we will write the terminal boundary conditions as

$$G(y, \epsilon) = 0, \tag{2.20}$$

where $y = (\nu_0, \nu_f)$ and G is an $(n + N_0)$ vector-valued functional.

Suppose we are given a solution (z_0, ϵ_0) of (2.20). Using homotopy, we want to find a solution $z(\epsilon_0 + \Delta\epsilon)$ for a small perturbation $\Delta\epsilon$. The implicit function theorem [50] gives conditions which one can solve (2.20) in some neighborhood of (z_0, ϵ_0):

a. $G(z_0, \epsilon_0) = 0$ for some z_0 and ϵ_0;

b. $G_z(z_0, \epsilon_0)$ is nonsingular;

c. $G(z, \epsilon)$ is continuous on some neighborhood of z_0 and ϵ_0.

Condition (b) implies that the flight envelope has been properly decomposed into regions, such that there are no singularities between trajectories for any given region. Note that Keller [50] proposes several numerical methods to get around singular points.

By taking the total derivative of (2.20) with respect to ϵ and solving for $\frac{dz}{d\epsilon}$, we obtain the following differentiable equation:

$$\frac{dz}{d\epsilon} = -G_z^{-1}(z,\epsilon)G_\epsilon(z,\epsilon). \tag{2.21}$$

In order to solve equation (2.21), it is necessary to determine the Jacobians $G_z(z,\epsilon)$ and $G_\epsilon(z,\epsilon)$. This can be accomplished numerically using a finite-difference approximation or through an integration of a set of ordinary differential equations.

To find $G_z(z,\epsilon)$ and $G_\epsilon(z,\epsilon)$ numerically, we can use a numerical approximation to these Jacobians. For example, we can use a forward difference approximation

$$\frac{\partial G}{\partial z_i} = \frac{G(z+\delta_i e_i,\epsilon) - G(z,\epsilon)}{\delta_i} \qquad \frac{\partial G}{\partial \epsilon} = \frac{G(z,\epsilon+\Delta\epsilon) - G(z,\epsilon)}{\Delta\epsilon}$$

with e_i the ith unit vector and δ_i a scalar.

We can also get an expression for $G_z(z,\epsilon)$ by integrating a system of ordinary differential equations. This method is recommended since its results are more accurate than the finite difference method in approximating $G_z(z,\epsilon)$.

Differentiating both sides of equations (2.6) and (2.7) with respect to z, while holding ϵ constant, and defining two new variables

$$\xi = \frac{\partial x}{\partial x_0},\ \eta = \frac{\partial \lambda}{\partial \lambda_0}, \quad \xi,\eta \in \mathbb{R}^{n\times(n)},$$

we obtain the differential equations

$$\dot{\xi} = H_{\lambda x}\cdot\xi + H_{\lambda\lambda}\cdot\eta \tag{2.22}$$

$$\dot{\eta} = H_{xx}\cdot\xi + H_{x\lambda}\cdot\eta. \tag{2.23}$$

Changing z, while holding ϵ constant, we have $\frac{\partial x}{\partial x_0}(t_0) = 0 \in \mathbb{R}^{n\times n}, \frac{\partial \lambda}{\partial \lambda_0} = I_6 \in \mathbb{R}^{n\times n}$, $\frac{\partial x}{\partial \lambda_0} = 0 \in \mathbb{R}^{n\times n}$, and $\frac{\partial \lambda}{\partial x_0} = 0 \in \mathbb{R}^{n\times n}$. Now we can find $G_z(z,\epsilon)$ by integrating the $\eta(t)$ and $\xi(t)$ dynamics forward in time:

$$G_z(z,\epsilon) = G_z(\eta(t_f), \xi(t_f), \epsilon). \tag{2.24}$$

In an analogous manner, the following differential equations are derived which we will use to calculate the Jacobian $G_\epsilon(z,\epsilon)$ (holding z_0 constant). Defining

$$\zeta = \frac{\partial x}{\partial \epsilon},\ \chi = \frac{\partial \lambda}{\partial \epsilon}, \quad \zeta, \chi \in \mathbb{R}^n$$

we obtain

$$\dot{\zeta} = H_{\lambda x} \cdot \zeta + H_{\lambda\lambda} \cdot \chi + H_{\lambda\epsilon} \tag{2.25}$$

$$\dot{\chi} = H_{xx} \cdot \zeta + H_{x\lambda} \cdot \chi + H_{x\epsilon} \tag{2.26}$$

and observe that if we change ϵ, but hold z constant, we have, referring to equation (2.17), $\zeta(t_0) = x(t_0) - \tilde{x}(t_0)$ and $\chi(t_0) = 0$. Now we can find $G_\epsilon(z,\epsilon)$ by integrating the $\zeta(t)$ and $\chi(t)$ dynamics forward in time:

$$G_\epsilon(z,\epsilon) = G_\epsilon(\zeta(t_f), \chi(t_f), \epsilon). \tag{2.27}$$

2.3.2 Ducted Fan Example

An example of the use of the homotopy technique will be illustrated using the planar ducted fan. Figure 2.1 depicts the coordinate systems and conventions used for the ducted fan. Writing Newton's equations about O_w we have

$$\begin{aligned} m\ddot{x} &= F_{X_b}\cos\theta + F_{Z_b}\sin\theta && (2.28)\\ m\ddot{z} &= -F_{X_b}\sin\theta + F_{Z_b}\cos\theta + mg && (2.29)\\ J\ddot{\theta} &= F_{Z_b} r_p. && (2.30) \end{aligned}$$

The numerical values of the constants used in this example are the following:

$$m = 2.2\ kg \qquad mg = 4\ N \qquad r_p = .2\ m \qquad J = 0.05\ kgm^2. \tag{2.31}$$

The optimal cost with free final time is

$$\min_u J \quad \equiv \quad \int_0^T (R_T + R_1 F_{X_b}^2 + R_2 F_{Z_b}^2) dt, \tag{2.32}$$

with input constraints $-5 \leq F_{Z_b} \leq 5$ N, $0 \leq F_{X_b} \leq 17$ N, as well as constraints on the initial and final state. The problem can be described as the following: Minimize a balance between time and energy subject to initial and final time constraints as well as a trajectory constraint on the input.

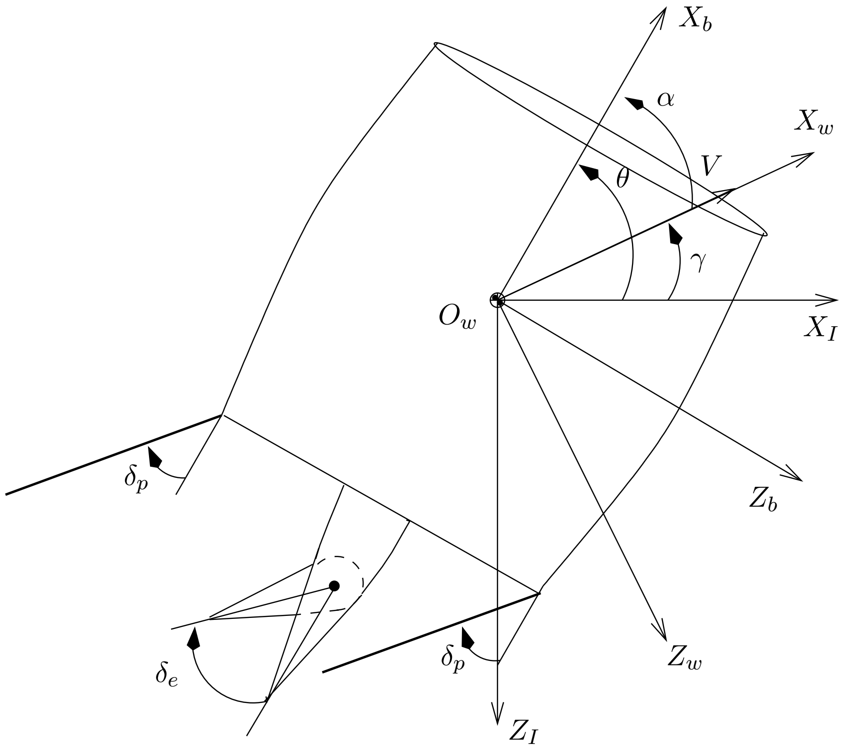

Figure 2.1: UCAV coordinate system conventions.

In this example, the integration technique used was a predictor-corrector algorithm described by Allgower in [3].

For the first example we computed off-line a trajectory using the optimal trajectory solver RIOTS [90] for the initial conditions $x(t_0) = (0\ 0\ 0\ 0\ \pi/2\ 0)^T$ and the final conditions $\hat{\psi}(x(t_f)) = \psi(x(t_f)) = x(t_f) - (1\ 0\ -1\ 0\ \pi/2\ 0)^T$. The cost function weights were chosen as $R_T = 4$, $R_1 = 1$, and $R_2 = 2$. It is desired to deform the initial condition to $\tilde{x}(t_0) = (-1\ 0\ -1\ 0\ \pi/2\ 0)^T$. Note that the final unknown time is a free variable. The predictor-corrector algorithm was coded in Matlab and consumed approximately 11 minutes of CPU time on a Sun Ultra 30. The results of the simulation showed that the homotopy could be performed, but the computation time was prohibitive.

The same setup was used for the second example, except that $R_T = 200$ and $\tilde{x}(t_0) = (-.8\ 0\ -.8\ 0\ \pi/2\ 0)^T$. The difference between the two examples is that, in the second, there is significant weight on the time. The results of this simulation are shown in Figure 2.2. For this example, it took approximately 38 minutes of CPU time.

At $\epsilon = .35$ the trajectory suddenly changes. This can be explained by the significant positive to negative change of Lagrange multiplier $\lambda_3(t_0)$. The sign of the determinant of $G_z(z, \epsilon)$ indicates a singularity. However, the implementation of the algorithm is robust enough so that we still proceed with the homotopy in effect, jumping over the singularity. Note that this singularity is not explicitly occurring the the time domain since the homotopy parameter does not enter explicitly in the state or costate equations. It is the significant change in $\lambda_3(t_0)$ that is causing the large change in the dynamical behavior seen in Figure 2.2.

The unexpected singularity illustrates the necessity of an off-line study to find trajectories such that we can steer around singularities. The nonsingularity of $\det(G_z(z, \epsilon)) \neq 0$ must be off-line. If a singularity occurred, a modification of the cost function would be of first consideration. If a modification of the cost function does not work, one could break the problem into separate homotopy problems: $x(t_0) \rightarrow \bar{x}(t_0)$ and $\psi(x(t_f)) \rightarrow \bar{\psi}(x(t_f))$ and then $\bar{x}(t_0) \rightarrow \hat{x}(t_0)$ and $\bar{\psi}(x(t_f)) \rightarrow \hat{\psi}(x(t_f))$. Updating the algorithm presented in the previous subsection by using the techniques given in Allgower [3] or Keller [50] to handle singularities,

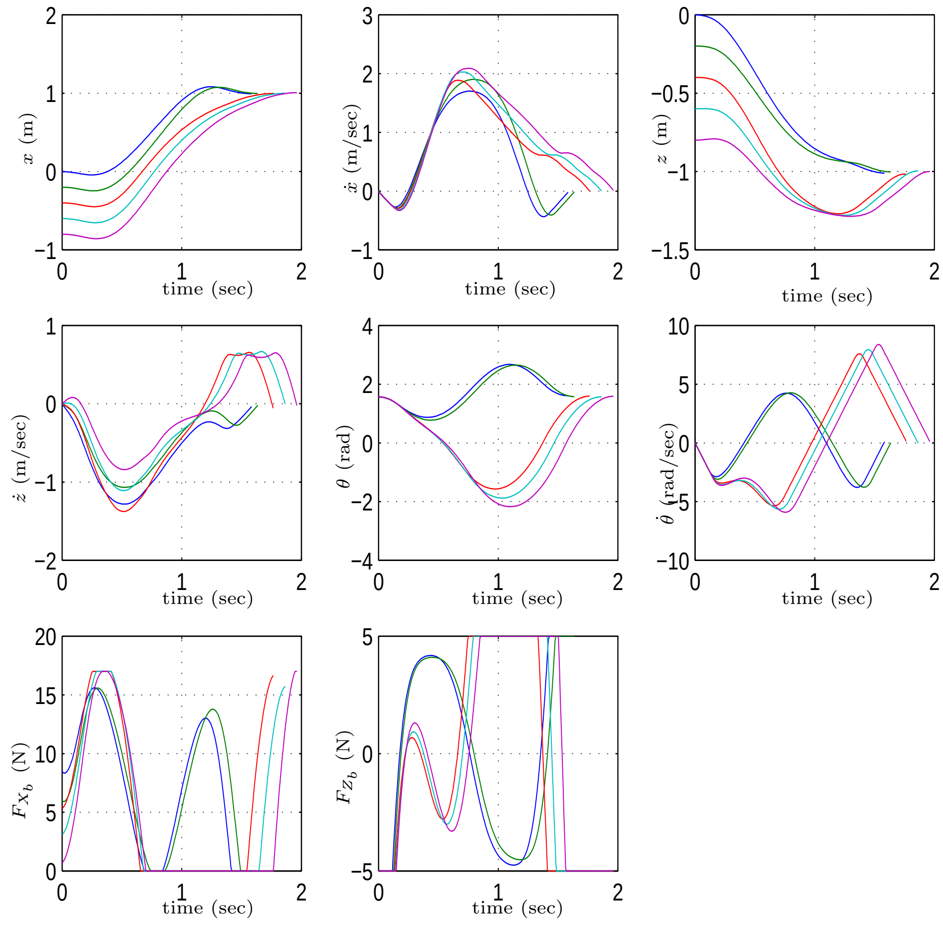

Figure 2.2: Ducted fan trajectory generation homotopy example: x: $0 \rightarrow -.8$ and z: $0 \rightarrow -.8$ with large time weight.

would be a last resort.

In general, another fundamental problem with this technique is sensitivity. We are using a form of shooting to find $G_z(z, \epsilon)$ by integrating a system of ordinary differential equation that have both stable and unstable components. Integrating such a system of ODE's over a significant time span can become ill-conditioned. This results in the ODE solver proceeding very slowly, producing large CPU times. This sensitivity is due the nature of the necessary conditions of optimal control. Multiple shooting may help mitigate some of these integration problems.

2.4 Nonlinear Programming Techniques

The problem of finding a local minimizer $x \in \mathbb{R}^n$ for a nonlinear function $F(x)$ subject to a set of nonlinear constraints $c \geq 0$, where $c(x) \in \mathbb{R}^n$, is a *nonlinear constrained* optimization problem. All the problems of interest to be solved in this thesis can be generalized into the form

$$\begin{aligned} &\min_x F(x) \\ &\text{subject to} \quad c(x) \geq 0. \end{aligned} \tag{2.33}$$

Optimization problems of the form (2.33) can be a very difficult problem to solve. For instance, it is NP-hard in the traditional complexity model [105]. Algorithms to solve (2.33) may take many iterations and function evaluations. A promising global optimization technique, based on surrogate functions, is given by Dennis *et al.* [23]. Global optimization of (2.33) is a difficult problem and an open area of reserach . In this thesis, we will concentrate on using the well understood numerical techniques that will find local minima of (2.33).

Sequential Quadratic Programming (SQP)and Interior Point Methods (IPM) are popular classes of methods considered to be effective and reliable method for locally solving Equation (2.33). At each iteration of an SQP method, one solves a quadratic program (QP) subproblem that models (2.33) locally at the current iterate. The solution to the QP is used as a search direction to determine the next iterate. Eliminating inequality constraints by adding slack variables and incorporating them into a logarithmic barrier function in the objective function, (2.33) is transformed into

$$\begin{aligned} &\min_x F(x) - \mu \sum_{i=1}^{m} \log s_i \\ &\text{subject to} \quad c(x) - s = 0. \end{aligned} \tag{2.34}$$

Interior point refers to the fact that the slack variables are required to remain strictly positive throughout the optimization.

Successful algorithms need to be both efficient and robust. Under reasonable assumptions, a robust algorithm must be globally convergent (convergent from any starting point) and able to solve, in practice, both well-conditioned and ill-conditioned problems. NPSOL [38], SNOPT [39], CFSQP [57], KNITRO [110], and LOQO [104] are the nonlinear programming solvers we consider in this thesis.

2.4.1 Optimality Conditions

Definition 2.1. A point x^* is a local minimizer of (2.33) if

1. $c(x^*) \geq 0$;
2. there exists a $\delta > 0$ such that $F(x) > F(x^*)$ for all x satisfying

$$||x - x^*|| \leq \delta \text{ and } c(x) \geq 0.$$

The term *active constraint* will be used to designate a constraint $c_i(x) \geq 0$ if $c_i(x) = 0$. If $c_i(x) > 0$, a constraint is considered *inactive*.

The Lagrangian $\mathcal{L}(x, \lambda) \equiv F(x) - \lambda^T c(x)$ is a scalar function with the n variables x and the m variables λ which correspond to the Lagrange multipliers. The Lagrangian is used to express first-order and second-order *optimality conditions* for a local minimizer.

Let $\hat{J}(x^*)$ denote the set of gradients of the active constraints at x^*. A constraint qualification ensures the existence of the Lagrange multipliers. One such constraint qualification is that the gradients of the active constraints at x^* be linear independent, i.e., that matrix $\hat{J}(x^*)$ should have full row rank. A point that satisfies this particular constraint qualification is known as a *regular point*.

Necessary conditions for x^* to be a local minimizer are that there exist Lagrange multipliers λ^* such that

$$c(x^*) \geq 0 \tag{2.35}$$

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla F(x^*) - \hat{J}(x^*)^T \lambda^* = 0 \tag{2.36}$$

$$\lambda^* \geq 0. \tag{2.37}$$

These conditions are known as the *first-order Karush-Kuhn-Tucker (KKT) necessary optimality conditions*. Note that x^* is a stationary point of $\nabla_x(x^*, \lambda^*)$ but not necessarily an unconstrained minimizer of the Lagrangian.

Suppose that x^* is a local minimizer and a regular point. Then, x^* is a KKT point and

$$Z^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) Z \text{ is positive semidefinite,} \tag{2.38}$$

where Z is a basis for the nullspace of $\hat{J}$. These conditions are known as the *second-order KKT conditions*. If we replace (2.37) and (2.38) by

$$\begin{array}{rl} \lambda^* & > 0 \\ Z^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) Z & \text{is positive definite,} \end{array} \tag{2.39}$$

we obtain sufficient conditions for optimality.

A key challenge to developing a fast algorithm for solving this problem is to find an accurate approximation to the Hessian of the Lagrangian $\nabla^2_{xx} \mathcal{L}(x, \lambda)$, the second derivative that reflects the curvature of the objective and constraints.

The common approach to approximating $\nabla^2_{xx} \mathcal{L}(x, \lambda)$ has been to follow quasi-Newton techniques for unconstrained optimization. A single, positive-definite approximation matrix is maintained using the *Broyden-Fletcher-Goldfarb-Shanno*

(BFGS) quasi-Newton update, and typically the dependence of the Lagrange multipliers is ignored. See Gill *et al.* [40] for a complete discussion on the BFGS method. The BFGS direct approximation may be poor, even for ideal problems, but has been used in CFSQP and NPSOL successfully. KNITRO and LOQO use analytical Hessians of the constraint and objective to form the approximation to the Hessian of the Lagrangian.

2.4.2 Techniques for Global Convergence

Nearly all techniques for nonlinear programming are iterative, producing a sequence of subproblems related in some way to the original problem. Newton methods have rapid local convergence rates, but fail to converge from all starting points. Gradient descent methods converge from nearly any starting point but have poor local convergence properties. *Line-Search* methods are one means of ensuring global convergence while attempting to maintain fast local convergence. Line-Search methods limit the size of the step taken from the current point to the next iterate. Such methods generate a sequence of iterates of the form

$$x_{k+1} = x_k + \alpha p,$$

where p is the search direction obtained from the subproblem, and α is a positive scalar *steplength*. For unconstrained minimization, or if feasibility is maintained for the constraints as with CFSQP, the best steplength is one that minimizes the objective function $F(x_{k+1})$. However, determining a minimizer along p is an iterative process and frequently time consuming. Typically, x is determined by a finite process that ensures a reduction in $F(x)$. The nonlinear programming code NPSOL and LOQO use line-search methods. See Fletcher [31] for an overview of line search methods.

2.4.3 Trust-Region Methods

The main alternative to line-search methods are *trust-region* methods. The motivation behind the trust-region approach is that the minimum of the local quadratic model should be accepted so long as the model adequately reflects the behavior of the function under consideration. Trust-region methods choose a radius δ and determine x_{k+1} that is the global minimizer of a model of the function subject to $||x_{k+1} - x_k|| \leq \delta$. The nonlinear programming code KNITRO uses trust region methods.

2.4.4 Merit Functions

Regardless of whether a line-search or trust-region method is used, when feasibility of the iterates is not maintained, it can be difficult to guide the choice of steplength. For problems with linear constraints, it is straightforward to maintain feasibility at every iteration. However, when even a single constraint is nonlinear, maintaining feasibility at every iteration becomes difficult. For infeasible iterates, it is not immediately obvious how to choose the step length; we would like the next iterate to minimize the objective function, but we would also like to reduce the infeasibilities of the constraints. Merit functions are used to guide the improvement of the feasibility and the optimum at the same time. Since NPSOL, LOQO, and KNITRO use merit functions, they are considered infeasible methods. CFSQP is a feasible method; thus, there is no need for a merit function. If we assume, for simplicity that all constraints are equalities of the form $c(x) = 0$, two popular merit functions are the l_1 merit function

$$M(x) = F(x) + \rho||c(x)||_1$$

and the augmented Lagrangian

$$M(x, \lambda) = F(x) - \lambda^T c(x) + \frac{\rho}{2} c(x)^T c(x)$$

where ρ is a penalty parameter and λ is a set of Lagrange multiplier estimates.

2.4.5 Simple Infeasible Nonlinear Programming Algorithm Using a Line-search

Algorithm 2.1. Choose an initial guess for the optimization variables x_0 and the Lagrange multipliers λ_0. Set $k = 0$, while the KKT conditions are not satisfied:

1. Set up a subproblem to compute a search directions $\bar{p}$ and $\bar{\lambda}$ for both the decision variables and Lagrange multipliers.

2. Compute step length α_k that reduces the merit function.

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \alpha_k \begin{pmatrix} \bar{p} \\ \bar{\lambda} \end{pmatrix}$$

3. Update $c(x_k)$, $F(x_k)$, $\nabla F(x_k)$,$\nabla c(x_k)$.

4. Update H_k, the approximation to the Hessian of the Lagrangian.

5. Set $k = k + 1$.

This algorithm is the basic one used in NPSOL and LOQO. The difference between the two algorithms is in the computation of the subproblem. NPSOL finds the search direction by solving a sequence of QP problems with an active set method. LOQO uses barriers for the constraints so it only has one subproblem iteration. LOQO solves directly for the Newton step $(\bar{p}, \bar{\lambda})$ from (x_0, λ_0) to (x^*, λ^*) by the following:

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x_0, \lambda_0) & G(x_0)^T \\ G(x_0) & 0 \end{pmatrix} \begin{pmatrix} \bar{p} \\ -\bar{\lambda} \end{pmatrix} = - \begin{pmatrix} g(x_0) - G(x_0)^T\lambda_0 \\ c(x_0) \end{pmatrix}. \qquad (2.40)$$

Similarly, NPSOL solves quadratic program

$$\min_p g(x_0)^T p + \frac{1}{2} p^T \nabla^2_{xx} \mathcal{L}(x_0, \lambda_0) p \tag{2.41}$$

$$\text{subject to} \quad G(x_0)p - c(x_0) = 0. \tag{2.42}$$

2.4.6 Active Set Methods

The active set method starts with a feasible point p_0 and a working set of variables. The active set method proceeds to move to a constrained stationary point of the QP by holding a set of variables constant and temporarily ignoring the other bounds. If the solution (the new search direction p) to the QP program (2.42) is feasible with respect to the bounds, a full step is taken. Otherwise, the maximum feasible step is taken along the search direction and a bound is added to the working set. This sequence repeats until a full step is taken. At the stationary point, if for any bound there exists a negative Lagrange multiplier, the associate bound is dropped from the working set and the procedure starts over. If all the multipliers are positive, the algorithm stops. NPSOL and CFSQP are active set methods.

It looks promising that interior point methods may show a significant reduction in the total number of iterations compared to active set methods. The subproblem may take longer than any one QP subproblem computation, but the time of this one iteration will be faster than the *total* QP computational time. Active set methods take a combinatorial number of iterations for the subproblem.

2.5 Numerical Solutions of Optimal Control Problems Using Nonlinear Programming

2.5.1 Direct Methods of Solution Using Nonlinear Programming

We can deduce from Section 2.2 that the trajectory generation problem can be formulated in terms of an optimal control problem. In general, the optimal control problem can be solved by either indirect or direct methods.

Indirect methods are based on the calculus of variations and the maximum principle. In the direct approach, the optimal control problem is transformed into a nonlinear programming problem (NLP). In Section 2.2 an indirect method was used to solve the optimal control problem. For an overview of the indirect and direct methods see Betts [7, 5] and Stryk and Burlisch [109]. In this section we will concentrate on the direct method of solution.

Direct methods are generally more robust to the initial solution guess than indirect methods. This is a result of not having to explicitly solve the necessary conditions of optimal control, which are frequently ill-conditioned. In addition, unlike indirect methods, direct methods do not require an a priori specification of the switching structure when inequality state constraints are present. However, it appears that the computational requirements of direct methods are at least that of indirect methods.

The collocation method of [42] and adjoint method [82] are the methods of transcription that are most relevant to the trajectory generation problem. Sequential quadratic programming, presented in Gill *et al* ([40] and [56] *et al.*,is the technique we will use to solve the nonlinear programming problems presented in this thesis.

The nonlinear programming problem can be stated as the following:

$$\begin{aligned} &\min_{y\in\mathbb{R}^M} F(y) \\ &\text{subject to } l_j \leq c_j(y) \leq u_j \quad j = 1, \ldots, m_N, \end{aligned} \tag{2.43}$$

where l_j and $u_j \in \mathbb{R}$,the constraint function $c_j : \mathbb{R}^n \to \mathbb{R}$ is at least C^2, and the cost function $F : \mathbb{R}^n \to \mathbb{R}$ is also at least C^2. We will rely on commercially available nonlinear programming packages. It will be required that the nonlinear programming problem be provided, not only the cost function and the constraints, but also gradients of cost function and constraints with respect to the decision variables y.

The rest of this section we will discuss the conversion techniques from the optimal control problem to the nonlinear programming problem.

2.5.2 Transcription Techniques from the Optimal Control Problem to the Nonlinear Programming Problem

Collocation

A reliable method to convert an optimal control problem to a nonlinear programming problem is collocation. For an overview of collocation methods see Stryk [108].

The first step in collocation process is to break the time domain into smaller intervals

$$t_0 < t_1 < \ldots < t_N = t_f. \tag{2.44}$$

The nonlinear programming decision variables then become the values of the state and the control at the grid points, namely,

$$y = (u_0, x_1, u_1, x_2, u_2, \ldots, x_N, u_N). \tag{2.45}$$

The key notion of collocation methods is to replace the original system in equation (2.1) with a set of defect constraints $\zeta_i = 0$, which are imposed on each interval of discretization. An expression of ζ_i can be derived based on the choice of numerical integration.

If we assume, for illustration, that there is also a state variable equality constraint $S(x(t)) = 0$, initial condition $\psi_0(x(t_0)) = 0$, and final output constraint

$\psi_f(x(t_f)) = 0$, then the complete set of nonlinear programming constraints are

$$c(y) = \begin{bmatrix} \psi_0(x(t_0)) \\ S(x_0) \\ \zeta_0 \\ S(x_1) \\ \vdots \\ S(x_{N-1}) \\ \zeta_{N-1} \\ S(x_N) \\ \psi(x(t_N)) \end{bmatrix} = 0. \tag{2.46}$$

Suppose, for simplicity, that we want to choose the control $u(t)$ to minimize the cost function $J = \phi(x(t_f))$, the nonlinear programming objective function will become

$$F(y) = \phi(x_N). \tag{2.47}$$

The gradients of the cost function and the constraints in equation (2.43) with respect to decision variables in equation (2.45), easily follow. It is apparent that the nonlinear programming problem that results from this formulation is very large, rendering a real-time implementation difficult.

Stryk *et al.* in [109] show how direct methods are related to indirect methods.

This result is particularly useful when one wants to use the direct method to estimate constrained arcs as well as provide an initial guess to an indirect method.

Adjoints

The Adjoint method is a direct method that uses a combination of nonlinear programming and shooting. RIOTS [90] and COOPT [93] use the Adjoint method. RIOTS uses single shooting while COOPT uses modified multiple shooting.

In contrast to the collocation method, the adjoint method has significantly less decision variables. In fact, there are as many decision variables $u(t_i)$ as there are collocation points N. The trade off is that an integration has to be performed backward in time on an adjoint system for each constraint. Bryson and Ho [12] state that the numerical integration required to find the adjoints is quite stable numerically since integration is carried out in backward time. The assumption is that the system dynamics are stable in forward time. Since there are many integration schemes possible, we will sketch the formulation of the gradients in continuous time.

By taking the total differential of equation (2.2), applying the Adjoint Lemma [49] and constructing a $\delta u(t)$ history such that the cost function is decreasing, the gradient of the cost function with respect to the decision variables is

$$\begin{aligned} \nabla_u J(u) &= p_c^T f_u + L_u \\ \dot{p}_c &= -f_x^T p_c - L_x^T \qquad p_c(t_f) = \frac{\partial \phi(x(t_f))}{\partial x(t_f)}, \end{aligned} \tag{2.48}$$

where $p_c \in \mathbb{R}^n$. Likewise, the gradient of the endpoint equality constraint is

$$\begin{aligned} \nabla_u \psi(x(t_f)) &= p_{ee}^T f_u \\ \dot{p}_{ee} &= -f_x^T p_{ee} \qquad p_{ee}(t_f) = \frac{\partial \psi(x(t_f))}{\partial x(t_f)}, \end{aligned} \tag{2.49}$$

where $p_{ee} \in \mathbb{R}^n$. Finally, the gradient of a state inequality constraint $S(x(t)) < 0$, $S : \mathbb{R}^n \rightarrow \mathbb{R}$ which is active between t_a and t_b ($t_0 < t_a < t_b < t_f$) is

$$\nabla_u S(t) = \begin{cases} 0 & \text{if } t_s < t_a \\ p_{ic}^T f_u & \text{if } t_s \in (t_a, t_b) \\ 0 & \text{if } \ t_s \geq t_b, \end{cases} \tag{2.50}$$

$$\dot{p}_{ic} = -f_x^T p_{ic} \qquad p_{ic}(t_s) = \frac{\partial S(x(t_s))}{\partial x(t_s)} \qquad t_a \leq t_s \leq t_b$$

where $p_{ic} \in \mathbb{R}^n$. Expressions for other constraints, such as endpoint inequality constraints and trajectory equality constraints, can be found in a similar manner. Once the gradients are known, there is enough information to solve the nonlinear programming problems in (2.43).

2.6 Trajectory Generation Using Feedback Linearization

The system under consideration in this section is

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{2.51}$$

$x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^m$. The non-affine system in equation (2.1) can be transformed into this form by adding integrators to all inputs.

2.6.1 Mathematical Background

We define $\mathrm{ad}_f g_j = [f, g_j]$ as the *Lie Bracket* of the smooth vector fields f and g_j and

$$[f,g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g = L_g f - L_f g.$$

where g_j is the jth column of $g(x)$ in equation (2.51). We note that $\mathrm{ad}_f^0 = g$ and $\mathrm{ad}_f^k g = [f, \mathrm{ad}_f^{k-1} g]$ for all $k \geq 1$.

Define the distributions

$$\begin{aligned} \Delta_0 &= \mathrm{span}\{g_j, 1 \leq j \leq m\} \\ \Delta_i &= \mathrm{span}\{\mathrm{ad}_f^i g_j, 0 \leq l \leq i,\ 1 \leq j \leq m\} \ \ i > 0, \end{aligned}$$

which have the recursion properties

$$\Delta_{i+1} = \Delta_i = \mathrm{ad}_f^{i+1} \Delta_0 = \Delta_i + \mathrm{ad}_f \Delta_i$$

where $\mathrm{ad}_f \Delta = \mathrm{span}\{\mathrm{ad}_f Y, Y \in \Delta\}$. (By definition $\Delta_0 \subset \Delta_1 \subset \cdots \Delta_i$.)

The Lie derivative of a function h with respect to a vector field f

$$L_f h = \frac{\partial h}{\partial x} f \tag{2.52}$$

is the scalar function corresponding to the directional derivative of h along the direction of f. The k-th Lie derivative of h with respect to f is defined recursively by the function

$$L_f^k h = L_f L_f^{k-1} h. \tag{2.53}$$

A distribution Δ is involutive if and only if, given any pair of vector fields g_1 and g_2 in Δ, their Lie Bracket $[g_1, g_2]$ belongs to Δ.

2.6.2 Classical Feedback Linearization

We will make use of the notion of observability in the sequel. A system 2.51 is said to be observable if

Theorem 2.1. Observability (Sontag [97])

$$dim(Span(h_1, ad_f h_1, \ldots, ad_f^{k_1-1} h_1, \ldots, (h_m, ad_f h_m, \ldots, ad_f^{k_m-1} h_m)) = n$$

for some k_i *such that* $\sum_{1=1}^{m} k_i = n$.

Definition 2.2 (Feedback linearizability). The nonlinear system in (2.51) is *feedback linearizable* if there is a dynamic feedback

$$\begin{aligned} \dot{z} &= \alpha(x, z, v) \qquad & z \in \mathbb{R}^p \\ u &= \beta(x, z, v) \qquad & v \in \mathbb{R}^m \end{aligned} \tag{2.54}$$

and new coordinates $\xi = \phi(x, z)$ and $\eta = \psi(x, z, v)$ such that in the new coordinates the system has the form

$$\dot{\xi} = A\xi + B\eta, \tag{2.55}$$

where $A \in \mathbb{R}^{(n+p)\times(n+p)}$ and $B \in \mathbb{R}^{(n+p)\times m}$. If $\dim z = 0$, then we say the system is *static* feedback linearizable.

The system (2.51) has a well defined *vector relative degree* if there exists a vector of integers $r = (r_1, \ldots, r_m)$, such that

$$B_{ij}(x) = L_{g_j} L_f^{r_i - 1} h_i(x) \tag{2.56}$$

satisfies $\det B(x_0) \neq 0$ and $L_{g_j} L_f^k h_i(x) = 0$ for every $1 \leq i \leq m$, $1 \leq j \leq m$, $k < r_i - 1$ for all x in a neighborhood of x_0.

If the system has well defined vector relative degree, the system can be partially linearized. To partially linearize the system, differentiate the outputs z_i until the at least one u_j explicitly appears. Following this procedure for each output we obtain the following:

$$\begin{pmatrix} y_1^{(r_1)} \\ \vdots \\ y_m^{(r_m)} \end{pmatrix} = \begin{pmatrix} L_f^{r_1} h_1 \\ \vdots \\ L_f^{r_m} h_m \end{pmatrix} + B(x)u := \alpha(x) + B(x)u. \tag{2.57}$$

Define $\xi_k^i = z_i^{(k-1)}$ for $i = 1, \ldots, m$ and $k = 1, \ldots, r_i$,and denote

$$\begin{aligned} \xi &= (\xi_1^1, \xi_2^1, \ldots, \xi_{r_1}^1, \xi_1^2, \ldots, \xi_{r_2}^2, \ldots, \xi_m^{r_m - 1}) \\ &= (z_1, \dot{z}_1, \ldots, z_1^{(r_1 - 1)}, z_2, \ldots, z_2^{(r_2 - 1)}, \ldots, z_m^{(r_m - 1)}). \end{aligned} \tag{2.58}$$

Selecting η to be a $n - \sum_i r_i$ dimensional function such that the coordinate transformation $(\xi, \eta) = \Psi(x)$ is a diffeomorphism with $\Psi(0) = 0$,

$$\begin{aligned} u &:= B(\xi, \eta)^{-1}[v - \alpha(x)] \\ x &:= \Psi^{-1}(\xi, \eta) \end{aligned} \tag{2.59}$$

the system dynamics in (2.51) can be put in the form

$$\begin{aligned}
\dot{\xi}_1^1 &= \xi_1^2 \\
&\vdots \\
\dot{\xi}_1^{r_1} &= v_i \\
&\vdots \\
\dot{\xi}_m^1 &= \xi_m^2 \\
&\vdots \\
\dot{\xi}_m^{r_m} &= v_m \\
\dot{\eta} &= q_1(\eta, \xi) + q_2(\eta, \xi)u.
\end{aligned} \tag{2.60}$$

If $\sum r_i = n$, the system (2.51) is *feedback linearizable* by static state feedback.

The internal dynamics of the system that result in finding the control input that maintains the outputs to zero

$$\dot{\eta} = q_1(\eta, 0) + q_2(\eta, 0)u, \tag{2.61}$$

are called the zero dynamics.

Asymptotically stable zero dynamics are called *minimum phase*, else they are considered *non-minimum phase.* Unstable zero dynamics cannot be ignored in constrained trajectory generation. In the case that the relative degree is not well defined, the *Dynamic Extension Algorithm* can be used to maximize the vector relative degree. See Isidori [46] or [74] for a thorough discussion of the *Dynamic Extension Algorithm* .

The question arises whether or not there exists a set of outputs, such that the system can be linearized. The following theorem states a necessary and sufficient condition.

Theorem 2.2. (Isidori [46]) *The system (2.51) is locally static feedback linearizable if and only if, in U_0, a neighborhood of the origin in $\mathbb{R}^n$:*

1. *Δ_i is an involutive distribution of constant rank for every $i \geq 0$;*

2. *rank $\Delta_{n-1} = n$.*

Remark 2.1. The theorem implies that there exists outputs $\lambda_1(x), \ldots, \lambda_m(x)$ such that, relative to the outputs λ_i the system has $\sum r_i = n$ in a neighborhood or the origin. The functions $\lambda_1(x), \ldots, \lambda_m(x)$ can be constructed by solving a set of first order partial differential equations of the form

$$L_{g_j} L_f^k \lambda_i(x) = 0 \text{ for all } 0 \leq k \leq r_i - 1, 1 \leq j \leq m.$$

It has been shown by Charlet *et al.* in [16] that (2.51) with $m = 1$ local static feedback linearization is equivalent to dynamic feedback linearization. For $m > 1$, necessary and sufficient conditions do not exist for dynamic feedback linearization. Thus, the difficulty of finding linearizing outputs for multi-input-multi-output systems is compounded by the fact that the system may not be static feedback linearizable, but still dynamically feedback linearizable.

Example 2.1. Consider the dynamics of the planar ducted fan shown in Figure 2.1 with unity mass, inertia, distance from the center of mass to the F_{Z_b} application point, and gravitational force.

$$\ddot{x} = F_{X_b} \cos\theta + F_{Z_b} \sin\theta \tag{2.62}$$

$$\ddot{z} = -F_{X_b} \sin\theta + F_{Z_b} \cos\theta + 1 \tag{2.63}$$

$$\ddot{\theta} = F_{Z_b}. \tag{2.64}$$

Choosing outputs to be

$$z_1 = x + \cos(\theta) \text{ and } z_2 = z - \sin(\theta), \tag{2.65}$$

we obtain the decoupling matrix:

$$B(x) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix}. \tag{2.66}$$

The vector relative degree is $(2, 2)$. Since $B(x)$ is singular, the system is a candidate for dynamic feedback linearization. The dynamic extension algorithm in Isidori [46] can be applied to maximize the vector relative degree with a dynamic compensator and hopefully feedback linearize the system. Applying the dynamic extension algorithm, we obtain the coordinate transformation,

$$\sigma \mapsto F_{X_b} - \dot{\theta}^2 \tag{2.67}$$

and the dynamic compensator,

$$\ddot{\sigma} = v_1 \tag{2.68}$$

with the new input v_1. The decoupling matrix for the extended system

$$B_1(x) = \begin{bmatrix} \cos\theta & -\sin\theta\sigma \\ -\sin\theta & -\cos\theta\sigma \end{bmatrix} \tag{2.69}$$

is nonsingular and has vector relative degree $(4, 4)$. Therefore, the system in equation (2.62) is feedback linearizable, with the outputs in equation (2.65), change of coordinates in equation (2.67), and the dynamic compensator in equation (2.68). The attitude of the ducted fan can be found from the following expression:

$$\tan\theta = \frac{(mg - m\ddot{z}_2)}{m\ddot{z}_1}.$$

The other dependent variables can easily be found once θ is known.

Closely related to dynamic feedback linearization is differential flatness. Fliess and coworkers in [32] introduced the notion of endogenous feedback, which is essentially a dynamic feedback of the form (2.54), but with the added requirement that the compensator can be uniquely determined as function of x, u and a finite number of derivatives of u. They have shown that dynamic feedback linearization via endogenous feedback is equivalent to differential flatness.

2.6.3 Devasia-Chen-Paden Non-minimum Phase Zero-Dynamics Algorithm

The Devasia-Chen-Paden provides a way to generate trajectories for a class of systems with unstable zero dynamics. For this algorithm to work, the system must have a well-defined relative degree and its zero dynamics must have a hyperbolic fixed point. Chen *et al.* show in [17] and Devasia *et al.* in [25] that finding a solution to the two-point boundary problem

$$\dot{\eta} = s(\eta, Y_d) \qquad \eta(\pm\infty) = 0 \tag{2.70}$$

will produce a bounded solution to the non-minimum phase zero-dynamics problem. The solution of the two-point boundary value problem in (2.70) will move the zero dynamics along the zero dynamics unstable manifold for $-\infty < t \leq t_0$, to an initial condition of the zero dynamics at t_0, such that the zero dynamics will acquire the zero dynamics stable manifold at some future time t_f.

Remark 2.2. Note that for $-\infty < t \leq t_0$ and $t_f < t < \infty$ the output is zero. Truncation of the trajectory is necessary for practical implementation. In addition, the non-casual part of the truncated trajectory can be shifted to t_0.

2.7 Differential Flatness

Definition 2.3. The nonlinear system in (2.1) with states $x \in \mathbb{R}^n$ is *differentially flat*, if there exists a change of variables $z \in \mathbb{R}^m$, given by an equation of the form

$$z = h(x, u, \dot{u}, \ldots, u^{(p)}), \tag{2.71}$$

such that the states and inputs may be determined from equations of the form

$$(x, u) = w(z, \dot{z}, \dots, z^{(l)}). \tag{2.72}$$

The change of variable will transform the system (2.1) into the trivial system $\dot{z} = v$. Differential flatness is not bound to an equilibrium. The transformation may take place around arbitrary trajectories. We will refer to the change of variables z as the *flat outputs*. The *flat outputs* are not necessarily the sensor outputs of a system. Note that equation (2.72) is only required to hold locally.

The significance of a system being flat is that all system behavior can be expressed without integration by the flat outputs and a finite number of its derivatives. That is, referring to Figure 2.3, the problem of finding curves that take the system from $x(0), u(0)$ to $x(T), u(T)$ in equation (2.1) is reduced to finding *any* sufficiently smooth curve that satisfies $z^k(0)$ and $z^k(T)$ up to some finite number. There is no need to solve a two-point boundary value problem if the system is differentially flat.

Once all the boundary conditions and trajectory constraints are mapped into the flat output space, (optimal) trajectories will be planned in the flat output space and then lifted back to the original state and input space as shown in Figure 2.3. The idea is that this methodology will alleviate adjoining the system dynamics in the optimal control problem formulation. Consequently, the number of variables in the optimal control problem will be reduced to expedite real-time computation.

It is debatable whether or not the necessary and sufficient conditions for differential flatness exist. Fliess *et al.* in [32] and [33] provide necessary conditions and Charlet *et al.* [16] provide sufficient conditions for a class of systems. Chetverikov in [19] is the first to provide necessary and sufficient conditions, but the solution appears to involve solving a set of nonlinear partial differential equations. Although the work of Chetverikov is promising, one frequently has to resort to trial and error to construct the flat outputs.

Example 2.2. Suppose we wish to generate trajectories for the system (2.73) from

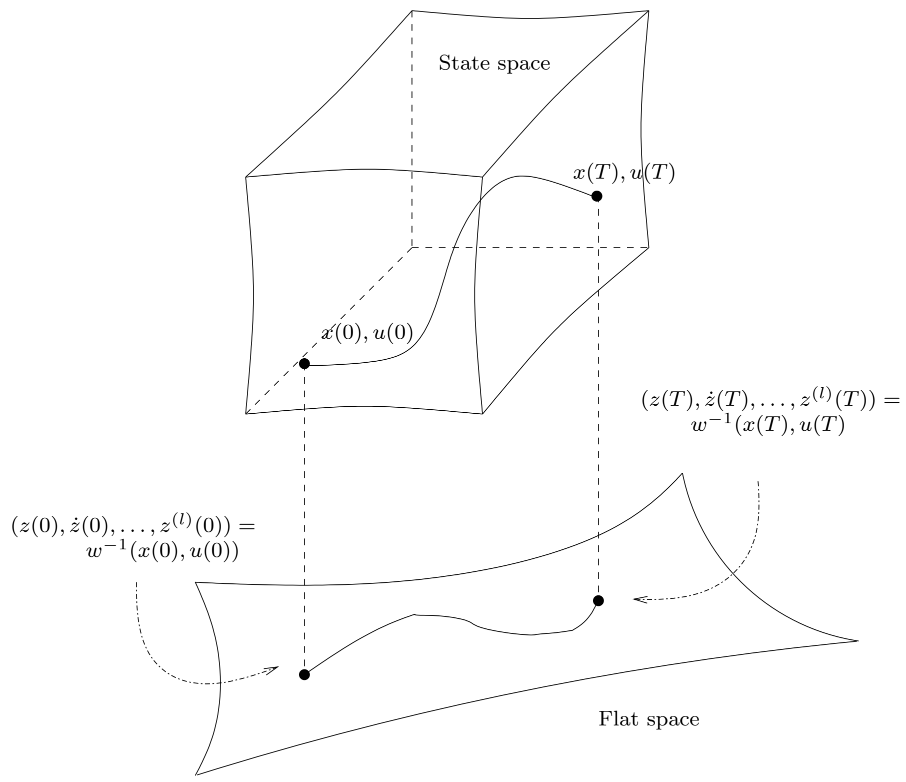
State space
$x(T), u(T)$
$x(0), u(0)$
$(z(T), \dot{z}(T), \ldots, z^{(l)}(T)) =$
$w^{-1}(x(T), u(T)$
$(z(0), \dot{z}(0), \ldots, z^{(l)}(0)) =$
$w^{-1}(x(0), u(0))$
Flat space

Figure 2.3: A flat system has the important property that the states and the inputs can be written in terms of the outputs z and their derivatives. Thus, the behavior of the system is determined by the flat outputs. Note that the map w is bijective.

the initial point $x(0), u(0)$ to the final point $x(T), u(T)$:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= u_1 \\ \dot{x_3} &= u_2 \cos x_2 \\ \dot{x_4} &= u_2. \end{aligned} \tag{2.73}$$

The flat output for this system is given by Chetvirkov in [19] to be $z_1 = x_1$ and

$z_2 = x_3 - x_4 \cos x_2$. Taking derivatives, we have,

$$\begin{aligned} z_1 &= x_1 \qquad \dot{z}_1 = x_2 \qquad \ddot{z}_1 = u_1 \qquad z_1^{(3)} = \dot{u}_1 \\ z_2 &= x_3 - x_4 \cos x_2 \qquad \dot{z}_2 = x_4 \sin x_2 u_1 \\ \ddot{z}_2 &= u_2 \sin x_2 u_1 + x_4 \cos x_2 u_1^2 + x_4 \sin x_4 \dot{u}_1. \end{aligned} \tag{2.74}$$

It can be shown that there is a local diffeomorphism between the variables

$$x_1, \ldots, x_4, u_2, u_1, \dot{u}_1$$

and the variables

$$z_1, \ldots, z_1^{(3)}, z_2, \dot{z}_2, \ddot{z}_2.$$

Therefore, by specifying the initial and final state, input, and auxiliary state ($\xi = \dot{u_i}$) we uniquely specify the flat outputs and their derivatives in flat output space. The problem has been resolved into one of solving an algebraic problem. Moreover, any curve that satisfies the boundary conditions in the flat output space is a trajectory of the original system (2.73).

Example 2.3. This example illustrates that the flat output may contain the input. This problem can be found in Martin *et al.* [66]:

$$\begin{aligned} \dot{x_1} &= u_1 \\ \dot{x_2} &= u_2 \\ \dot{x_3} &= u_1 u_2. \end{aligned} \tag{2.75}$$

The flat outputs are $z_1 = x_3 - x_1 u_2$ and $z_2 = x_2$. It can be shown that there is a local invertible map between the variables $x_1, x_2, x_3, u_1, u_2, u_2^{(1)}, u_2^{(2)}$ and the variables $z_1 z_1^{(1)}, , z_1^{(2)}, z_2, \ldots, z_2^{(3)}$.

2.8 System Design and Differential Flatness

A salient feature of flight control systems is that the requirements imposed are often conflicting. In this section, we look at the role differential flatness plays in a

VTOL design.

The four propeller “Aeroranger” is shown in Figure 2.4. The orientation of

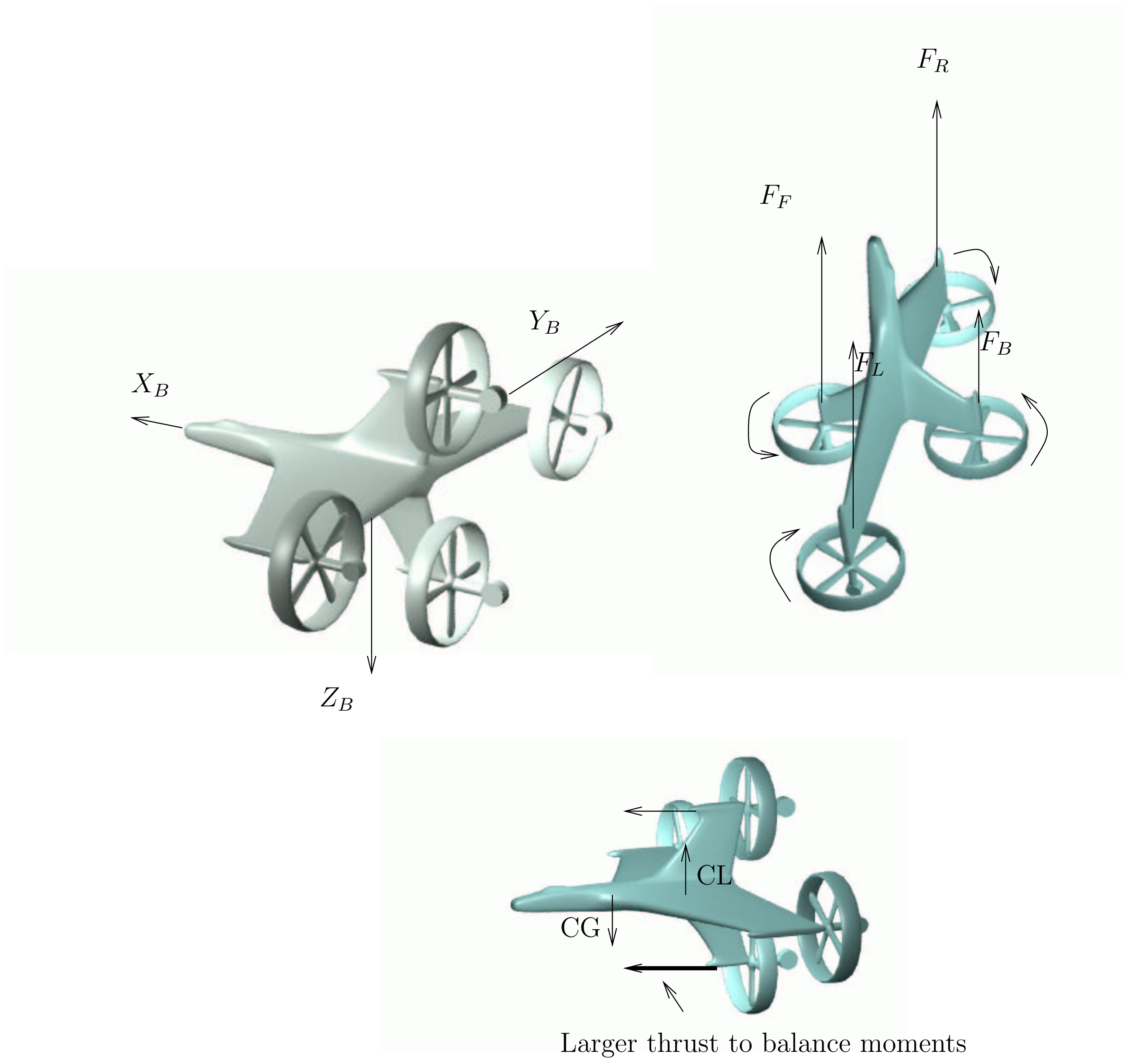

Figure 2.4: The views of the “Aeroranger” show the body coordinate frame, direction of rotation of the propellers, the definition of the applied forces due to the ducted fan thrust, and a method used for stabilizing the aircraft in aerodynamic flight without using aerodynamic surfaces.

the vehicle is given by

$$R_1 = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}. \tag{2.76}$$

R_1 maps vectors expressed with respect to body coordinates to inertial coordinates. The Euler angle rotation sequence $\psi\theta\phi$ is used to map a vector in inertial coordinates to body coordinates. We use Euler angles for visual clarity. Quaternions may be used to remove the orientation singularities associated with Euler angles.

The lift (L), drag (D), and side force (Y) are referenced with respect to wind coordinates. R_2 maps vectors in the wind coordinates to the body coordinates in terms of the angle of attack (α) and the side slip angle (β):

$$R_2 = \begin{bmatrix} \cos\alpha\cos\beta & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix}. \tag{2.77}$$

Applying Newton's laws gives the translational dynamics in inertial coordinates

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} - g \end{pmatrix} = R_1 \begin{pmatrix} F_{X_B} \\ F_{Y_B} \\ F_{Z_B} \end{pmatrix} + R_1 R_2 \begin{pmatrix} F_{X_A} \\ F_{Y_A} \\ F_{Z_A} \end{pmatrix}, \tag{2.78}$$

where (x, y, z) denotes the position, m is the mass, g is gravity, F_{X_B}, F_{Y_B}, and F_{Z_B} are the applied thrust forces in body coordinates, and F_{X_A}, F_{Y_A}, and F_{Z_A} are the aerodynamic forces. The translational equations of motion are written in inertial coordinates, since we are interested in generating trajectories for the position of the vehicle at low velocities. The applied thrust forces are fixed in the body as shown in Figure 2.4.

Using Euler's equation in body coordinates (assuming no products of inertia) gives the rotational dynamics

$$\begin{aligned} I_x\dot{\Omega}_x + (I_z - I_y)\Omega_y\Omega_z &= M_{X_b} + M_{X_a} \\ I_y\dot{\Omega}_y + (I_x - I_z)\Omega_z\Omega_x &= M_{Y_b} + M_{Y_a} \\ I_z\dot{\Omega}_z + (I_y - I_x)\Omega_x\Omega_y &= M_{Z_b} + M_{Z_a}, \end{aligned} \tag{2.79}$$

where I_x,I_y, and I_z are the principal moments of inertia, M_{X_b}, M_{Y_b}, and M_{Z_b} are the body moments due to the differential thrust forces, M_{X_a}, M_{Y_a}, and M_{Z_a} are the aerodynamic moments, and Ω_1, Ω_2, and Ω_3 are the angular rates in body coordinates given by

$$\begin{aligned} \Omega_x &= \dot{\phi} - \dot{\psi}\sin\theta \\ \Omega_y &= \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \\ \Omega_z &= \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi. \end{aligned} \tag{2.80}$$

The body moments are

$$\begin{aligned} M_{X_b} &= k_w(F_L + F_R - F_F - F_B) \\ M_{Y_b} &= l_{FB}(F_F - F_B) \\ M_{Z_b} &= l_{LR}(F_L - F_R), \end{aligned} \tag{2.81}$$

where k_w is the torque constant of the propeller, l_{FB} is half the distance between the applied force F_F and F_B, and l_{LR} is half the distance between the applied force F_L and F_R as seen in Figure 2.4.

Example 2.4. Near hover, the "Aeroranger" is differentially flat. Since the system is near hover, we will assume that the aerodynamic forces F_{X_a}, F_{Y_a}, and F_{Z_a} in equation (2.78) are zero. Choose the outputs for the "Aeroranger" to be

$$z_1 = x,\ z_2 = y,\ z_3 = z,\ z_4 = \phi. \tag{2.82}$$

Taking up to the fifth derivative of equation (2.82) and substituting equations (2.78), (2.79), and (2.80) we can write the flat outputs in terms of the states, inputs, and their derivatives.

$$\begin{aligned}
z_1^{(1)} &= \dot{x} \qquad z_1^{(2)} = \zeta_1(\cos\theta\cos\psi)\\
z_2^{(1)} &= \dot{y} \qquad z_2^{(2)} = \zeta_1(\cos\theta\sin\psi)\\
z_3^{(1)} &= \dot{z} \qquad z_3^{(2)} = -\zeta_1\sin\theta - g\\
z_4^{(1)} &= \dot{\phi} \qquad z_4^{(2)} = f_1(\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi},M_{X_b},M_{Y_b},M_{Z_b})\\
z_1^{(3)} &= f_2(\zeta_1,\dot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi})\\
z_2^{(3)} &= f_3(\zeta_1,\dot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi})\\
z_3^{(3)} &= f_4(\zeta_1,\dot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi})\\
z_1^{(4)} &= f_5(\zeta_1,\dot{\zeta}_1,\ddot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi},M_{Y_b},M_{Z_b})\\
z_2^{(4)} &= f_6(\zeta_1,\dot{\zeta}_1,\ddot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi},M_{Y_b},M_{Z_b})\\
z_3^{(4)} &= f_7(\zeta_1,\dot{\zeta}_1,\ddot{\zeta}_1,\psi,\phi,\theta,\dot{\psi},\dot{\theta},\dot{\phi},M_{Y_b},M_{Z_b}),
\end{aligned} \tag{2.83}$$

where

$$\zeta_1 = F_L + F_R + F_F + F_B \tag{2.84}$$

is the combined force of the propellers along X_b as shown in Figure 2.4. In short,

$$(z_1,\dots,z_2^{(4)},z_2,\dots,z_2^{(4)},z_3,\dots,z_3^{(4)},z_4,\dot{z}_4,\ddot{z}_4) = \Psi(\xi), \tag{2.85}$$

where

$$\xi = (x,y,z,\theta,\phi,\psi,\dot{x},\dot{y},\dot{z},\dot{\theta},\dot{\phi},\dot{\psi},M_{X_b},M_{Y_b},M_{Z_b},\zeta_1,\dot{\zeta}_1,\ddot{\zeta}_1).$$

The above relation is locally invertible, with the exception of a few points, since

$$\det(\frac{\partial\Psi}{\partial\xi}) = \frac{-\zeta_1^6\cos 2\theta - \zeta_1^6}{2I_xI_yI_z}$$

in nonzero. The "Aeroranger" is differentially flat by Definition 2.3. For implementation, it is desirable to have a closed form solution of equation (2.85). First,

ψ can be found from the following:

$$\tan\psi = \frac{\ddot{z}_2}{\ddot{z}_1}. \tag{2.86}$$

Second, θ can be found from the following:

$$\tan\theta = \frac{g - \ddot{z}_3}{\ddot{z}_1\cos\psi + \ddot{z}_2\sin\psi}. \tag{2.87}$$

Using the information provided by two derivatives of equation (2.86), (2.87), the flat output z_4 and then substituting into equation (2.79), it is possible to recover M_{X_b}, M_{Y_b}, and, M_{Z_b}. Using equation (2.84), ζ_1 may be recovered from the following:

$$\zeta_1 = m\sqrt{\ddot{z}_1{}^2 + \ddot{z}_2{}^2 + (\ddot{z}_3 - g)^2}. \tag{2.88}$$

Finally, the applied forces may be recovered using equation (2.88) and equation (2.81). As a result of using the two argument tangent function, we can avoid the singularity at $\theta = \frac{\pi}{2}$.

Remark 2.3. It is not difficult to show that the "Aeroranger" is differentially flat, when the aerodynamic forces are not negligible. However, a closed form solution may be elusive. We will assume that actuation for this system will not only be the thrusts due to the propellers, but also four aerodynamic surfaces, one on each wing of the "Aeroranger". An additional assumption is needed, that the aerodynamic surfaces only contribute to the aerodynamic moments and not the aerodynamic forces. We choose the flat outputs the same as those in equation (2.82). We will assume that the aerodynamic forces are a function of V, α, β. Moreover, we will assume that the aerodynamic moments are a function of V, α, β, and δ_i, where δ_i $i = 1, \ldots, 4$ is the deflection angle of the *ith* surface.

First, we will solve for ψ, θ and ζ_1. Expressions for α and β can be determined from

$$\tan\alpha = \frac{w}{u} \qquad \sin\beta = \frac{v}{V}$$

where u, v, and w are the body velocities. The body velocities maybe found from

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R_1^T \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{pmatrix}$$

and $V = \sqrt{\dot{z}_1^2 + \dot{z}_3^2 + \dot{z}_3^2}$. Now, we can write

$$(\psi, \theta, \zeta_1) = a_1(\dot{z}_1, \ddot{z}_1, \dot{z}_2, \ddot{z}_2, \dot{z}_3, \ddot{z}_3, z_4) \tag{2.89}$$

using equations (2.78). Similarly, using equations (2.79), we can write

$$(M_1, M_2, M_3) = a_2(\dot{z}_1, \ldots, z_1^{(4)}, \ldots, z_2^{(4)}, \dot{z}_3, \ldots, z_3^{(4)}, z_4, \dot{z}_4, \ddot{z}_4) \tag{2.90}$$

where $M_1 = M_{X_b} + M_{X_a}$, $M_2 = M_{Y_b} + M_{Y_a}$,$M_3 = M_{Z_b} + M_{Z_a}$. The combination of F_L, F_R, F_F, F_B, and δ_i $i = 1, \ldots, 4$ should be optimized, depending on the flight regime.

Remark 2.4. Martin in [64] showed that a planar approximation of a VTOL is differentially flat. In addition, Martin also illustrated in [63] that a conventional aircraft in forward flight is differentially flat. The difficulty has been determining a set of flat outputs for a VTOL on the configuration manifold $SE(3)$. We have shown that, with appropriate actuation, an aircraft can be designed that is differentially flat in both the forward flight and hover flight regimes.

The differential flatness characteristics have been summarized for several VTOL configurations near hover and in forward flight, as seen in Figure 2.5. The class of VTOL systems under consideration has rigid rotor blades. Due to the complexity, we do not consider thrust vectoring as an option for the 3-D systems or engine air bleeds for actuation. All configurations are assumed to have aerodynamic surfaces in forward flight that contribute only to the aerodynamic moments, and not the aerodynamic forces. Table 2.8 is a summary of the results.

VTOL	Hover actuation	Flat near hover?	Flat in forward flight?
Ducted Fan	Thrust vectored ducted fan	Yes	Yes, with no vectoring; unknown otherwise
Two Ducted Fans	Two ducted fans	Yes	Yes
"Aeroranger"	Four propellers	Yes	Yes
"Aerojeep"	Four ducted fans	Unknown	Unknown

Table 2.1: Summary of the flatness characteristics of several simple VTOL configurations. Flat in forward flight implies that the mixing of hover actuation and aerodynamic surfaces is possible.

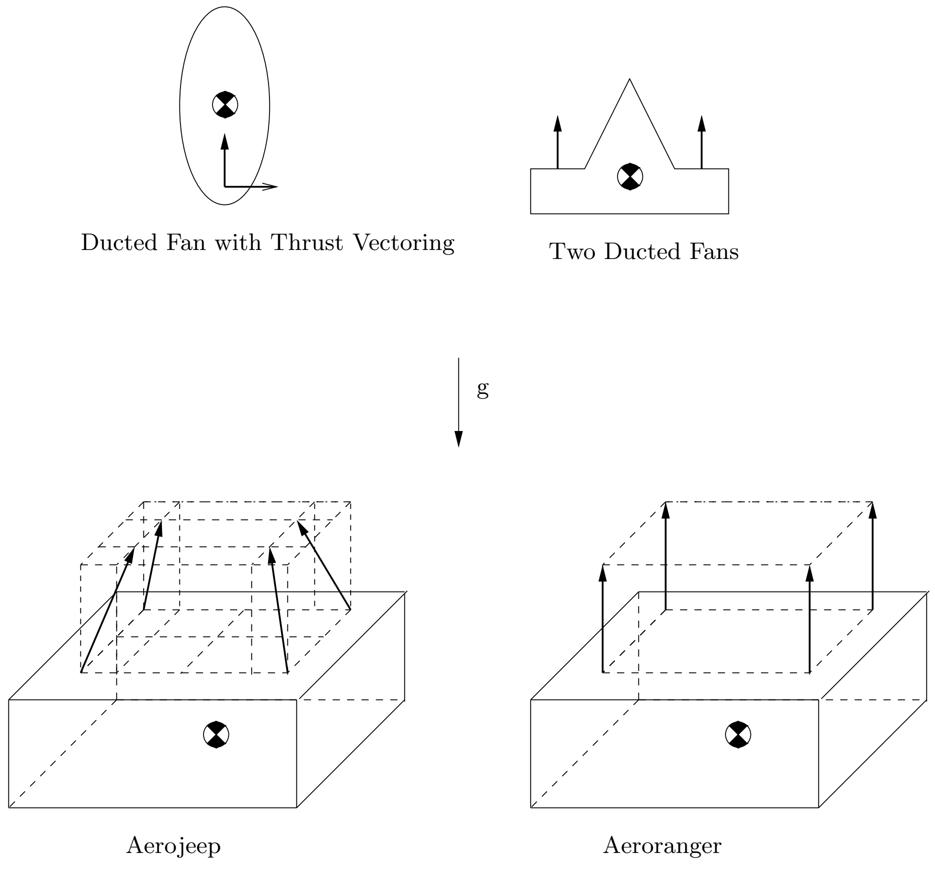

Figure 2.5: Two planar VTOL configurations and two six DOF configurations

2.9 Parameterization of the Output

In the previous section, techniques were presented to reduce or eliminate the dynamic constraints by selecting a special set of variables (outputs) that could completely characterize the states and inputs of the system under consideration. In this section, we will discuss how to select the outputs from a finite dimensional space, in order that the problem under consideration can be efficiently solved.

There are many curves that can be used to approximate the outputs (Fourier series, polynomials, rational segments, etc.). Aside from accurately representing a basis of the solution of the trajectory generation problem under consideration with a reasonable number of decision variables, the main requirements of the curve are the ability to set a level of continuity C^k, without adding additional constraints. Local support is also a desirable property of the basis functions. Local support means that the curves only influence a region of the curve local to the current point of interest. Specifying the level of continuity is necessary, since the states and inputs are a function of the outputs and their *derivatives.* Local support is favorable for numerically stable computer implementation. A high order single polynomial would be necessary to satisfy complex constraints. Solving for the coefficients of the polynomial would be an inefficient and ill-conditioned operation.

A solution that meets the main requirements is piecewise Bezier polynomials or B-splines. An overview of B-splines, from which much of the following is derived, can be found in Deboor [9].

An output $y(t)$ may be defined in terms an order k Bezier curve by

$$y(t) = \sum_{i=0}^{n} N_{i,k}(t) C_i \qquad 0 \leq t \leq 1.$$

The coefficients C_i are called *control points.* The basis functions $N_{i,n}$ are the kth-order Berstein polynomials given by

$$N_{i,k} = \left(\begin{array}{c} k \\ i \end{array} \right) (1-t)^{k-i} t^i.$$

A B-spline curve is constructed from Bezier curves joined together with a prescribed level of continuity between them. The points at which the curves are joined are called the *breakpoints*. The breakpoints are a strictly increasing sequence of real numbers. Figure 2.6 provides an example of a B-spline output with and a constraint. A nondecreasing sequence of real numbers containing *breakpoints* $\tau = t_0, \ldots, t_K$ are called the *knot vector*. The difference between the breakpoints and the knot vector is that there may contain additional breakpoints in the knot vector containing the same value. Frequently, the breakpoints are referred to as the *knot points*. The number of times a breakpoint appears in the interior of knot vector is known as the *multiplicity* m_i. The *smoothness* s_i of a breakpoint provides the level of continuity at a breakpoint. A breakpoint is C^{s_i-1} times continuously differentiable. The smoothness and multiplicity are related by the following:

$$k_i = s_i + m_i \qquad \text{for interior breakpoints} \in (t_0, t_p)$$

where k_i is the order of the piecewise polynomial segments. A recurrence relation is used to define the B-spline basis functions $B_{i,j}$ of the B-spline curves

$$y(t) = \sum_{i=0}^{n} B_{i,k}(t) C_i \qquad t_0 \leq t \leq t_p.$$

Given the knot vector τ and the order k, the B-spline basis functions are defined by:

$$\begin{aligned} B_{i,0}(t) &= \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ B_{i,k}(t) &= \tfrac{t-t_i}{t_{i+k+1}-t_i} B_{i,k-1}(t) + \tfrac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(t). \end{aligned} \tag{2.91}$$

B-spline Curve Properties

A comprehensive list of B-spline properties can be found in Deboor [9] and Piegl and Tiller [81]. Several important properties of B-splines that are useful are listed:

1. Local Support: the B-spline basis function $B_{i,k}$ is zero outside the interval $[t_i, t_{i+k}]$.

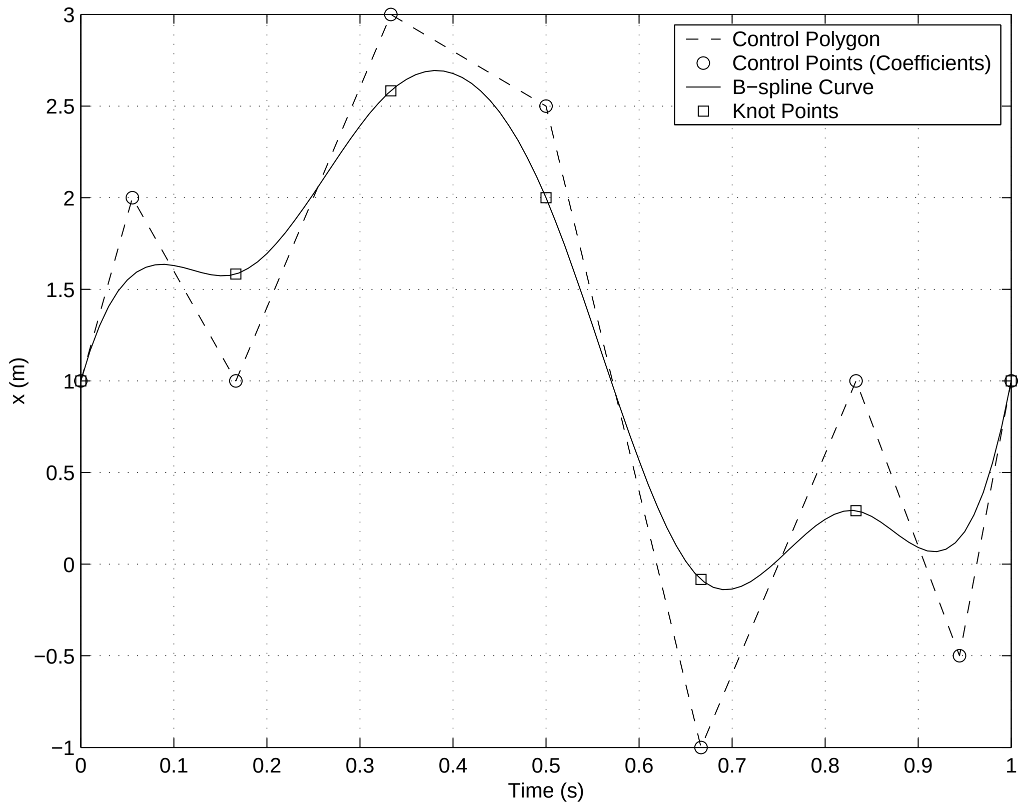

Figure 2.6: The B-spline curve has six intervals ($l = 6$), forth order ($k = 4$), and is C^3 at the breakpoints (or smoothness $s = 3$). The nine control points are the decision variables.

2. Number of Control Points: the number of control points (B-spline coefficients)

$$P = lk - \sum_{i=0}^{l-1} s_i,$$

   where l is the number of intervals, k the order, and s the smoothness. If all breakpoints have the same smoothness $s = s_i \forall i$, the number of control points is $P = l(k - s) + s$.

3. Convex Hull: if $t \in [t_i, t_{i+1})$, then $y(t)$ lies within the convex hull of control points $C_{i-k-1}, \ldots, C_i$.

4. Non-negativity: $B_{i,k} \geq 0$ for all k,i, and t.

5. Differentiability: all derivatives of the B-spline curve exits on the interior of the breakpoint span. The curve is C^{s_i-1} times continuously differentiable at the breakpoints.

Derivatives of the B-spline curve are

$$y^{(j)}(t) = \sum_{i=0}^{n} B_{i,k}^{(j)}(t) C_i \qquad t_0 \leq t \leq t_p,$$

where the jth derivative of the B-spline basis function is given by

$$\begin{aligned} B_{i,k}^{(j)}(t) &= \tfrac{k-1}{k-i-j} \left[\tfrac{t-t_i}{t_{i+k+1}-t_i} B_{i,k-1}^{(j)}(t) + \tfrac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}^{(j)}(t) \right]. \\ j &= 0, \ldots, k-1 \end{aligned} \tag{2.92}$$

Other possibilities exist for the use of B-splines to approximate outputs. Functions of B-splines of the form $z = h(y(t))$ are useful when piecewise polynomials are not sufficiently accurate to represent the basis functions of the optimal control problem under consideration. For example, consider a three-link model of a fish with the angle of one link as the input (flapping motion) into the system. In this situation, a sinusoid with varying amplitude may be a better choice of output than a strictly piecewise polynomial.

Quaternion as basis functions

Quaternions are a convenient choice of curves to represent orientations in $SO(3)$. However, the orientation is represented by the quaternion $q \in \mathbb{R}^4$ with one constraint $||q||= 1$. This implies that if B-spline curves are used to represent each component of the quaternion, it will also be necessary to preserve the nonlinear unit norm condition when solving for the control points, which is clearly undesirable.

Given a vector $v = \theta \hat{v}$, with $\hat{v} \in S^2$, the exponential

$$\exp(v) = \sum_{i=0}^{\infty} v^i = (\cos\theta, \hat{v}\sin\theta) \in S^3$$

is the unit quaternion which represents the rotation by angle 2θ about the axis $\hat{v}$, where v^i is computed using quaternion multiplication.

Suppose that it is desired to move the orientation of an object in $SO(3)$ from one point to another point. Assume that it is required to rotate about n known axes $v_1, v_2, v_n,\ v_i \in \mathbb{R}^3$ consecutively starting from a known initial quaternion q_0. The quaternion evolution may be written

$$q(t) = q_0 \prod_{i=1}^{n} \exp\left(v_i y_i(t)\right),$$

where $y_i(t)$ are B-spline curves. Since $||\exp(v_i y_i)||= 1\ \forall\ i$, the quaternion curves are in S^3; thus, the unit norm condition is automatically satisfied. It can easily be determined that local controllability of the curve is preserved.

Another representation of the quaternion may be the following:

$$q(t) = q_0 \exp \begin{pmatrix} \omega(t) \cos y_1(t) \cos y_2(t) \\ \omega(t) \cos y_2(t) \sin y_1(t) \\ \omega(t) \sin y_2(t) \end{pmatrix}, \tag{2.93}$$

where $\omega(t)$, $y_1(t)$, and $y_2(t)$ are B-spline curves. In this case, the axis of rotation $\omega(t)$ is allowed to change.

2.10 Summary and Conclusions

In this chapter, we presented an overview of of the classical numerical methods for solving constrained optimal control problems. Solving a constrained optimal control problems by the maximum principle, or related necessary conditions, is known as an indirect method. The problem does not contain a closed form solution, multiple shooting or relaxation techniques can be employed for a numerical solution. The advantage of indirect methods is that very accurate solutions can be obtained. The main disadvantage of indirect methods is their lack of robustness to a poor initial guess. We presented a homotopy method that uses a database of solutions as an initial guess. The technique looks promising, but suffers from undesirable bifurcations.

Direct methods obtain solutions through the direct minimization of the objective functions subject to the constraints imposed by the optimal control problem. Two direct methods were discussed: direct collocation and the methods of adjoints. Direct collocation has the advantage that it fairly robust to the initial guess. The disadvantage it that the problem formulation usually results in large optimization problems, not amenable to real-time implementation, due to finite dimension approximations of the optimal control problem. The adjoint method has the advantage that its solution accuracy rivals that of indirect methods. The disadvantage of the adjoint method is that is cumbersome to use with trajectory constraints and is prone to integration problems over large time periods.

In addition, we discussed nonlinear control techniques that are useful in trajectory generation. Namely, if a system is differentially flat, one can design arbitrary trajectories to take the system from any initial and final conditions.

Finally, we discussed using differential flatness in VTOL design and demonstrated differential flatness for a neoteric unmanned VTOL design.

Chapter 3

The Nonlinear Trajectory Generation (NTG) Algorithm

In the previous chapter, several classical techniques for optimal trajectory generation of constrained systems were presented. The advantages and disadvantages of each technique for implementation in a real-time environment were discussed. In this chapter, we present a new approach to trajectory generation that combines the benefits of several classical trajectory generation techniques. There are three primary steps to the real-time nonlinear trajectory generation (NTG) approach we propose. The first is to determine outputs, such that equation (2.1) can be mapped to a lower dimensional output space. Once this is done, the cost in equation (2.2) and the constraints in equation (2.3) can also be mapped to the output space. The second is to parameterize the outputs in terms of B-spline curves. Finally, nonlinear programming is used to solve for the coefficients of the B-splines to minimize the cost subject to constraints in output space.

3.1 Transforming the System Dynamics to a Lower Dimensional Space

The first step of the NTG algorithm is to map the system dynamics in equation (2.1) to a lower dimensional space. This is done by exploiting the concepts of feedback linearization and differential flatness in Section 2.6 and 2.7, respectively.

To summarize, our goal is to find outputs

$$z = h(x, u) \;\; z \in \mathbb{R}^m, \tag{3.1}$$

such that the original states x and inputs u can be recovered from the outputs and their derivatives. That is,

$$\xi = \Phi(x, u) \;\; \xi = (z_1, \ldots, z_1^{r_1}, \ldots, z_m, \ldots, z_m^{r_m}) \tag{3.2}$$

Since Φ is at least locally invertible, we will assume that

$$x = w_1(\xi) \;\text{ and } u = w_2(\xi) \tag{3.3}$$

can be written in closed form. If such outputs can be found, the system is called *differentially flat*.

3.1.1 Differentially Flat Systems

If the system dynamics are differentially flat, then the algebraic equations (3.3) implicitly reflects the system dynamics as in (2.1). Substituting (3.3) into equations (2.2) and (2.3), the optimal control problem can be reformulated in the following form:

$$\begin{aligned}
\min_{\xi} J \;\; &:= \;\; \phi_0(w_1(\xi_0), w_2(\xi_0), t_0) + \phi_f(w_1(\xi_f), w_2(\xi_f), t_f) + \\
&\qquad \int_{t_0}^{t_f} L(w_1(\xi), w_2(\xi), t) dt \\
&\qquad \text{subject to:} \\
lb_0 \;\; &\leq \;\; \psi_0(w_1(\xi_0), w_2(\xi_0)) \leq ub_0, \\
lb_f \;\; &\leq \;\; \psi_f(w_1(\xi_f), w_2(\xi_f)) \leq ub_f, \\
lb_t \;\; &\leq \;\; S(w_1(\xi), w_2(\xi), t) \leq ub_t.
\end{aligned} \tag{3.4}$$

In this optimal control problem the decision variables are ξ which are the outputs and their derivatives. There are several issues that need to be addressed:

1. We have noted in Section 2.7 that not all systems are differentially flat. In general, even if flat outputs can be proven to exist, finding the flat outputs requires solving a set of nonlinear partial differential equations, for which an analytic solution might not exist.

2. There may not be a closed form solution to Φ^{-1}.

3. The flat output transformation may complicate the expressions for the objective and constraint functions.

In the next section, we will develop a systematic approach to address the above issues.

3.1.2 An Algorithm to Map the System Dynamics to Lower Dimensional Space

Algorithm 3.1. Steps to map system dynamics into a lower dimensional space.

```
Input: System dynamics in the form of equation (2.51)
Check: Static feedback linearization (Theorem 2.2)
If Static feedback linearizable and resulting PDE solvable
    Output: Flat outputs, end algorithm
Input: Trial set of $m$ outputs
Compute: Zero dynamics from equation (2.61)
Compute: Decoupling matrix from equation (2.56)
If Relative degree well defined
    Output: Zero dynamics, end algorithm
Else
    Apply: (MIMO only) Dynamic Extension Algorithm in Isidori [46]
    If Dynamic extension algorithm succeeds
        Output: Zero dynamics and dynamic compensator, end algorithm
    Else
        Compute: Equality constraints due to ill-defined relative degree
        Output: Zero dynamics, dynamic compensator and
        equality constraints due to ill-defined relative degree, end algorithm
```

Managing the zero dynamics and ill-defined relative degree

If the decoupling matrix has $\text{rank}(B(x)) < m$, we apply a linear transformation to the input vector $u = T\hat{u}$ so that there are $m - q$ zero columns in the decoupling matrix

$$\begin{pmatrix} z_1^{(r_1)} \\ \vdots \\ z_m^{(r_m)} \end{pmatrix} = \begin{pmatrix} L_f^{r_1} h_1 \\ \vdots \\ L_f^{r_m} h_m \end{pmatrix} + B(x)u := \alpha(x) + E(x)\hat{u} \tag{3.5}$$

$$\dot{\eta} = q_1(\eta, \xi) + q_2(\eta, \xi)u,$$

where

$$\xi = (z_1, \dot{z}_1, \ldots, z_1^{(r_1-1)}, z_2, \ldots, z_2^{(r_2-1)}, \ldots, z_m^{(r_m-1)})$$

and

$$E(x)\hat{u} = \begin{pmatrix} E_{11} & 0 \\ E_{21} & 0 \end{pmatrix} \begin{pmatrix} \hat{u}_1 \\ \vdots \\ \hat{u}_q \\ \hat{u}_{q+1} \\ \vdots \\ \hat{u}_m \end{pmatrix} \tag{3.6}$$

with $\text{rank}(E_{11}) = q$. Now, equation (3.5) can be written

$$\begin{pmatrix} z_1^{(r_1)} \\ \vdots \\ z_q^{(r_q)} \end{pmatrix} = \begin{pmatrix} L_f^{r_1} h_1 \\ \vdots \\ L_f^{r_q} h_q \end{pmatrix} + E_{11} \begin{pmatrix} \hat{u}_1 \\ \vdots \\ \hat{u}_q \end{pmatrix} \tag{3.7}$$

$$\begin{pmatrix} z_1^{(r_{q+1})} \\ \vdots \\ z_q^{(r_m)} \end{pmatrix} = \begin{pmatrix} L_f^{r_{q+1}} h_{q+1} \\ \vdots \\ L_f^{r_m} h_m \end{pmatrix} + E_{21} \begin{pmatrix} \hat{u}_1 \\ \vdots \\ \hat{u}_q \end{pmatrix} \tag{3.8}$$

$$\dot{\eta} = q_1(\eta, \xi) + q_2(\eta, \xi)\hat{u} \tag{3.9}$$

Equation (3.7) indicates which states and inputs can be determined from the outputs. Equation (3.8), due to the ill-defined relative degrees, and equation (3.9), due to the zero dynamics, will remain as dynamic constraints. These constraints must be reflected in our algorithm. This is done through introduction of additional "pseudo" outputs $z^{m+1}, \ldots, z^{p+m}$ such that all the remaining states and *inputs* are observable. A test for observability was given in Theorem 2.1.

In summary, when it is not possible or desirable to use the flat output, additional outputs along with equality constraints must be introduced to represent the original system dynamics. Let

$$z = h(x, u) \;\; z \in \mathbb{R}^{p+m} \tag{3.10}$$

include the original and extended outputs, such that the original states x and inputs u can be recovered from the outputs and their derivatives as in the following:

$$\xi = \Psi(x, u) \;\; \xi = (z_1, \ldots, z_1^{r_1}, \ldots, z_{p+m}, \ldots, z_{p+m}^{r_{p+m}}) \tag{3.11}$$

$$\begin{aligned} x = w_1(\xi) \;\; &\text{and } u = w_2(\xi) \\ \Upsilon(\xi) = 0&, \;\; \Upsilon \in \mathbb{R}^p \end{aligned} \tag{3.12}$$

where $\Upsilon(\xi)$ are constraints due to the "pseudo" outputs. The optimal control formulation in (3.4) is extended to include the equality constraints due to the additional "pseudo" outputs to the following:

$$\begin{aligned} \min_{\xi} J \;\; &:= \;\; \phi_0(w_1(\xi_0), w_2(\xi_0), t_0) + \phi_f(w_1(\xi_f), w_2(\xi_f), t_f) + && (3.13) \\ & \quad\quad \int_{t_0}^{t_f} L(w_1(\xi), w_2(\xi), t)dt \\ & \quad\;\; \text{subject to:} \\ \Upsilon(\xi) \;\; &= \;\; 0 && (3.14) \\ lb_0 \;\; &\leq \;\; \psi_0(w_1(\xi_0), w_2(\xi_0)) \leq ub_0, && (3.15) \\ lb_f \;\; &\leq \;\; \psi_f(w_1(\xi_f), w_2(\xi_f)) \leq ub_f, && (3.16) \\ lb_t \;\; &\leq \;\; S(w_1(\xi), w_2(\xi), t) \leq ub_t. && (3.17) \end{aligned}$$

3.2 Parameterization of the Flat Outputs and “Pseudo” Outputs

The second step to the NTG algorithm was parameterizing the outputs with a finite-dimensional approximation. In Section 2.9, we explained the properties of B-splines that make them desirable as basis functions to parameterize the outputs. These include including compact (local) support, ease of enforcing continuity at breakpoints, and numerical stability. For each output z_i , the order k_i , continuity C^s or smoothness s_i, and knot breakpoints $\tau_i = t_0, ..., t_{K_i}$ will be selected in consideration of the maximum derivative that occurs in the output and the number of desired decision variables.

A sample spline trajectory is depicted in Figure 3.1.

The outputs are written in terms of finite dimensional B-spline curves as:

$$\begin{aligned}
z_1 &= \textstyle\sum_{i=1}^{q_1} B_{i,k_1}(t)C_i^1 \text{ for the knot breakpoint sequence } \tau_1 \\
z_2 &= \textstyle\sum_{i=1}^{q_2} B_{i,k_2}(t)C_i^2 \text{ for the knot breakpoint sequence } \tau_2 \\
&\vdots \\
z_{p+m} &= \textstyle\sum_{i=1}^{q_{p+m}} B_{i,k_q}(t)C_i^{p+m} \text{ for the knot breakpoint sequence } \tau_{p+m} \\
&\quad \text{and} \quad q_i = l_i(k_i - s_i) + s_i
\end{aligned}$$

where $B_{i,k_j}(t)$ is the B-spline basis function defined in Section 2.9 and $l_i = dim(\tau_i) - 1$ as the number of knot intervals for the ith output. After the outputs have been parameterized in terms of B-spline curves, the coefficients of the B-spline basis functions will be found using nonlinear programming.

3.3 Transformation into a Nonlinear Programming Problem

The last step to the NTG algorithm is to transform the optimal control problem, represented in the new coordinates by equations (3.14) through (3.17), into a nonlinear programming problem. Let $d_c = t_0, \ldots, t_N$ denote the collocation points.

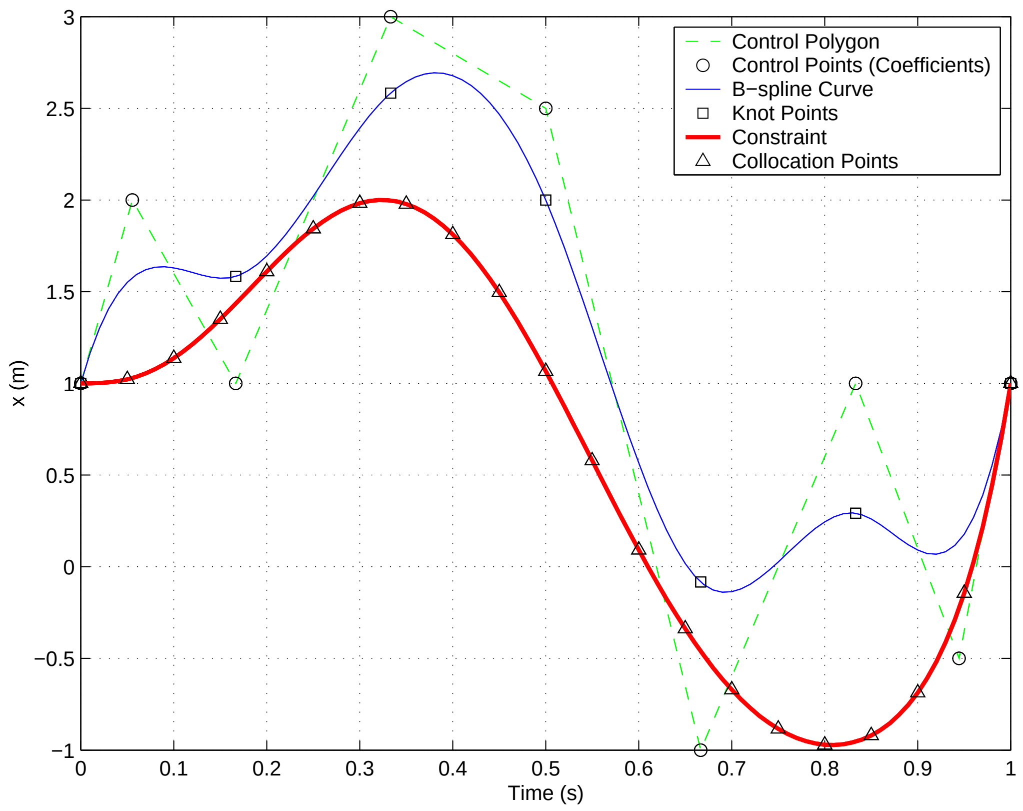

Figure 3.1: In this hypothetical problem, the B-spline curve has six intervals ($l = 6$), fourth order ($k = 4$), and is C^3 at the breakpoints (or smoothness $s = 3$). The constraint on the B-spline curve (to be larger than the constraint in this example) will be enforced at the 21 collocation points. The nine control points are the decision variables.

The collocation points are the points in the time interval that the constraints are enforced (see Figure 3.1). The integration points or mesh points must also be specified. Without loss of generality, we will assume that the integration points are identical to the collocation points and the same for each constraint.

Integration

The integral $I = \int_{t_0}^{t_P} \tilde{L}(t)dt$ can be approximated using a *quadrature rule*, which is a sum of the form

$$\hat{I} = \sum_{i=0}^{P} \mu_i L(t_i),$$

where the weights μ_i and the collocation points t_i are determined in advance. The P-point form of these rules typically obtain a convergence rate of $\mathcal{O}(P^{-r})$ for some integer $r \geq 1$, provided that the integrand has sufficient continuity. For example, the error using Simpson's rule is $|\hat{I} - I| = \mathcal{O}(P^{-4})$ provided that $\tilde{L}$ has at least four continuous derivatives. An overview of quadrature methods can be found in Stoer and Burlisch [99].

Using the evaluation rules in Section 2.9, the B-splines will have the sparse form as shown in equation (3.18) when evaluated at each of the prescribed collocation points d_i.

$$
\begin{bmatrix}
z_i^{(0)}(t_0)\\ z_i^{(1)}(t_0)\\ \vdots\\ z_i^{(r_i)}(t_0)\\ z_i^{(0)}(t_1)\\ z_i^{(1)}(t_1)\\ \vdots\\ z_i^{(r_i)}(t_1)\\ z_i^{(0)}(t_2)\\ z_i^{(1)}(t_2)\\ \vdots\\ z_i^{(r_i)}(t_2)\\ z_i^{(0)}(t_3)\\ z_i^{(1)}(t_3)\\ \vdots\\ z_i^{(r_i)}(t_3)\\ \vdots\\ z_i^{(0)}(t_{N-1})\\ z_i^{(1)}(t_{N-1})\\ \vdots\\ z_i^{(r_i)}(t_{N-1})\\ z_i^{(0)}(t_N)\\ z_i^{(1)}(t_N)\\ \vdots\\ z_i^{(r_i)}(t_N)
\end{bmatrix}
=
\begin{bmatrix}
\mathrm{x} & & & \\
\mathrm{x\,x} & & & \\
\vdots\ \ \ddots & & & \\
\mathrm{x\,x} \cdots \mathrm{x} & & & \\
\mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & & \\
\mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & & \\
\vdots \quad \vdots \ \vdots \ \vdots & & & \\
\mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& \vdots \quad \vdots \ \vdots \ \vdots & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& \vdots \quad \vdots \ \vdots \ \vdots & & \\
& \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} & & \\
& & \ddots\ \ddots & \\
& & & \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} \\
& & & \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} \\
& & & \vdots \quad \vdots \ \vdots \ \vdots \\
& & & \mathrm{x\,x\,x} \cdots \mathrm{x\,x} \cdots \mathrm{x} \\
& & & \mathrm{x} \\
& & & \mathrm{x\,x} \\
& & & \vdots \\
& & & \mathrm{x} \cdots \mathrm{x\,x}
\end{bmatrix}
\begin{bmatrix}
C_1^i\\ C_2^i\\ \vdots\\ C_{k_i-s_i}^i\\ C_{k_i-s_i+1}^i\\ \vdots\\ C_{2(k_i-s_i)}^i\\ C_{2(k_i-s_i)+1}^i\\ \vdots\\ C_{l_i(k_i-s_i)}^i\\ C_{l_i(k_i-s_i)+1}^i\\ \vdots\\ C_{l_i(k_i-s_i)+s_i}^i
\end{bmatrix}
\tag{3.18}
$$

Let

$$Z_i(t) := z_i^{(0)}(t_0), z_i^{(1)}(t_0), \dots, z_i^{(r_i)}(t_0), \dots, z_i^{(0)}(t_N), z_i^{(1)}(t_N), \dots, z_i^{(r_i)}(t_N) \in \mathbb{R}^{(r_i+1)\times d_c}$$

denote an output z_i and its derivatives evaluated at the *all* collocation points. Let $Z(t_s) := (Z_1(t_s), \dots, Z_{p+m}(t_s))$ designate ξ evaluated at the time $t_s \in d_c$. Let G_i be the collocation matrix resulting from the B-spline basis functions for the ith output in equation 3.18 and the B-spline coefficients for the ith input as

$$U_i := C_1^i, C_2^i, \dots, C_{l_i(k_i-s_i)+s_i}^i \in \mathbb{R}^{l_i(k_i-s_i)+s_i}.$$

Let $U := (U_1, \ldots, U_{p+m})$ denote the B-spline coefficients for all outputs. These are the decision variables in the nonlinear programming problem.

The cost in equation (3.14) is written

$$\begin{aligned} J(U) \approx \quad & \phi_0(w_1(Z(t_0)), w_2(Z(t_0), t_0) + \phi_f(w_1(Z(t_f)), w_2(Z(t_f), t_f) + \\ & \sum_{j=0, j=j+P}^{N} \sum_{k=0}^{P} \mu_k L(w_1(Z(t_{k+j}), w_2(Z(t_{k+j}), t_{k+j}) \end{aligned} \tag{3.19}$$

where μ_k is dependent on the quadrature. The trajectory constraints in equations (3.15) and (3.17) are written

$$\begin{aligned} \Upsilon_i(U) = \quad & \Upsilon(w_1(Z(t_i)), w_2(Z(t_i)), t_i) \; i = 1, \ldots, N \\ S_i(U) = \quad & S(w_1(Z(t_i)), w_2(Z(t_i)), t_i) \; i = 1, \ldots, N. \end{aligned} \tag{3.20}$$

Similarly, the initial and final constraints in equations (3.16) and (3.17), respectively, are

$$\begin{aligned} \psi_0(U) = \quad & \psi_0(w_1(Z(t_0), w_2(Z(t_0)) \\ \psi_f(U) = \quad & \psi_f(w_1(Z(t_f), w_2(Z(t_f)). \end{aligned} \tag{3.21}$$

Let $c(U)$ denote all the constraints

$$(\psi_0(U), \Upsilon_1(U), \ldots, \Upsilon_N(U), w_1(U), \ldots, S_N(U), \psi_f(U)).$$

The nonlinear programming problem can be stated in the form

$$\begin{aligned} \min_{U \in \mathbb{R}^M} F(U) \qquad & \text{subject to} \quad lb \leq c(U) \leq ub \\ & \text{where } M = \sum_{j=1}^{m+p} = l_j(k_j - s_j) + s_j. \end{aligned} \tag{3.22}$$

In Section 2.4 we noted that the gradient of the cost and the constraints were necessary for efficient solutions by the nonlinear programming solver. The gradient of the cost function with respect to the decision variables U is

$$\begin{aligned}\frac{\partial J}{\partial U} = & \frac{\partial \phi_0(w_1(Z(t_0)), w_2(Z(t_0), t_0)}{\partial Z(t_0)} \frac{\partial Z(t_0)}{\partial U} + \\ & \frac{\partial \phi_f(w_1(Z(t_f)), w_2(Z(t_f), t_f)}{\partial Z(t_f)} \frac{\partial Z(t_f)}{\partial U} + \\ \sum_{j=0, j=j+P}^{N} & \sum_{k=0}^{P} \mu_k \frac{\partial L(w_1(Z(t_{k+j}), w_2(Z(t_{k+j}), t_{k+j})}{\partial Z(t_{k+j})} \frac{\partial Z(t_{k+j})}{\partial U}\end{aligned} \tag{3.23}$$

and the gradient of the trajectory constraints are

$$\frac{\partial S_i}{\partial U} = \frac{\partial S(w_1(Z(t_i)), w_2(Z(t_i)), t_i)}{\partial Z(t_i)} \frac{\partial Z(t_i)}{\partial U}. \tag{3.24}$$

The gradients of the trajectory constraints $(\Upsilon(U))$ initial time $(\psi_0(U))$ and final time constraints $(\psi_f(U))$ can be computed in a similar manner. The gradients

$$\frac{\partial Z(t_s)}{\partial U}$$

are the components of collocation matrix (3.18) at time t_s.

All results produced in this thesis use the nonlinear programming package NPSOL. NPSOL does not accept analytical Hessians; it uses the BFGS algorithm to estimate the Hessian.

The NTG software package is an implementation of the above stated algorithm. Matrix multiplication optimized for the structure of 3.18 is used throughout the package. The NTG software package requires the following information:

1. The Optimal Control Problem in equations (2.1), (2.2), and (2.3) transformed into the form of equations (3.14) through (3.17).

2. The maximum derivative that occurs in each output.

3. The gradients of equations (3.14) through (3.17) with respect to ξ.

4. The “active variables”. That is, an array indicating the which components of ξ are being used to compute the nonlinear cost and constraints.

5. B-spline properties of *each* output: knot point sequence, order, and smoothness.

6. Collocation points.

7. Type of quadrature used to approximate the integration (e.g. trapezoidal, Simpson, etc.).

3.4 Conclusion

There were two main components to this chapter. First, we derived an algorithm to map system dynamics to a lower dimensional space 3.1. This algorithm applies to differentially flat as well as non-flat systems. Second, we develop the main result of the thesis: the NTG algorithm.Finally, the NTG software package is described, which is an implementation of the NTG algorithm. The NTG software package is an efficient set of routines for the real-time solution of optimal, constrained trajectory generation problems that combines the elements of geometric control, B-splines, and nonlinear programming.

Chapter 4

NTG Performance Comparisons

In this chapter, several comparisons are used to assess the performance of the NTG software package. Version 2.3 of NTG using the sequential quadratic programming solver NPSOL will be used in all computations.

4.1 An Investigation of NTG, Differential Inclusion, and Collocation Integration Accuracy

In this section, we examine a problem investigated by Fahroo *et al.* in [28]. Fahroo revisits the paper by Conway *et al.* [22] which attempts to refute Seywald's paper [94] on differential inclusions. Fahoo's results agrees with Seywald in that using the differential inclusion transcription method appears to reduce the size of the nonlinear programming problem, without a loss of accuracy. Since the method of differential inclusions is philosophically related to our method, we will put our results along those of Seywald, Conway, and Fahroo.

The problem under consideration is a simple cart problem which admits an analytical solution (see Fahroo *et al.* [28]). The equations of motions are

$$\dot{x}_1 = x_2 \qquad \dot{x}_2 = -x_2 + u. \tag{4.1}$$

The cost function to be minimized is

$$J := \int_0^2 u^2 dt. \tag{4.2}$$

The optimal cost of .577678 is provided by Fahroo *et al.* in [28]. The initial conditions are

$$x_1(0) = 0,\ x_2(0) = 0, \tag{4.3}$$

and the final time constraint is

$$x_1(2) - 2.694528 x_2(2) + 1.155356 = 0. \tag{4.4}$$

It is obvious that if we select the output $z_1 = x_1$ the equations of motion in (4.1) are differentially flat. The output is parameterized with a B-spline with order and smoothness $k = 7$ and $s = 6$, respectively. The number of knot intervals is $l = 2$. The results of running NTG and the other transcription techniques are found in Table 4.1.

The comparison is made with number of parameters versed optimality and error in approximating the equations of motion. Since the system is differentially flat, there is no error in approximating the equations of motion as with the other techniques. Table 4.1 shows that the NTG approach performs well over other transcription techniques when one compares cost versus the number of parameters. Higher-order quadrature techniques are useful in NTG since the inputs are written in terms of higher derivatives of the output.

4.2 Comparison of NTG and RIOTS

In this section, a comparison will be made between NTG and RIOTS [90]. RIOTS uses the method of adjoints, outlined in Section 2.5.2, for the solution of optimal control problems. RIOTS performs well for systems without constraints. However, once constraints are added to the optimal control problem, the algorithm performance quickly degrades. Direct collocation was not considered comparative

Method	Cost	Number collocation intervals	Number parameters
Simpson collocation	.577668	5	18
Simpson collocation	.577682	20	63
Pseudospectral	.577679	5	18
Pseudospectral	.577678	20	63
Spectral Differential Inclusion	.577679	5	12
Spectral Differential Inclusion	.577678	20	42
NTG (Simpson quadrature)	.5776891	10	8
NTG (Simpson quadrature)	.5776787	20	8
NTG (Milne quadrature)	.5776780	5	8
NTG (Milne quadrature)	.5776779	10	8

Table 4.1: Summary of results of simple cart comparison showing that exploiting differential flatness compares favorably with other transcription techniques.

in this problem, so the results are omitted. A comparison measure was based on computation times and convergence from random initial conditions. All tests were conducted on a Sun Ultra 10 333 MHz computer.

The problem used for the comparison is the forced van der Pol oscillator. The cost, dynamics, and constraints of the problem are the following:

$$\min_u J(u) \doteq \frac{1}{2}\int_0^5 x_1^2 + x_2^2 + u^2 dt$$

subject to

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + (1 - x_1^2(t))x_2(t) + u(t) \\ x_1(0) &= 1,\ x_2(0) = 0,\ x_2(5) - x_1(5) - 1 = 0. \end{aligned}$$

The forced van der Pol oscillator is differentially flat with the output $z_1(t) = x_1(t)$ This was exploited when implementing this problem in the NTG code. The smoothness and order of the B-spline parameterization for each interval was taken to be three and five, respectively. The number of collocation points was chosen to be four times the number of coefficients.

For RIOTS, the input was parameterized by a second order B-spline for each interval. Trapezoidal integration was used in both software packages.

First, a comparison was made between CPU usage and the cost. Each point on the first plot of Figure 4.1 is the average cost and CPU time of 100 random initial guesses for the free variable coefficients in both RIOTS and NTG. The plot shows that as the number of coefficients representing the input in RIOTS and the output in NTG was increased, the lower the cost. RIOTS needed a minimum of 11 intervals for convergence from a random initial guess, while NTG needed only one interval.

The second plot in Figure 4.1 shows the trajectories at the lowest number of intervals that converged for both RIOTS and NTG. Table 4.2 shows that NTG's computation time is one eighth that of RIOTS with a 12 percent increase in cost for the minimum interval case.

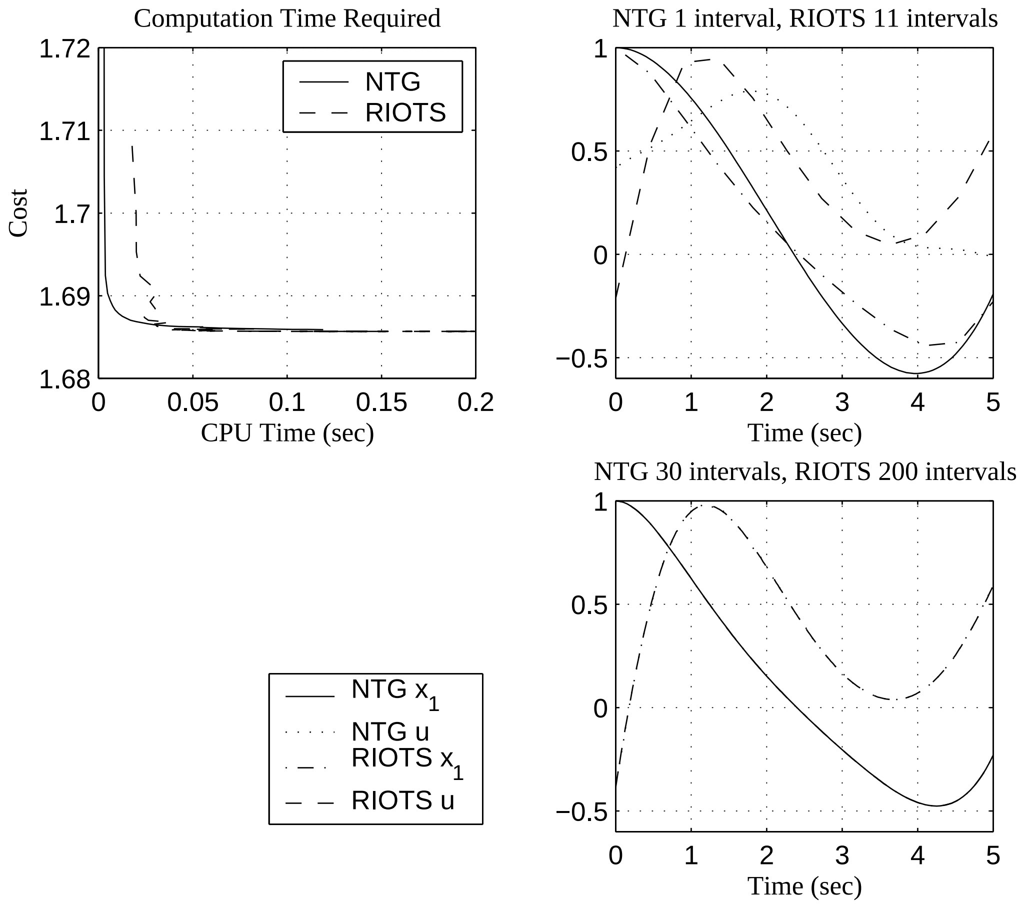

Figure 4.1: RIOTS and NTG van der Pol comparison

Method	CPU Time (s)	Intervals	Cost
NTG	.002	1	1.9127
RIOTS	.0178	11	1.7081
NTG	.1191	30	1.6859
RIOTS	.2261	200	1.6857

Table 4.2: RIOTS and NTG van der Pol comparison

The third plot in Figure 4.1 shows that both RIOTS and NTG converge to same cost for increasing numbers of coefficients. The results of this comparison show that for low intervals NTG can compute trajectories at significantly lower CPU times than RIOTS at comparative cost. For some real-time applications computing a feasible, albeit sub-optimal, trajectory may be necessary as a result of processing limitations.

4.3 An Investigation of the Accuracy of NTG vs. Shooting: The Goddard Problem

Seywald and Cliff in [95] solved the Goddard problem for different dynamic pressure constraints. Since they employ a combination of the Minimum Principle and the shooting techniques, there is no finite dimensional approximation to the solution. Therefore, the investigation in this section will be to determine how well the NTG solutions approximates the solution to the Goddard problem. We could not get RIOTS to converge for this problem.

The Goddard problem is stated as

$$\min_{u,T} J(u,T) := -h(T) \tag{4.5}$$

subject to

$$\begin{aligned}
\dot{V} &= \frac{1}{m}(u - D(h,V)) - \frac{1}{h^2},\ D(h,V) = \frac{1}{2}C_D A\rho_0 V^2 e^{\beta(1-h)} && V(0) = 0 \\
\dot{h} &= V && h(0) = 1 \\
\dot{m} &= -\frac{1}{c}u && m(0) = 1 \\
& 0 \le u(t) \le 3.5,
\end{aligned} \tag{4.6}$$

where $\beta = 500$, $C_D = 0.05$ and $A\rho_0 = 12400$. The endpoint constraint $m(T) = .6$ means that there is no more fuel left in the rocket and the trajectory constraint on the dynamic pressure is $\frac{1}{2}A\rho_0 V^2 \le 10$. See [95] for a complete variable description. Applying the NTG algorithm 3.1 shows that this problem is not differentially flat. Choosing the outputs $z_1 = m$, we obtain zero dynamics from algorithm 3.1:

$$\ddot{h} = \frac{1}{z_1}(\dot{z}_1 - D(\dot{h}, h)) - \frac{1}{h^2}. \tag{4.7}$$

The additional "pseudo output" $z_2 = h$ is necessary so that the zero dynamics are observable to the nonlinear programming problem. As a result of the Goddard problem not being differentially flat, the additional equality constraint in equation (4.7) must be included in the NTG problem. Both outputs will use an order $k = 5$ and a smoothness of $s = 2$.

Referring to Table 4.3, the difference in the optimal cost between the minimum and maximum number of knot intervals is approximately $1500m$. The equality constraint error was found by taking the maximum error in equation (4.7) over 2000 equally spaced intervals in $[0, T]$. Our results in Table 4.3 look accurate when comparing to the plot Seywald and Cliff provide in [94].

4.4 Variable Space Reduction

The research in this section has been the result of joint research with Nicolas Petit. A preliminary version of this material has appeared in [79]. In this section we provide numerical investigations for an example that exhibits an explicit rela-

Intervals	Cost	Maximum equality	Number
		constraint	collocation
	error	intervals	points
5	-1.012467	6.5e-3	40
10	-1.012661	8.4e-4	60
20	-1.012711	3.3e-4	80
25	-1.012713	1.0e-4	100
30	-1.012714	8.5e-5	120

Table 4.3: NTG Goddard example results showing the increase in accuracy in solution as the number of knot intervals is increased

tion between relative degree and computation time for a single-input single-output system. The relative degree of the system will directly relate to the amount of inversion used in the optimization problem. The computational implications of inversion are investigated. By example, we conclude that more inversion significantly increases the speed of execution with no loss in the rate of convergence.

For either open-loop reference trajectory design or receding horizon techniques, this example illustrates that the choice of adequate variables for representing a system and its dynamics is crucial in the context of implementation of real-time trajectory generation.

Classical collocation

A numerical approach to solving this optimal control problem is to use the direct collocation method outlined in Hargraves and Paris [41]. The idea behind this approach is to transform the optimal control problem into a nonlinear programming problem. This is accomplished by discretizing time into a grid of $N-1$ intervals

$$t_0 = t_1 < t_2 < \ldots < t_N = t_f \tag{4.8}$$

and approximating the state x and the control input u as piecewise polynomials $\hat{x}$ and $\hat{u}$, respectively. Typically a cubic polynomial is chosen for the states and a linear polynomial for the control on each interval. Collocation is then used at the midpoint of each interval to satisfy equation (2.1). Let $\hat{x}(x(t_1)^T, ..., x(t_N)^T)$ and $\hat{u}(u(t_1), ..., u(t_N))$ denote the approximations to x and u, respectively, depending on $(x(t_1)^T, ..., x(t_N)^T) \in \mathbb{R}^{nN}$ and $(u(t_1), ..., u(t_N)) \in \mathbb{R}^N$ corresponding to the value of x and u at the grid points. Then one solves the following finite dimension approximation of the original control problem in equation 2.1

$$\begin{cases} \min\limits_{y \in \mathbb{R}^M} F(y) = J(\hat{x}(y), \hat{u}(y)) \\ \text{subject to} \\ \dot{\hat{x}} - f(\hat{x}(y), \hat{u}(y)) = 0, \quad lb \leq c(\hat{x}(y), \hat{u}(y)) \leq ub, \\ \qquad \forall t = \dfrac{t_j + t_{j+1}}{2} \quad j = 1, \ldots, N-1, \end{cases} \tag{4.9}$$

where $y = (x(t_1)^T, u(t_1), \ldots, x(t_N)^T, u(t_N))$, and $M = \dim y = (n+1)N$.

Inverse dynamic optimization

Seywald in [94] suggested an improvement to the previous method (see also Bryson [10]). Following this work, one first solves a subset of system dynamics in equations (2.1) for the the control in terms of combinations of the state and its time derivative. Then, one substitutes for the control in the remaining system dynamics and constraints. Next, all the time derivatives $\dot{x}_i$ are approximated by the first order finite difference approximations

$$\dot{x}(t_i) = \frac{x(t_{i+1}) - x(t_i)}{t_{i+1} - t_i}$$

to get

$$\left.\begin{aligned} p(\dot{x}(t_i), x(t_i)) = 0 \\ q(\dot{x}(t_i), x(t_i)) \leq 0 \end{aligned}\right\} \quad i = 0, ..., N-1.$$

The optimal control problem is turned into

$$\begin{cases} \min\limits_{y \in \mathbb{R}^M} F(y) \\ \text{subject to} \\ p(\dot{x}(t_i), x(t_i)) = 0 \\ q(\dot{x}(t_i), x(t_i)) \leq 0. \end{cases} \tag{4.10}$$

where $y = (x(t_1)^T, \ldots, x(t_N)^T)$, and $M = \dim y = nN$. As with the Hargraves and Paris method, this parameterization of the optimal control problem in equation (2.1) can be solved using nonlinear programming.

The dimensionality of this discretized problem is lower than the dimensionality of the Hargraves and Paris method, where both the states and the input are the unknowns. This leads to substantial improvement in numerical implementation.

NTG Approach

In fact, it is usually possible to reduce the dimension of the problem further. Given an output, it is generally possible to parameterize the control and a part of the state in terms of this output and its time derivatives. In contrast to the previous approach, one must use more than one derivative of this output for this purpose. We propose the methodology outlined in the in Chapter 3, in particular, algorithm 3.1, that builds on the concept of differential flatness.

Comparisons

Our approach is a generalization of inverse dynamic optimization. Let us summarize the different ways we can write the optimal control problem:

- "Full collocation" solving problem (4.9) by collocating $(x, u) = (x_1, ..., x_n, u)$ without any attempt of variable reduction. After collocation, the dimension of the unknowns space is $\mathcal{O}(n + 1)$.
- "Inverse dynamic optimization" solving problem (4.10) by collocating $x =$

$(x_1, ..., x_n)$. Here the input is eliminated from the equation using one derivative of the state. After collocation the dimension of the unknowns space is $\mathcal{O}(n)$.

- "Flatness parametrization" (Maximal inversion), which is our approach, solving problem (2.1) using Algorithm 3.1. After collocation, the dimension of the unknowns space is $\mathcal{O}(n - r + 1)$.

4.4.1 Example

The example presented illustrates the benefits of transforming the optimization problem to the lowest space possible. The system under consideration is an academic example, without any particular physical meaning, chosen to contain various nonlinearities. Without loss of generality, its triangular form is chosen for the sake of simplicity of the presentation. By considering different outputs with increasing relative degrees results in different formulations of the optimal control problem from the full collocation to the flatness parameterization. Finally, runs done with these different approaches will be compared.

We consider the following fifth-order single-input dynamics

$$\begin{cases} \dot{x}_1 = 5x_2 \\ \dot{x}_2 = \sin x_1 + x_2^2 + 5x_3 \\ \dot{x}_3 = -x_1 x_2 + x_3 + 5x_4 \\ \dot{x}_4 = x_1 x_2 x_3 + x_2 x_3 + x_4 + 5x_5 \\ \dot{x}_5 = -x_5 + u \end{cases}$$

and the following optimal control problem: find $[0, 1] \ni t \mapsto (x, u)(t)$ that minimizes

$$J = \int_0^1 \left(x_1^2(s) + \frac{1}{100} u^2(s) \right) ds, \tag{4.11}$$

subject to the constraints

$$\begin{aligned}(x_1, x_2, x_3, x_4, x_5, u)(0) &= (0, 0, 0, 0, 0, 0)\\ (x_1, x_2, x_3, x_4, x_5, u)(1) &= (\pi, 0, 0, 0, 0, 0)\\ \forall i = \in \{1, 2, 3, 4, 5\}, |x_i| &\leq 100\\ |u| &\leq 100.\end{aligned}$$

To solve this problem by collocation, it is possible to use the three different approaches presented in the previous section

- "Full collocation". One must consider $(x_1, x_2, x_3,\ x_4, x_5, u)$ as unknowns.
- "Inverse dynamic optimization". For this example, we can solve for u by the following

  $$u = \dot{x}_5 + x_5.$$

  Thus the whole system variables are parameterized by $(x_1, x_2, x_3, x_4, x_5)$.
- "Flatness parameterization" (Maximal Inversion). Consider the variable x_4. Two differentiations give

  $$\begin{aligned}\dot{x}_4 =& x_1x_2x_3 + x_2x_3 + x_4 + 5x_5\\ \ddot{x}_4 =& 5x_2^2x_3 + (x_1 + 1)(\sin x_1 + x_2^2 + 5x_3)x_3\\ &+ (x_1 + 1)x_2(-x_1x_2 + x_3 + 5x_4)\\ &+ x_1x_2x_3 + x_2x_3 + x_4 + 5x_5 - 5x_5 + 5u. \qquad (4.12)\end{aligned}$$

  This system has relative degree 2, when x_4 is the output. By Result 1, it is possible to parameterize all the system by x_4 and 3 more variables. Here we can choose (x_1, x_2, x_3, x_4) for this parameterization.

  It is easy to check that when x_3 is the output the system has relative degree 3. The whole system can be parameterized by (x_1, x_2, x_3).

Choice of output	Relative degree	Variables for complete parameterization	Differential equations to be satisfied
u	0	$(x_1, x_2, x_3, x_4, x_5, u)$	5
x_5	1	$(x_1, x_2, x_3, x_4, x_5)$	4
x_4	2	(x_1, x_2, x_3, x_4)	3
x_3	3	(x_1, x_2, x_3)	2
x_2	4	(x_1, x_2)	1
x_1	5	(x_1)	0

Figure 4.2: The different formulations of the optimal control problem for the example. Top: full collocation. Bottom: flatness parametrization.

Similarly, when x_2 is chosen as the output, the system has relative degree 4 and it is possible to parameterize all its variables by (x_1, x_2).

At last, the system with x_1 as output has relative degree 5. The system is flat, i.e., it is possible to parameterize all its variables by x_1 only. In the optimal control problem, we can replace x_2, x_3, x_4, x_5 and u by combinations of $x_1, \dot{x}_1, \ddot{x}_1, x_1^{(3)}, x_1^{(4)}, x_1^{(5)}$.

The different formulations of the optimal control problem are summarized in Figure 4.2.

Results

Every solution was double checked by a numerical integration of the dynamics of the system. We only accepted valid solutions that satisfy $3.10 \leq x_1(1) \leq 3.20$. Choices of NTG parameters are motivated by this requirement and the desire for expedient execution. To perform as fair as possible comparisons we also give in each case the convergence rate, i.e., the percentage of runs ending up with an optimal solution satisfying the constraints. This is to prove that the use of inversion does not induce any particular degradation in terms of numerical sensibility.

In each case we simulated 200 runs with random initial conditions. All tests

Relative degree	Cpu-time (s.) (average)	Number of Variables (after collocation)	Rate of convergence (%)
0	70.0	90	79.5
1	52.2	76	92.5
2	25.7	65	85.0
3	5.1	43	99.5
4	1.7	29	91.5
5	0.50	14	92.0

Figure 4.3: Main results. The cpu-time is an exponential decreasing function of the relative degree. Top: full collocation. Bottom: flatness parametrization.

were conducted on a PC under Linux (Red Hat 6.2) with a Pentium III 733MHz processor.

The results detailed in Figure 4.3 show that the cpu-time is exponentially decreasing with the relative degree. The slowest problem is the one using full collocation. The fastest problem is the one that uses the flat output.

NTG internally invokes NPSOL, the Fortran nonlinear programming package developed by Gill *et al.* in [38]. NPSOL solves nonlinear programming problems using a sequential programming algorithm, involving major and minor iterations. At each major iteration a new quadratic programming (QP) problem is defined that approximates both the nonlinear cost function and the nonlinear constraints. This QP problem is solved during the minor iterations. The overall cpu-time required is highly correlated to the sum of all the minor iterations. Inspecting the runs, we concluded that the successive QP subproblems are generally well conditioned in all cases. In this example, each variable is represented by approximatively 15 coefficients. Therefore the number of variables is a decreasing function of the relative degree, see Figure 4.3.

It is known, see Gill *et. al* [37], that the cost of solving a well-conditioned QP problem grows as a cubic function of the number of variables. In Figure 4.4 one

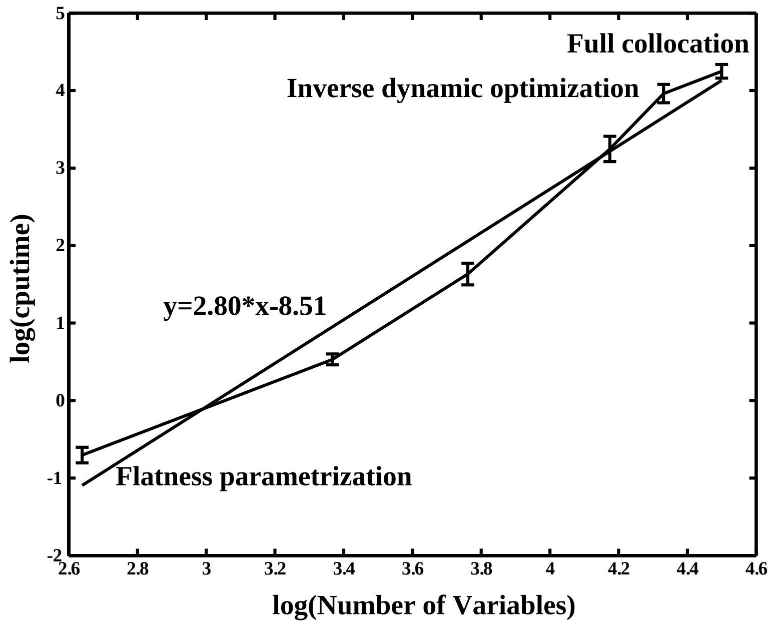

Figure 4.4: log(Number of variables) versus log(cpu-time). In each case 200 runs were done with random initial guesses. The *variance* of the results is represented by the error bar. The slope of the linear regression of the mean values of cpu-time is 2.80 .

can see that this is a good explanation of the differences in experimental cpu-time, the slope of the linear regression of the mean values of cpu-time versus the number of variables being 2.80 .

Planar Ducted Fan Parameterization Comparison

The planar ducted fan, as shown in Figure 2.1, will be used in the next comparison. The objective to move from equilibrium point to equilibrium point in minimum time, subject to a thrust vectoring input constraint of the form

$$0 \leq F_{X_b} \leq 17 \text{ and } -F_{X_b}/3 \leq F_{Z_b} \leq F_{X_b}/3.$$

The boundary conditions are the following:

$$x(t_0) = (*,*,*,*,\pi/2,0) \text{ and } x(t_f) = (*,*,*,*,\pi/2,0),$$

where $x(t) = (x, \dot{x}, z, \dot{z}, \theta, \dot{\theta})$ and $*$ can be either 1, 0, or -1. There are 6561 possible combinations of boundary conditions.

In order to account for the free final time variable, the planar ducted fan equations are scaled to yield

$$\begin{array}{rcl} mx^{''}\cos\theta - (mz^{''} - \xi^2 mg)\sin\theta & = & \xi^2 F_{X_b} \\ mx^{''}\sin\theta + (mz^{''} - \xi^2 mg)\cos\theta & = & \xi^2 F_{Z_b} \\ (J/r)\theta^{''} & = & \xi^2 F_{Z_b} \\ \xi^{'} & = & 0, \end{array} \tag{4.13}$$

where $x^{'}$ denotes $\frac{dx}{d\tau}$ and $\tau = t/\xi$.

Three different scenarios will be investigated in this comparison:

1. Collocation: The outputs are $z_1 = x, z_2 = z, z_3 = \theta, z_4 = F_{X_b}, z_5 = F_{Z_b}$,and $z_6 = \xi$. In order to use direct collocation, the states x, z, and θ were approximated with fourth-order B-splines and four intervals. Approximating the inputs F_{X_b} and F_{Z_b} with third-order B-splines and four intervals produced

the best results. The variable time ξ was approximated by a first-order B-spline with one interval. The resulting equality constraints were required to be satisfied at 20 equally spaced collocation points.

2. Partial Linearization: The outputs are $z_1 = x, z_2 = z, z_3 = \theta$, and $z_4 = \xi$. Sixth order B-splines with C^3 continuity across knot points and four intervals will be chosen for the first three outputs. A first-order B-spline with one interval is chosen to parameterize the final output.

3. Differentially Flat: The outputs are $z_1 = x+(J/rm)\cos\theta, z_2 = z-(J/rm)\sin\theta$, and $z_3 = \xi$. Seventh order B-splines with C^4 continuity across knot points and four intervals will be chosen for the first two outputs. A first-order B-spline with one interval is chosen to parameterize the final output.

In total, NTG has four trajectory constraints (three due to the constraint on the inputs and one due to the output selection). The number of B-spline coefficients was chosen such that the minimum time of each scenario was within 5% of one another.

Note: RIOTS was not included in this comparison since the problem is highly constrained and nonlinear. Single shooting based techniques, such as RIOTS, often do not work well for highly nonlinear constrained systems.

The point of this example is to compare the convergence of NTG with other transcription techniques. Since there are no guarantees of convergence for non-convex sequential quadratic programming based optimization techniques, it would be expected that any technique used in real-time application would need to be robust to the initial guess.

The simulations conducted to test convergence was the following: Choose 500 random initial guesses for NTG and 100 for direct collocation for the unknown free variables in each of the 6561 test cases and test for convergence. Figure 4.5 gives the results of the optimization. The first plot shows that for any given 6561 test case most of 500 initial guess converged to a solution using NTG. In fact, all of the 6561 test cases converged for more than 20 of the 500 initial guesses. On the other hand, the second plot in Figure 4.5 shows that most of the 6561 test cases did not converge for any initial guess using direct collocation. This test illustrates that it

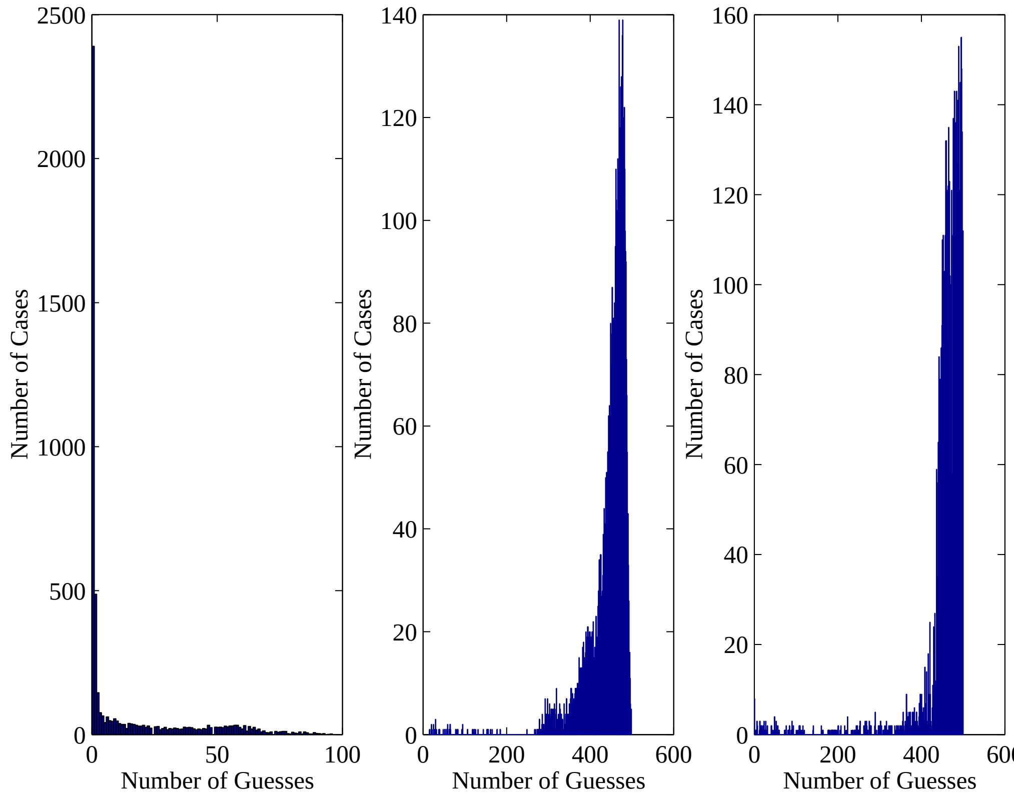

Figure 4.5: NTG direct collocation, “semi-flat” and “flat” convergence analysis. The abscissa is the number of convergent test cases out of the 500 for each 6561 test case. The ordinate shows the number of 6561 test cases.

is advantageous to parameterize an output in a lower dimensional space instead of parameterizing the inputs and the states when solving trajectory generation problems. The direct collocation technique was surprisingly worse than the other two parameterizations. The “semi-flat” test case converged in $2,991,107$ test cases and the “flat” parameterization converged in $2,924,928$ test cases out of a possible $3,280,500$. However, in the “flat” test case there exists a few of the 6561 test cases in which none of the runs converged. In the “semi-flat” test case at least 16 of the 500 initial guesses converged for each of the 6561 test cases. In general, the “flat” parameterization test cases computation time was lower than that of the "semi-flat" parameterization.

Remark 4.1. There are many possible causes for the fact the the "flat" parameterization convergence performance was comparable with the "semi-flat" test case. First, the forces are a very complicated expression of the derivatives of the outputs. NPSOL uses first-order information to approximate the Hessian. It may be necessary to use a nonlinear programming technique that uses analytical information. Second, the flat parameterization may not be the best one for some problems due to the fact that we added additional states for the dynamic compensator. Third, the flat parameterization may not be well suited to all optimal control problems. An optimal control problem that has a state constraint and a minimum energy cost may be more amenable to the "flat" parameterization.

Much more work needs to be done in developing a standardized methodology for comparing optimal control transcription techniques. Betts in [6] provides a measure of the complexity of a problem for a SQP method by

$$h_{SQP} = \frac{q}{n_g},$$

where q is the number of QP iterations and n_g is the number of gradient calls. Betts premise for this measure is that a problem is hard for an SQP method if the active set changes for one iteration to the next.

4.5 Conclusion

In this chapter we compared the NTG algorithm with the several popular techniques for numerically solving optimal control problems. We compared the accuracy of integration of NTG with collocation problems using the cart problem. The Goddard problem was used to investigate the errors associated with B-spline finite dimensional approximation to the solution as compared to an indirect method. Next, NTG was compared with adjoint method used in RIOTS using a measure of run-times and cost function minimization. Additionally, we choose outputs of a single-input-single-output system and the multi-input-multi-output (MIMO) planar duct fan example with varying relative degree. A relation showing that convergence rates and solution computation times were inversely related to the relative degree was observed. We concluded that results of the MIMO planar ducted fan needed further investigation and that the output parameterization may depend on the optimal control problem under considerations.

Chapter 5

Two Degree of Freedom and Receding Horizon Control (RHC) of the Caltech Ducted Fan

A preliminary version of a portion of this material has appeared in Milam *et al.* [70].

5.1 Experimental Setup and Mathematical Model

The Caltech Ducted Fan is an indoor flying, tethered representation of the longitudinal dynamics of a flight vehicle. In order to realistically emulate the longitudinal dynamics of a flight vehicle, a number of design considerations were taken into account. The dynamics of tethering, which constrain the operation of the ducted fan on a large cylinder, were designed in such a way that the overall system dynamics behaved like that of a flight vehicle from the ducted fan's point of view. Figure 5.1 shows an overview of the Caltech Ducted Fan testbed. The experiment consists of a vertical stand and a horizontal boom which holds a ducted fan and wing. This setup enables flight on a cylinder of height 2.5 m and radius 2.35 m. Because of the mass of 12.5 kg and a maximum thrust of only 14 N, a counterweight is attached to the boom via a cable and pulleys which reduces the effective weight to $mg_{eff} = 7\ N$. This allows the system to attain sizable vertical accelerations,

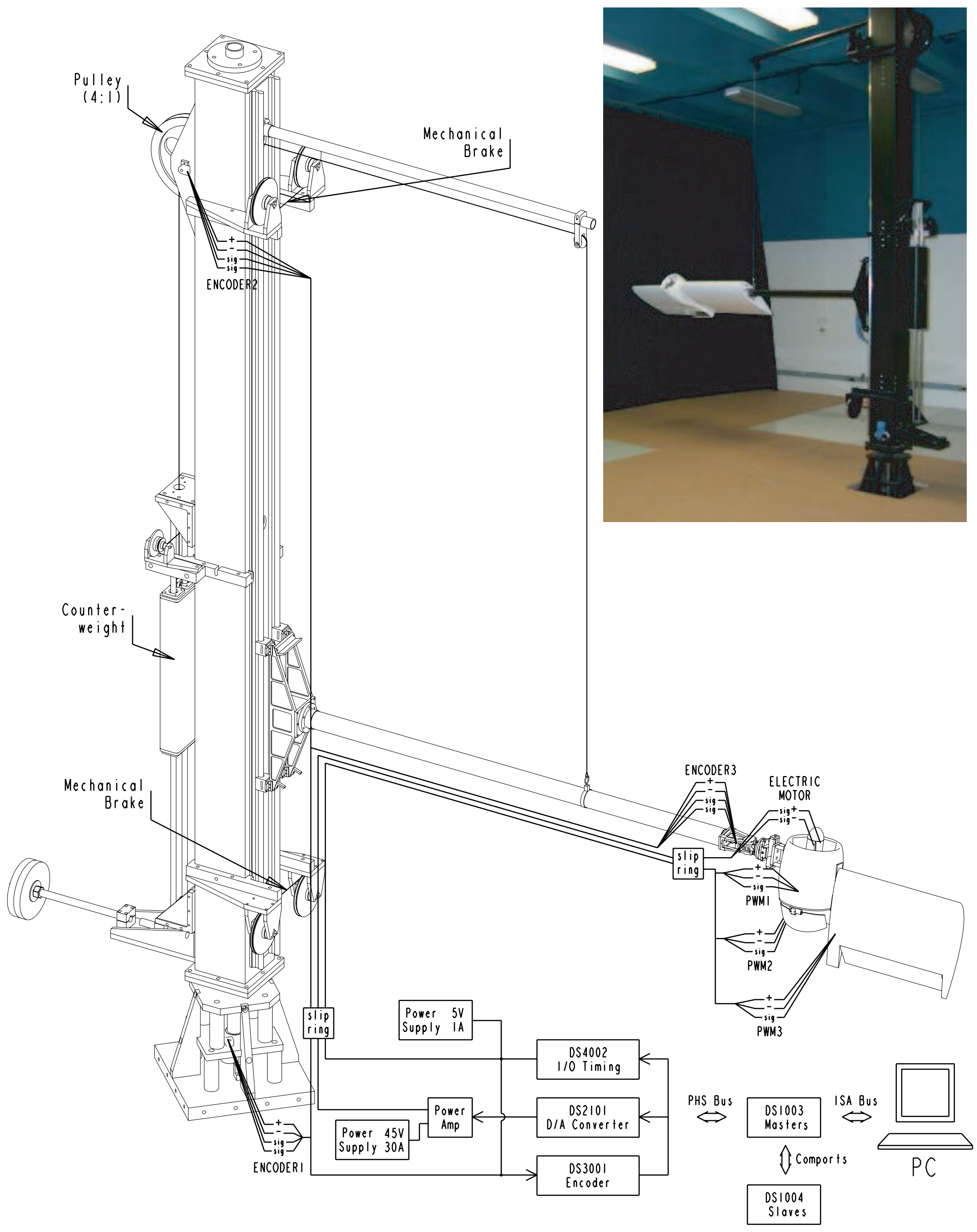

Figure 5.1: Ducted fan testbed

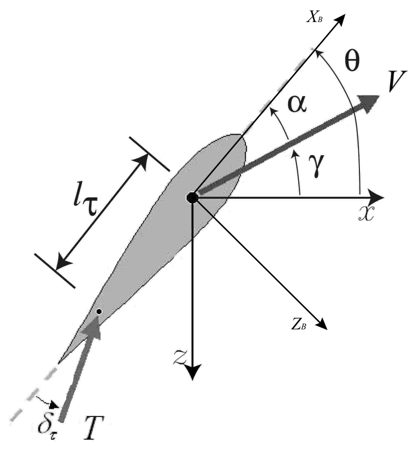

Figure 5.2: Ducted fan coordinate frames

while minimizing the force of potential crashes. Mechanical brakes in the vertical direction are used as well to aid in crash landings. The sensors are read and the commands are sent by a multi-processor system, comprised of a D/A card, an optical encoder card, a digital IO card, two Texas Instruments C40 signal processors, two Alpha processors (500 and 600 MHz), and a ISA bus to interface with a PC. The RHC control strategy used in this section resides on the 500MHz Alpha processor. Actuation of the ducted fan is accomplished in two ways: by controlling the current to the ducted fan, and by vectoring the resulting thrust via a servo controlled bucket. The bandwidth of the ducted fan motor is one Hertz and the bandwidth of the bucket servos are four Hertz.

Figure 5.2 depicts the inertial and body coordinate frames used in this section. In the inertial frame, the axes are fixed to the ground, and the x and z directions represent horizontal and vertical inertial translations. In the body frame, the X_B and Z_B axes are fixed to the vehicle. θ represents the rotation of the ducted fan about the boom axis. All three of these variables are measured via rotary encoders, and the resulting signals are routed to the computing platform via slip-rings.

A preliminary derivation of the equations of motion for the Caltech Ducted Fan experiment can be found in Milam *et al.* [71]. The equations of motion of the experiment are given by

$$\begin{aligned} m_x\ddot{x} + F_{X_a} - F_{X_b}\cos\theta - F_{Z_b}\sin\theta &= 0 && (5.1)\\ m_z\ddot{z} + F_{Z_a} + F_{X_b}\sin\theta - F_{Z_b}\cos\theta &= mg_{\text{eff}} && (5.2)\\ J\ddot{\theta} - M_a + \frac{1}{r_s}I_p\Omega\dot{x}\cos\theta - F_{Z_b}l_\tau &= 0, && (5.3) \end{aligned}$$

where

$$F_{X_a} = D\cos\gamma + L\sin\gamma,\ F_{Z_a} = -D\sin\gamma + L\cos\gamma$$

are the aerodynamic forces. We chose a spatial representation of the equations of motion so that we can consider both hover and forward flight modes. $J = .25\ kg\ m^2$ is the moment of inertia of the ducted fan about the boom, and $l_\tau = .35\ m$ is the distance from center of mass along the X_b axis to the effective application point of the thrust vectoring force. The "effective mass" in the X_I direction is $m_x = 8.5kg$ and the mass is $m_z = 12.5\ kg$. We call m_x an effective mass size it is actually derived by taking the Z_I inertia of the complete system divided by l_τ^2, Originally, we tried to make m_x and m_z equal so that the equations of motion of the ducted fan in body coordinates would look like that of a real aircraft. The parameter identification techniques presented in Franz *et al.* [34] revealed that m_x and m_z where actually significantly different. F_{X_b} and F_{Z_b} are thrust vectoring body forces; $I_p = 2e^{-5}\ kg\ m^2$ and $\Omega = 1300\ rad/s$ are the moment of inertia and angular velocity of the ducted fan propeller, respectively. The angle of attack α is related to the pitch angle θ and the flight path angle γ by $\alpha = \theta - \gamma$, where the flight path angle can be derived from the spatial velocities by

$$\gamma = \arctan\frac{-\dot{z}}{\dot{x}}.$$

The lift (L) ,drag (D), and moment (M) are given by

$$L = qSC_L(\alpha), D = qSC_D(\alpha),\ \text{and}\ M = \bar{c}SC_M(\alpha),$$

respectively. The dynamic pressure is given by $q = \frac{1}{2}\rho V^2$. The norm of the spatial velocity is denoted by V, ρ is the atmospheric density, and S is the surface area of the wing. The coefficients of lift ($C_L(\alpha)$), drag ($C_D(\alpha)$), and moment ($C_M(\alpha)$) were determined from a combination of wind tunnel and flight testing.

Figure 5.3 depicts the coefficients of lift ($C_L(\alpha)$) and drag ($C_D(\alpha)$) and the moment coefficient ($C_M(\alpha)$). These coefficients were determined from a combination of wind tunnel and flight testing and the parameter identification techniques presented in Franz *et al.* [34].

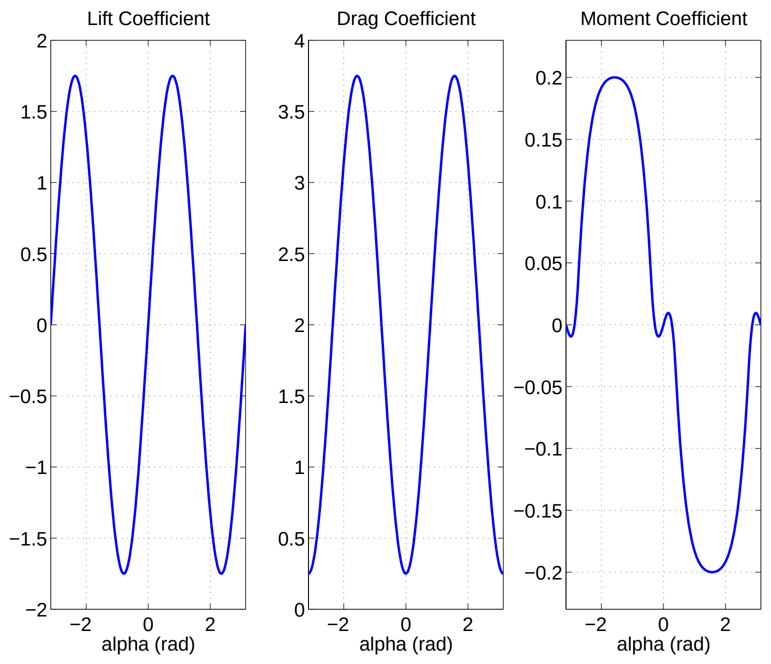

Figure 5.3: B-spline curve fits to wind tunnel and flight test results for the $C_L(\alpha)$,$C_D(\alpha)$, and $C_M(\alpha)$ aerodynamic coefficients.

Selecting the Outputs

We know from Example 2.1 that the planar ducted fan is differentially flat. However, it is unknown whether or not the planar ducted fan with aerodynamics is differentially flat. We apply Algorithm 3.1. Choosing the outputs to be $z_1 = x$

and $z_2 = z$, the algorithm terminates with a full rank decoupling matrix and the following zero dynamics:

$$\begin{aligned} J\ddot{\theta} - M_a(\dot{z}_1, \dot{z}_2), + \frac{1}{r_s} I_p \Omega \dot{z}_1 \cos\theta - \\ (m_x \ddot{z}_1 \sin\theta + m_z \ddot{z}_2 \cos\theta + \\ D(\theta, \dot{z}_1, \dot{z}_2) \sin\alpha(\theta, \dot{z}_1, \dot{z}_2) + \\ L(\theta, \dot{z}_1, \dot{z}_2) \cos\alpha(\theta, \dot{z}_1, \dot{z}_2) - mg_{\text{eff}}) l_\tau = 0 \end{aligned} \tag{5.4}$$

The additional "pseudo output" $z_3 = \theta$ will make the zero dynamics observable in the nonlinear programming problem. Equation (5.4) must be included as an equality constraint in the NTG problem formulation. When computing analytical gradients, care must be taken with the flight path angle γ to prevent a singularity at hover.

5.2 Two-degree-of-Freedom Design for Constrained Systems

5.2.1 Optimization Problem Formulation

A very aggressive optimization problem was chosen to be solved on-line: minimize time (T) subject to the trajectory constraints $0 \leq F_{X_b} \leq F_{X_b}^{max}$, $F_{X_b}^{max}/2 \leq F_{Z_b} \leq -F_{X_b}^{max}/2$, and $z^{min} \leq z \leq z^{max}$, the boundary constraints at the initial time

$$x(0),\ \dot{x}(0),\ \ddot{x}(0),\ z(0),\ \dot{z}(0),\ \ddot{z}(0),\ \theta(0), \dot{\theta}(0),\ \ddot{\theta}(0),$$

and boundary conditions at the final unknown time T

$$x(T), \dot{x}(T), z(T), \dot{z}(T), \theta(T), \dot{\theta}(T).$$

$F_{X_b}^{max} = 11$ N, $z^{min} = -1$ m, $z^{max} = 1$ m are the values of the constraints used for all the test results presented in this section. In the *two degree of freedom* design, the force constraints were chosen in the trajectory generation to be conservative so that the stabilizing controller has some remaining control authority to track the reference trajectory. The reason that minimum time was chosen as the objective was to make the trajectories as aggressive as possible. The boundary constraints on the initial time accelerations provide us with smooth inputs in the case when a new trajectory is computed away from an equilibrium. Our final boundary condition will always be an equilibrium.

5.2.2 Timing and Trajectory Management

Two different modes are considered in the experimental results: hover-to-hover and forward-flight. These modes may also be combined. In the hover-to-hover mode the user commands a desired position x_d and z_d via the joystick positions. Every t_{sample} seconds a new minimum time trajectory is computed from the boundary conditions t_{sample} seconds into the future to the desired equilibrium position given by current position of the joysticks. Equilibrium is defined for the hover-to-hover mode as being the desired translational position, zero velocities, $\theta = \pi/2$, $F_{Z_b} = 0$, and $F_{X_b} = 7$ N. The forward-flight mode is similar to the hover-to-hover mode except that the user commands the desired position in the vertical direction z_d and the desired spatial velocity $\dot{x}_d$. The equilibrium manifold is found by solving the resulting transcendental equations when $\dot{z} = 0$ in equation (5.1) . A plot of the equilibrium manifold is shown in Figure 5.4. Figure 5.5 shows the timing scheme used in the experiment. A higher level management function controlling which trajectory to stabilize about is necessary since there is the possibility of the algorithm not converging as well as excessive computation time in computing a trajectory. Before any optimizations have been computed, a nominal equilibrium trajectory is used, denoted by Traj_0. The first optimization is provided with the state and inputs of this nominal trajectory t_{sample} seconds in the future as an initial boundary condition and the equilibrium condition indicated by the joystick positions as the final boundary condition. If the optimization has finished successfully before t_{sample} seconds, at $t = t_{\text{sample}}$ the resulting trajectory is used and another optimization is triggered in the same fashion. If the optimization takes longer than

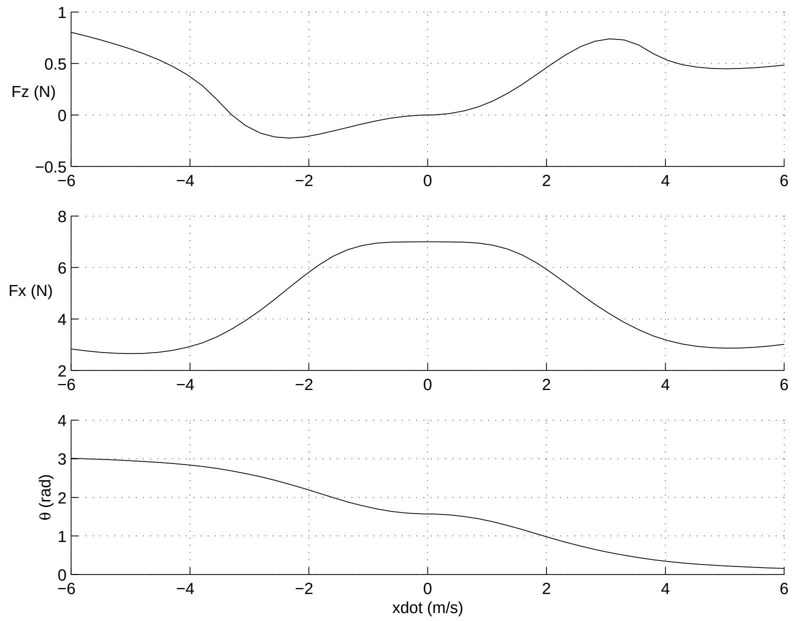

Figure 5.4: forward-flight mode equilibrium manifold

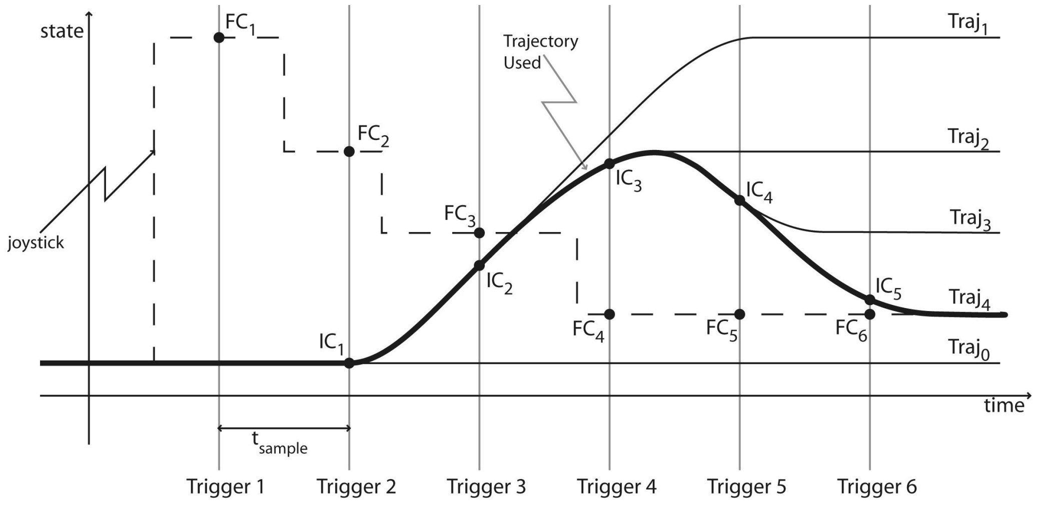

Figure 5.5: Sample run showing joystick input and timing concepts. The initial conditions for each run are denoted with IC, and the final conditions with FC.

t_{sample} seconds, the trajectory is truncated in the first $t_{\text{runtime}} - t_{\text{sample}}$ seconds to attempt to maintain continuity in the trajectory, but a small discontinuity may occur. For this reason, t_{sample} should ideally be longer than the expected run times of NTG. An important point is that the value of t_{sample} is not a constant. The reason is as follows: the initial bound for the next optimization must lie on a point on the last accepted trajectory where the differential equations are exactly satisfied. This is enforced at 21 points along each trajectory, but because of the variable horizon length (due to minimum time), the spacing of the points in time varies. For this reason, t_{sample} is chosen to coincide with the closest enforcement point, within some nominal sample time.

Due to the nature of the trajectory generation methodology used in this experiment, the convergence to an optimal solution is not guaranteed. Because of this, higher-level management logic also has to decide whether to use a given trajectory computed by NTG. The most obvious criterion to accept a trajectory is an indication of convergence returned by NTG. Other criteria include an upper bound on the acceptable run-time. For example, if the runtime is more than 10 percent longer than t_{sample}, the current trajectory generation computation is aborted. If the decision is made to reject a trajectory, the last accepted trajectory continues to be used and another optimization is triggered as usual. If the existing trajectory is exhausted before another one is accepted, the final equilibrium condition is continued as long as necessary. In hover, this simply means that x and z are kept at the desired values and all velocities are zero; in forward-flight, x is incremented with time according to the desired velocity and the vertical position z is kept at the desired value.

5.2.3 Stabilization around Reference Trajectory

Although the reference trajectory is a feasible trajectory of the model, it is necessary to use a feedback controller to counteract model uncertainty. There are two primary sources of uncertainty in our model: aerodynamics and friction. Elements such as the ducted fan flying through its own wake, ground effects and thrust not modeled as a function of velocity and angle of attack contribute to the aerodynamic uncertainty. The friction in the vertical direction is also not considered in the model. The prismatic joint has an unbalanced load creating an effective moment on the bearings. The vertical frictional force of the ducted fan stand varies with the vertical acceleration of the ducted fan as well as the forward velocity. Actuation models are not used when generating the reference trajectory, resulting in another source of uncertainty.

The separation principle was kept in mind when designing the observer and stabilizing controller. Since only the position of the fan is measured, the rates must be estimated. The gains were scheduled on the forward velocity.

The stabilizing LQR controllers were gain scheduled on pitch angle (θ) and the forward velocity. The weights were chosen differently for the hover-to-hover and forward-flight modes. For the forward flight mode, a smaller weight was placed on the horizontal (x) position of the fan compared to the hover-to-hover mode. Furthermore, the z weight was scheduled as a function of forward velocity in the forward-flight mode. There was no scheduling on the weights for hover-to-hover. The elements of the gain matrices for both the controller and observer are linearly interpolated over 51 operating points.

In Section 5.2.1, the optimal trajectory generation problem we outlined the optimal control problem we intended to solve. The three outputs $z_1 = x$, $z_2 = z$, and $z_3 = \theta$ will each be parameterized with four (intervals) , sixth order, C^4 (multiplicity), piecewise polynomials over the time interval scaled by the minimum time. The last output ($z_4 = T$), representing the time horizon to be minimized, is parameterized by a scalar. Choosing the outputs to be parameterized in this way has the effect of controlling the frequency content of inputs. Since the actuators are not included in the model, it would be undesirable to have inputs with a bandwidth higher than the actuators. There are a total of 37 variables in this optimization problem. The trajectory constraints are enforced at 21 equidistant breakpoints over the scaled time interval.

There are many considerations in the choice of the parameterization of the outputs. Clearly there is a trade between the parameters (variables, initial values of the variables, and breakpoints) and measures of performance (convergence, runtime, and conservative constraints). Extensive simulations were run to determine the right combination of parameters to meet the performance goals of our system.

5.2.4 "Shuttle" Maneuver

In this test the system was run in hover-to-hover mode. Two makers were placed 3/4 of a revolution apart (approximately 11 m if we unwrap the cylinder). The goal is to fly back and forth from marker to marker as many times as possible in 60 seconds by commanding the x and z positions with the joystick ($t_{\text{sample}} = 2.0$ sec). This type of highly aggressive maneuvering will test NTG's convergence properties since a new trajectory is computed starting from the last trajectory unless it is the first trajectory. The first trajectory is poor guess that does not satisfy the system dynamics. The results of the test are shown in Figure 5.6. We were able to finish nearly 4.5 cycles. This compares roughly with 3 cycles for a gain scheduled LQR controller. Note that we must be careful when making comparisons since the results are dependent on the skill level of the pilot. Figure 5.6 also shows a very large delay from the commanded position to the actual motion of the system due to the computation times (see Figure 5.8). On-board computation may limit the effectiveness of the two-degree of freedom design. Figure 5.8 shows all trajectories for the run, accepted or not. All trajectories are convergent. One trajectory that was accepted reached an equilibrium state.

5.2.5 Terrain Avoidance

In this section we present the results of two terrain avoidance maneuvers. To obtain hover-to-hover test data, the operator commanded a desired horizontal and vertical position with the joysticks with $t_{\text{sample}} = 1.0$ sec. By rapidly changing the joystick positions, NTG can produce very aggressive trajectories.

In the first test case, the ducted fan is requested to fly between, and then out of, two large peaks in the z state. The trajectory constraint is modeled by a B-spline in the NTG software. Figure 5.9 shows the position and orientation of the ducted fan. It is hypothesized that the tracking of the gain scheduled linear controllers

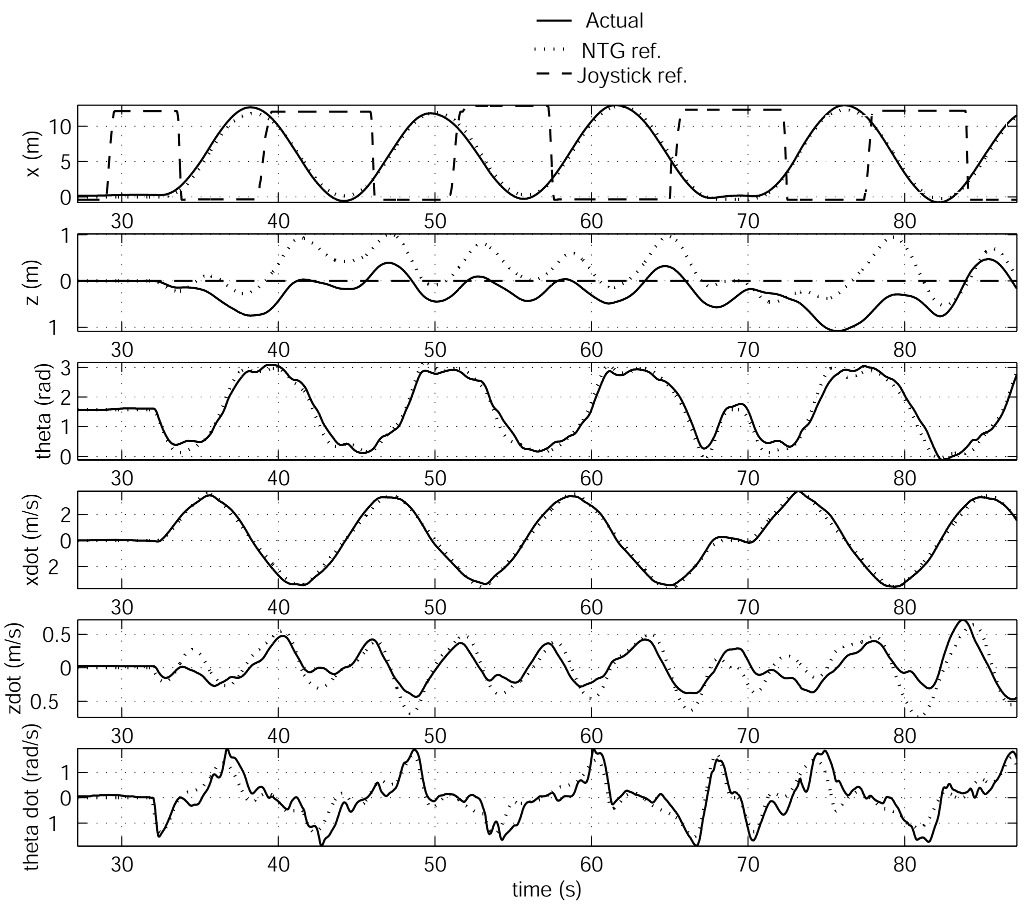

Figure 5.6: "Shuttle" maneuver depicting joystick command , NTG reference trajectory, and actual state

are marginal due the fact that the mass properties were not updated in the linear control design when the mass properties were updated in the reference trajectory design. (See Franz *et al.* [34].) In addition, the friction in the z direction is not modeled. Adding an integrator to the linear compensator would improve the performance in the z direction.

In the second test case, he top plot in Figure 5.11 shows z and x positions of the ducted fan and the typical z axis trajectory constraints. The maneuvers are created by holding the commanded z constant and changing the commanded x by the following sequence: $0 \mapsto 7.5 \mapsto 15 \mapsto 7.5 \mapsto 0$. (Note: these commanded positions

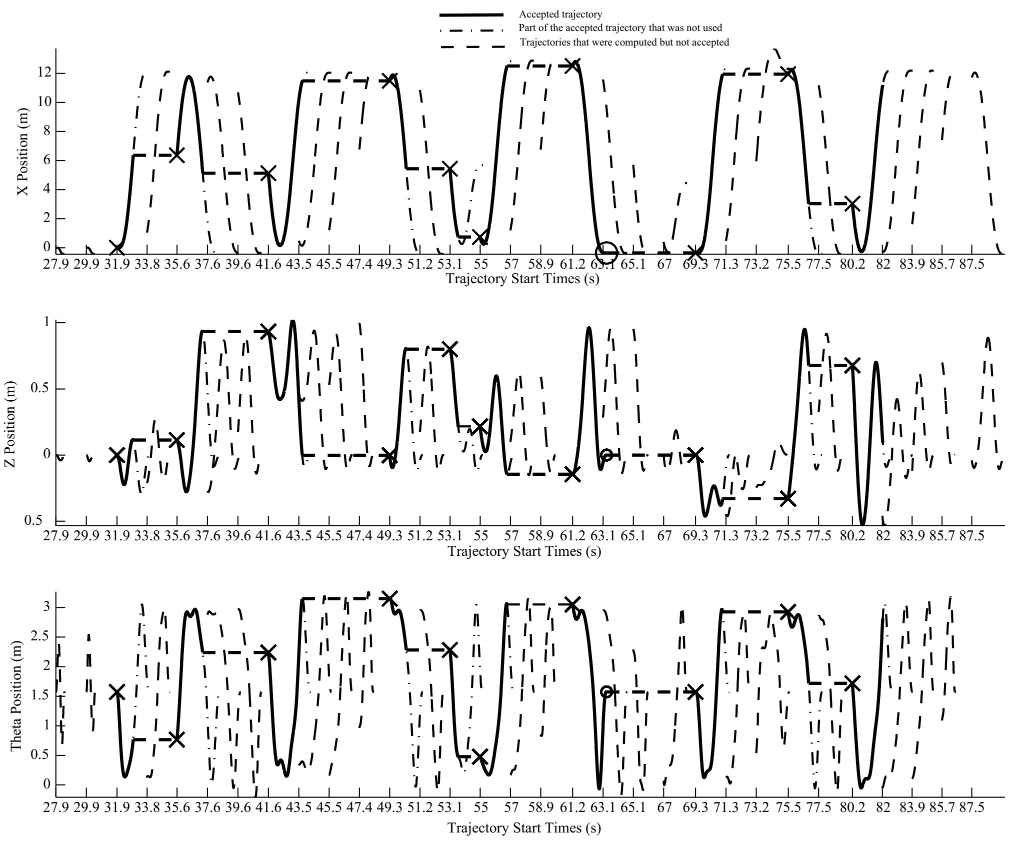

Figure 5.7: "Shuttle" maneuver depicting all the trajectories that are computed and which trajectories are accepted. For each trajectory, a fixed constant is added to the time axis so that all trajectories can be seen. "X" denotes the start of a trajectory. "O" denotes an accepted trajectory that was applied until completion; that is, the system reached a hover equilibrium.

are approximate.) These maneuvers were done over a time period of 60 seconds with a computation time on average of .7 seconds for 4 to 8 seconds of trajectory. Each of the 60 trajectories converged to a locally optimal solution. The bottom plot

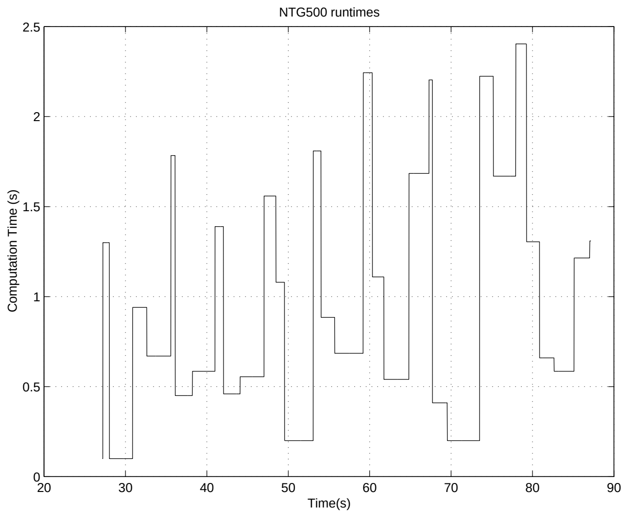

Figure 5.8: "Shuttle" Maneuver run times

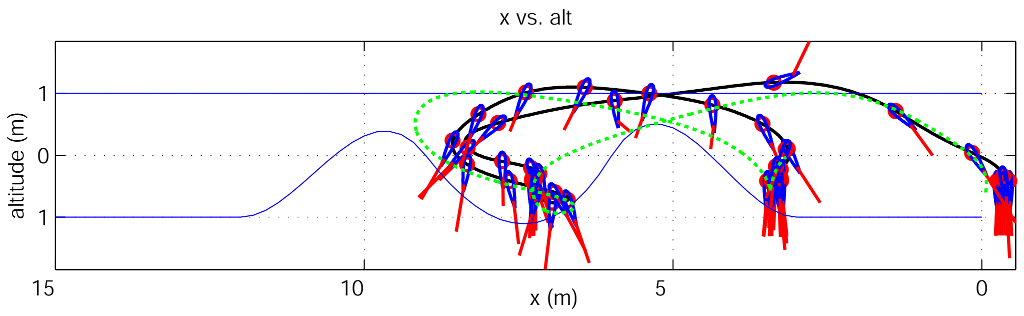

Figure 5.9: Terrain avoidance position and pitch attitude of the Caltech Ducted Fan. NTG reference trajectory (dashed) and actual (solid). The attitude (θ) is denoted by the orientation of the airfoil.

in Figure 5.11 corresponds to approximately the same changes in x but with some terrain added in real-time. There was no new initial guess provided to NTG when it was required to solve this optimization problem. The new terrain profile was also not tested off-line. These results give a reasonable argument for computing the trajectories on-line. It would be difficult to store trajectories for unknown threats and changes in terrain. All but one of the 60 trajectories converged to a

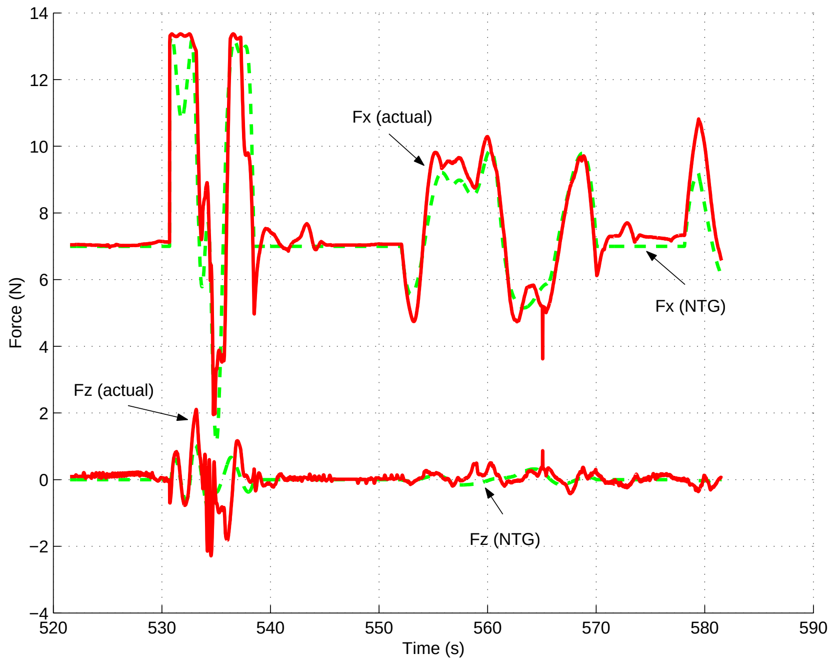

Figure 5.10: Terrain avoidance commanded and actual forces

local optimal solution. The one that did not converge was a result of asking the ducted fan to move to a position in violation of the trajectory constraints. The average computation time of each trajectory was approximately .8 seconds for 4 to 9 seconds of trajectory.

5.2.6 Forward-Flight Trajectory Generation

To obtain the forward-flight test data, the operator commanded a desired forward velocity and vertical position with the joysticks with $t_{\text{sample}} = 2.0$ sec. By rapidly changing the joystick positions, NTG produces high angle of attack maneuvers. Figure 5.12 depicts the reference trajectories and the actual θ and $\dot{x}$ over 60 sec. Figure 5.13 shows the commanded forces for the same time interval. The sequence of maneuvers in this plot are the following: First, the ducted fan transitions from near hover to forward-flight. Second, the ducted fan is commanded from a large forward velocity to a large negative velocity. Finally, the ducted fan is commanded

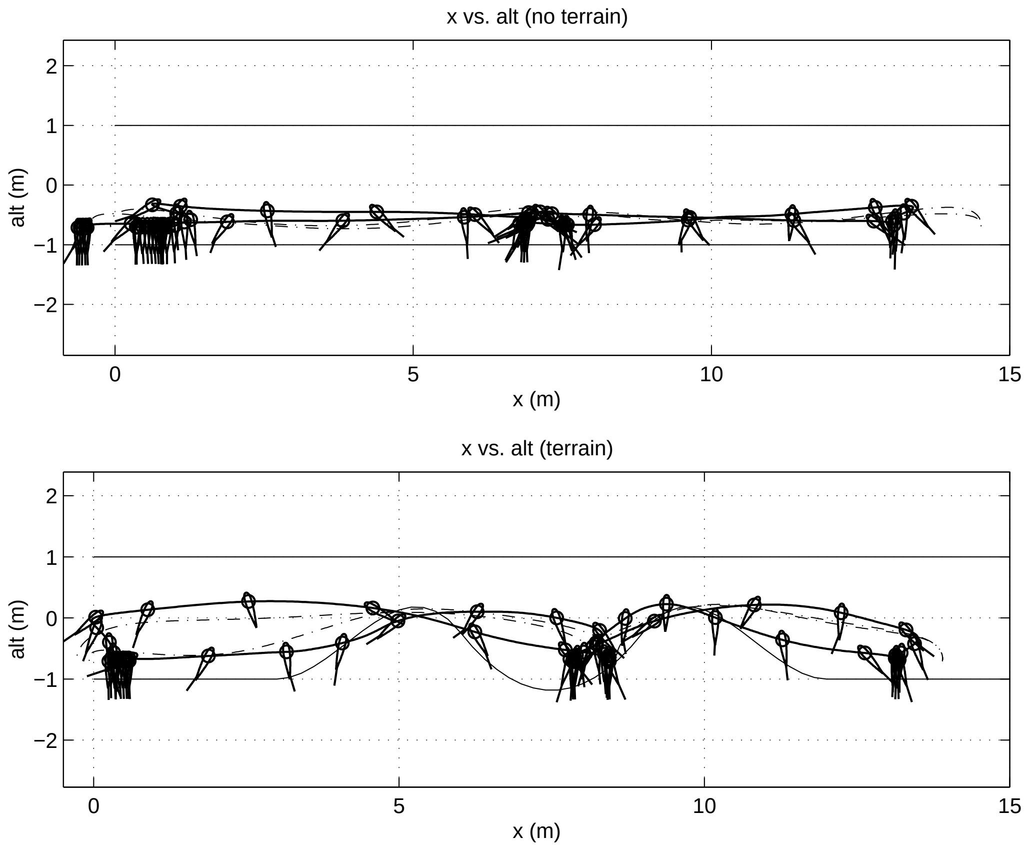

Figure 5.11: Hover to hover test case: Altitude and x position for two different vertical trajectory constraints. The actual (solid) and NTG reference trajectories (dashed). The attitude (θ) is denoted by the orientation of the airfoil.

to go to hover. Figure 5.15 is an illustration of the ducted fan altitude and x position for these maneuvers. The airfoil in the figure depicts the pitch angle (θ). It is apparent from this figure that the stabilizing controller is not tracking well in the z direction. This is due to the fact that unmodeled frictional effects are significant in the vertical direction. Figure 5.16 shows the run times for the 30 trajectories computed in the 60 second window. The average computation time is less than one second. Each of the 30 trajectories converged to an optimal solution and was approximately between 4 and 12 seconds in length. A random initial guess was used for the first NTG trajectory computation. Subsequent NTG computations used the previous solution as an initial guess. Much improvement can be made in determining a “good” initial guess. Improvement in the initial guess will improve not only convergence but also computation times.

The error in the equality constraint in equation (5.4) is shown in Figure 5.14. The equality constraint was required to satisfy to equality constraint within 0.1 N. The density of the collocation points will dictate how well the equality constraint is satisfied between collocation points.

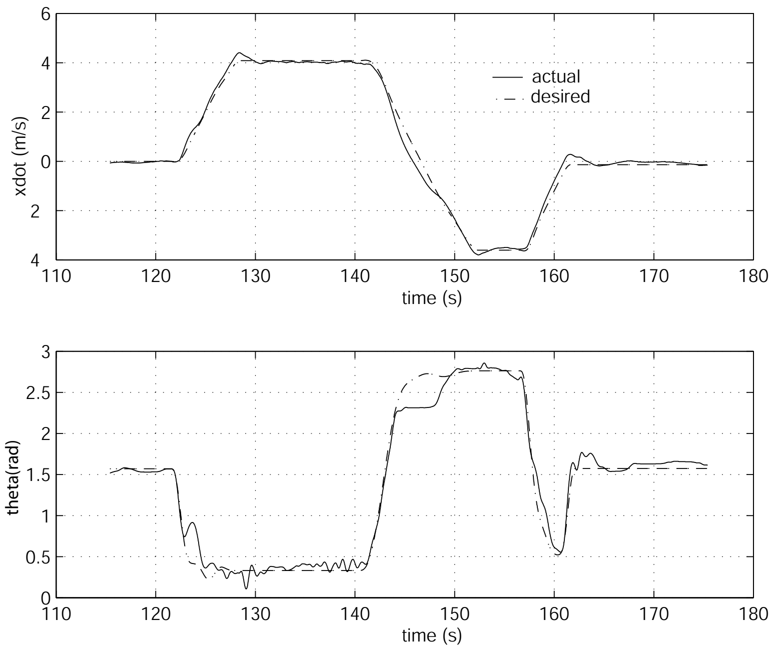

Figure 5.12: Forward-flight test case: θ and $\dot{x}$ desired and actual.

5.2.7 Conclusions and Future Work

In this section a methodology for real-time trajectory generation and validation of this approach with experimental results was presented. It was demonstrated that minimum time constrained trajectory generation is possible in real-time for two different flight modes on the Caltech Ducted Fan. In addition, it was illustrated that dynamically changing trajectory constraints can be taken into account in real-time.

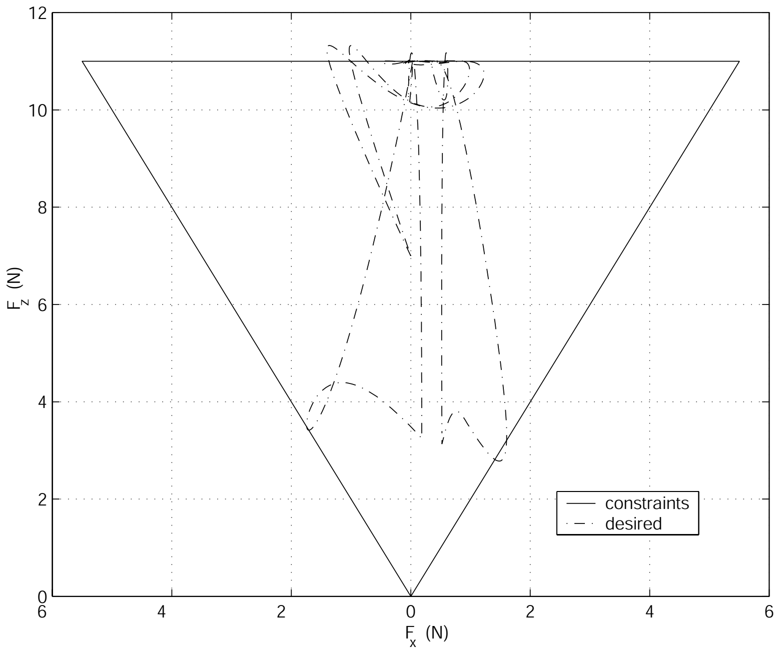

Figure 5.13: Forward-flight test case: Desired F_{X_b} and F_{Z_b} with bounds.

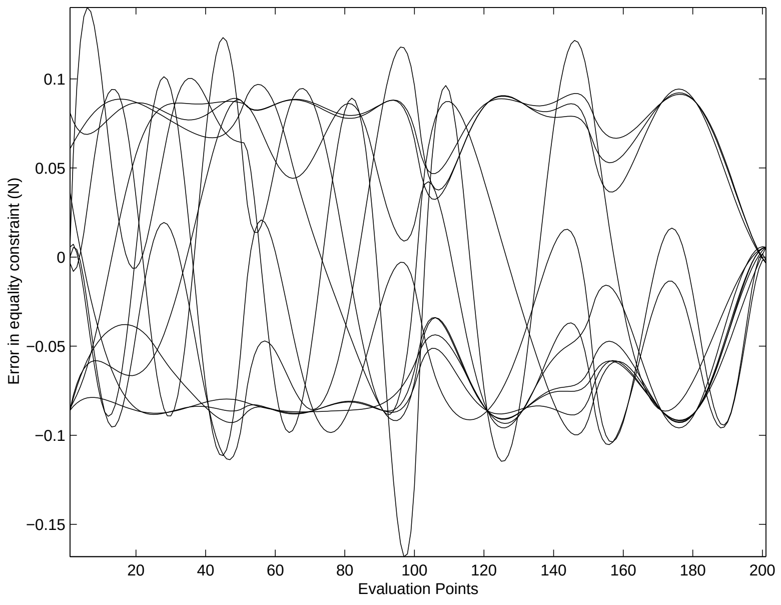

Figure 5.14: Forward-flight accepted trajectory errors

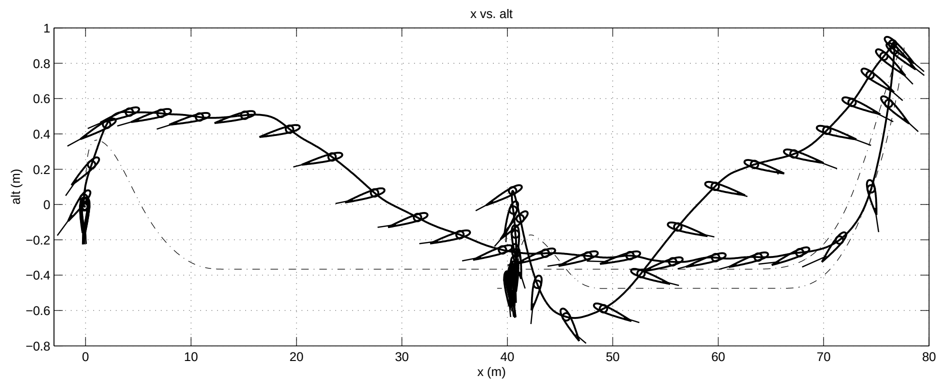

Figure 5.15: Forward-flight test case: Altitude and x position (actual (solid) and desired(dashed)). Airfoil represents actual pitch angle (θ) of the ducted fan.

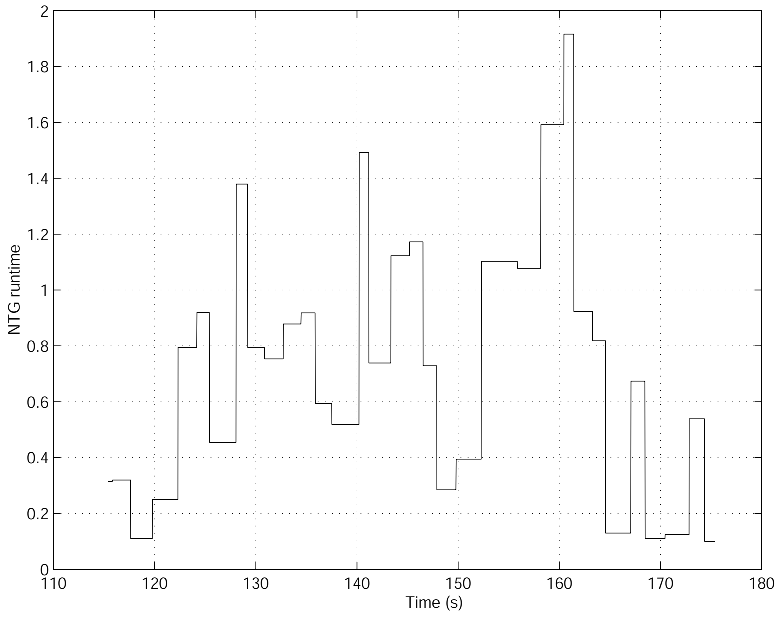

Figure 5.16: Forward-flight test case: 60 second run, 30 computed trajectories, t_{sample}= 2 sec.

For both the forward-flight and the hover-to-hover test cases, it was always assumed that the ducted fan could track the reference trajectory. Recall that the initial state of the reference trajectory starts from a point on the previous reference trajectory, not from the actual position of the fan. There may be circumstances in which the ducted fan cannot track the reference trajectory. In this case, one may want to update the reference trajectory using the current state of the ducted fan.

Developing a high confidence hierarchical control scheme is a direction of future research. In this work, confidence is achieved in our trajectory generation routine by defining a set of logic to manage the output from NTG. Standard measures of convergence and confidence need to be developed for on-line systems hosting algorithms that are not guaranteed to converge.

Along the same lines of a hierarchical control scheme is to develop different levels of trajectory generation. In our tests, it was noticed that trajectories could be any length from 1 sec to 25 sec with the same number of variables for each trajectory. It may be useful to have NTG determine trajectories using a kinematic model of the ducted fan at a high level and then determine trajectories at a lower level using a dynamic model. By doing this, it could be possible to get a more consistent length of trajectory for each computation.

Another topic for future research would be further development of on-line trajectory generation tools such as NTG. Developing a sequential quadratic programming routine designed specifically to run in real-time is a research goal. A sequential quadratic programming technique that incorporates an analytical Hessian and/or is based on the Interior Point Method are potential candidates to improve run-times and rates of convergence. B-splines are only one possibility of basis functions to use to parameterize the outputs. There may be other basis functions, such as rational B-splines, that better span the trajectory space of a system than B-splines and are more suitable for real-time computations.

5.3 Receding Horizon Control for Constrained Systems

The research in this section has been the result of joint research with Ryan Franz and John Hauser. A preliminary version of this material has appeared in Milam *et al.* [69] and Franz *et al.* [35].

In receding horizon control (RHC), an open-loop trajectory is found by solving a finite-horizon constrained optimal control problem starting from the current state. The controls of this trajectory are then applied for a certain fraction of the horizon length, after which the process is repeated.

Receding horizon control has found successful applications in the process control industry for some time, where dynamics are relatively slow. However, the algorithm demands tremendous computational power, and can exhibit poor convergent stability if not implemented properly. These difficulties have largely prevented its application to stability critical nonlinear systems with fast dynamics. Increasingly powerful and affordable computing facilities combined with better understanding of receding horizon control's stability properties have revived interests in this area. See Mayne *et al.* [67], Findeisen *et al.* [30] and the references therein for a good review of recent work in this field.

To implement the receding horizon control strategy, a constrained nonlinear optimization problem must be solved on-line. Due to the complexity of solving a nonlinear programming problem in real-time, the computational delay cannot be ignored. This is particularly important in aerospace applications, where the timescales of the vehicle dynamics (and the requisite control loops) are very short and comparable to the time required to solve a finite-horizon optimization problem.

The application of receding horizon control to aerial vehicles has been proposed and analyzed by several researchers. Representative examples include the mixed integer linear programming approach of Richards and How [88], the LMI framework for receding horizon control of Bhattacharya and Balas [8], and the work of Singh *et al.* [96], which provide simulation results for stabilization of an Unmanned Aerial Vehicle (UAV) about an open loop trajectory using receding horizon control.

The receding horizon strategy offers many benefits in this environment, such as the inherent ability to deal with constraints in the state and control. Examples of such constraints commonly encountered include static terrain obstacles, dynamic or pop-up threats and saturations on the actuators. However, these approaches would serve little practical purpose until stable and efficient computational techniques are developed to provide real-time solutions to the underlying constrained nonlinear optimal control problems.

The goal of the work in this section is twofold. The first goal is to address issues of implementation with substantial computation times and fast system dynamics and the second is to provide a validation of theoretical results through implementation on an actual nonlinear experiment. A full nonlinear model of the Caltech Ducted Fan including aerodynamics is used in order to test the viability of this technique on a flight platform. The results are among the first to demonstrate the use of receding horizon control for agile flight in an experimental setting using representative dynamics and computation.

This section is structured as follows: Section 5.3.1 provides theoretical background as well as some motivation for the choices made in terms of timing; Section 5.1 describes the actual experiment and its math model; Section 5.3.2 and 5.3.3 provides the detail of the RHC problem formulation and the description of the system used in NTG, respectively; Section 5.3.3 describes in detail the two different timing methods used in the experiment; and finally, Section 5.3.4 provides results before concluding.

5.3.1 Theoretical Background

Problem formulation

In RHC, the current optimal control $u(\cdot; y_k) \in [0T]$ for the current initial state y_k at time t_k is the solution to following optimal control problem with a scalar objective and constraints:

$$\begin{aligned}
\min_u \int_0^T \quad & q(y(\tau), u(\tau))\ d\tau + V(y(T)), \\
\text{s.t.} \quad & \dot{y}(t) = f(y(t), u(t)),\ y(0) = y_k, \\
& lb_0 \leq \psi(y(0), u(0)) \leq ub_0, \\
& lb_t \leq S(y(t), u(t)) \leq ub_t.
\end{aligned}$$

The vector $S(y(t), u(t))$ is a trajectory constraint (enforced over the entire time interval) while ψ is an initial time constraint. The control objective is to steer the state to an equilibrium point, usually the origin. No terminal constraint is enforced in this study. In theory, the resulting control $u(\cdot; y_k)$ is instantaneously applied until a new state update occurs, usually at a pre-specified sampling interval of time t_{sample} seconds. Repeating these computations yields a feedback control law.

Computational Delays

A major issue in the implementation of receding horizon control is handling the computational delay associated with the real-time optimization. We present here a summary of relevant theory which motivates our choices for timing made in the sequel.

Our system is described by

$$\dot{y} = f(y, u) + g(y, u, w), \tag{5.5}$$

where $f(\cdot, \cdot)$ is the nominal (i.e., model) system vector field and $g(\cdot, \cdot, \cdot)$ describes the effect of the external disturbance w, together with that portion of the system dynamics that is not explicitly modeled. Thus, for the purpose of control design, etc., we will use

$$\dot{x} = f(x, u) \tag{5.6}$$

as the *model* system. Now, suppose that

$$u = k(x) \tag{5.7}$$

is a state feedback that exponentially stabilizes the origin for the nominal system (5.6) and that $V(x)$ is a quadratic Lyapunov function proving such. For example, $k(\cdot)$ might arise as the solution to an infinite horizon optimization problem with $V(\cdot)$ as the corresponding minimum cost (to go). In the case that the perturbation is nonzero but can be bounded by a constant, one may use Lyapunov arguments to show that the state of the true closed-loop system (5.5), (5.7) will converge to a neighborhood of the origin.

Next we construct a sampled data feedback structure such that at every time $t_k := k\delta$ we obtain a measurement $y_k := y(t_k)$. At every time step, we calculate a trajectory $x(\cdot; y_k)$, $u(\cdot; y_k)$ by simulating the closed loop model system (5.6), (5.7) for a length of time (either δ or 2δ seconds).

We propose the following four methods for applying the resulting *open-loop* input trajectory to the actual system (5.5):

1. apply $u_{[0,\delta]}(y_k)$ (the control trajectory over the interval $t_{sim} \in [0, \delta]$ resulting from a simulation starting at y_k) over the interval $t \in [t_k, t_{k+1}]$. Note that implementing this option requires that the simulation be run in zero time.

2. apply $u_{[0,\delta]}(y_k)$ over the interval $t \in [t_{k+1}, t_{k+2}]$. Note that this option will always involve a delay.

3. apply $u_{[\delta,2\delta]}(y_k)$ over the interval $t \in [t_{k+1}, t_{k+2}]$.

4. apply $u_{[0,\delta]}(x(\delta; y_k))$ over the interval $t \in [t_{k+1}, t_{k+2}]$. Here $x(\delta; y_k)$ represents the state of the system x starting at y_k simulated ahead δ s.

When the system perturbation is identically zero $g(x, u, w) \equiv 0$, we see that options 1, 3, and 4 will be identical. Options 2, 3, and 4 are all implementable if the simulation computation can be completed in less than δ seconds (i.e., faster than real-time). Because option 2 involves a delay (even in the no perturbation case), we propose that 3 and 4 will be the best methods with non-zero run-times. Clearly, the performance of the sampled data system schemes with nonzero perturbation will depend on the sample time δ.

As a next step, suppose that we compute the input trajectory $u(\cdot; y_k)$ by solving the finite horizon optimal control problem

$$J_T^*(y(t_k)) = \min_{u(\cdot)} \int_0^T q(x(\tau), u(\tau))\, d\tau + V(x(T)), \tag{5.8}$$
$$\dot{x}(t) = f(x(t), u(t)),\ x(0) = y(t_k),\ lb_t \le S(x(t), u(t)) \le ub_t,$$

where the incremental cost satisfies $q(x,u) \geq c_q(\|x\|^2 + \|u\|^2)$ with $c_q > 0$ and $S(u(t), u(t))$ are constraints on the trajectory. If the terminal cost $V(\cdot)$ is chosen to be a control Lyapunov function (CLF) satisfying $\min_u((\dot{V}) + q)(x,u) \leq 0$ on a neighborhood of the origin, option 1 (with $g(x,u,w) \equiv 0$) is the receding horizon control scheme $\mathcal{RH}(T,\delta)$, analyzed in Jadbabaie *et al.* [48]. Now allowing $g(x,u,w)$ to be nonzero, we discuss some stability properties of this structure.

As in the stability analysis of unperturbed receding horizon control in Jadbabaie *et al.* [48], we will use $J_T^*(\cdot)$ as a Lyapunov function. Roughly speaking, we require that $J_T^*(y_k)$ be a strictly decreasing sequence, ensuring the convergence of the state to a (hopefully small) neighborhood of the origin.

Note that $J_T^*(\cdot)$ is Lipschitz continuous with constant K over the compact region of interest. The properties of $q(\cdot,\cdot)$ and $V(\cdot)$ ensure that

$$J_T^*(x(t_k+\delta)) \leq J_T^*(x(t_k)) - Q_\delta(x(t_k)), \tag{5.9}$$

where the decrement $Q_\delta(\cdot)$ is a positive definite function (given by integrating the optimal incremental cost over a δ second interval).

Suppose, now that we apply the same open loop control $u(\cdot)$ (e.g., the just computed optimal $u(\cdot)$) to the real and model systems, (5.5) and (5.6), with potentially different initial conditions. By a standard argument (using the Bellman-Gronwall lemma, see [51]), we have

$$\|y(t_k+\delta) - x(t_k+\delta)\| \leq e^{L\delta}\|y(t_k) - x(t_k)\| + \frac{b}{L}\left(e^{L\delta} - 1\right), \tag{5.10}$$

where b is a bound on $\|g(y(t), u(t), w(t))\|$, $t \in [t_k, t_k+\delta]$, and L is a Lipschitz constant for $f(\cdot,\cdot)$.

Combining (5.9) and (5.10) and noting that $y(t_k) = x(t_k)$, we obtain

$$\begin{aligned} J_T^*(y(t_k+\delta)) &\leq J_T^*(x(t_k+\delta)) + K\|y(t_k+\delta) - x(t_k+\delta)\| \\ &\leq J_T^*(y(t_k)) - Q_\delta(y(t_k)) + K\frac{b}{L}(e^{L\delta} - 1)\ . \end{aligned}$$

For small $\delta > 0$, we can bound the terms on the right according to $Q_\delta(x) \geq \frac{\delta}{2} c_q \|x\|^2$ and $K\frac{b}{L}(e^{L\delta} - 1) \leq 2Kb\delta$. We conclude that, for small δ, $J_T^*(y(t))$ will decrease provided that

$$\|y(t)\|^2 \geq \frac{4Kb}{c_q} \, . \tag{5.11}$$

This determines the *radius* r_b for an invariant sub-level set of $J_T^*(\cdot)$ to which the state of the true system will converge to under the scheme of option 1. Namely, r_b such that $\|x\|^2 \geq \frac{4Kb}{c_q}$ on $\Gamma^T_{\bar{r}(T)} \backslash \Gamma^T_{r_b}$, that is the annulus between the ball imposed by the error caused by g and the ball imposed by the finite horizon. A picture of this is shown in Figure 5.17. If g is zero, we have the same result shown in [47].

Finally, we extend this discussion to include situations in which $y(t_k) \neq x(t_k)$, which is the case in options 2, 3, and 4. In this case (5.10) necessarily contains an exponential (in δ) term multiplied by the error in the initial conditions. Performing analysis similar to that detailed above, we obtain the relation

$$\begin{aligned} J_T^*(y(t_k + \delta)) \; &\leq \; J_T^*(y(t_k)) - Q_\delta(y(t_k)) + K\tfrac{b}{L}(e^{L\delta} - 1) \\ &\quad + (K(1 + e^{L\delta}) + K_Q)\|y(t_k) - x(t_k)\|, \end{aligned} \tag{5.12}$$

where K_Q is a Lipschitz constant for $Q_\delta(\cdot)$. Clearly, mismatches in the initial conditions lead to performance degradations, including an enlargement of the terminal set (r_b increased) as well as potential destabilization. It is therefore of prime importance to minimize the initial condition mismatch to the extent possible. We conjecture that option 2 does a poor job of this; indeed, even in the no perturbation case such an error is induced by delay. Accordingly, we study options 3 and 4 both in simulation and experimentally on the physical system in an attempt to determine which will provide the best performance.

A final note is meant to justify the use of a "fast as possible" timing scheme, whereby δ is taken as the last computation time and thus is not constant. The reference [48] provides as a result the stability of $\mathcal{RH}(T, \{\delta_k\})$, where $0 \leq \delta_k \leq T$ and $\lim_{l\to\infty} \sum_{k=0}^{l} \delta_k = \infty$.

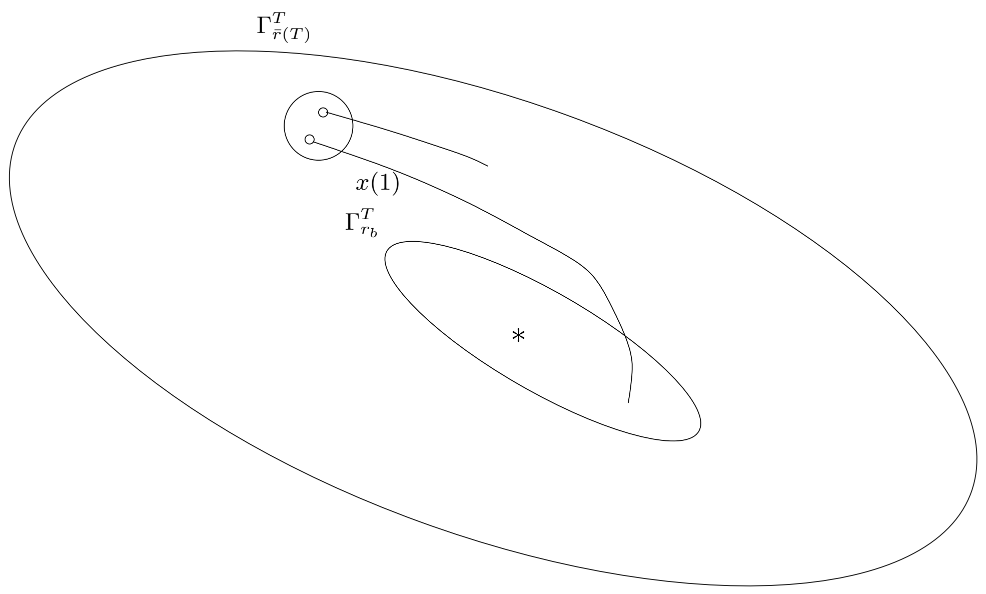

Figure 5.17: Illustration of sub-level set of $J_T^*(\cdot)$ to which the state of the true system will converge

5.3.2 Receding Horizon Control Formulation

We first state explicitly the cost functions used in this section, as defined in equation (5.8):

$$\begin{aligned}
q(x(t),u(t)) &= \frac{1}{2}x_{err}^T(t)Qx_{err}(t) + \frac{1}{2}u_{err}^T(t)Ru_{err}(t) \\
V(x(T)) &= \frac{1}{2}x_{err}^T(T)Px_{err}(T) \qquad (5.13)\\
x_{err} &\equiv x - x_{eq} = [x, z, \theta - \pi/2, \dot{x}, \dot{z}, \dot{\theta}]^T, \\
u_{err} &\equiv u - u_{eq} = [F_{X_b} - mg_{eff}, F_{Z_b}]^T, \\
Q &= \mathrm{diag}[4, 3, 15, 3, 4, 0.3], \\
R &= \mathrm{diag}[0.5, 0.5],
\end{aligned}$$

where the equilibrium point of interest is hover:

$$x_{eq} \equiv [0, 0, \pi/2, 0, 0, 0]^T, \quad u_{eq} \equiv [mg, 0]^T.$$

We choose Q and R to be the same as weights used to generate LQR gains with good performance, and P to be the corresponding stabilizing solution to the algebraic Riccati equation resulting in a CLF terminal cost around a hover equilibrium.

The sole trajectory constraint on the state is $-.9 \leq z \leq .9$. The input constraints are

$$\begin{bmatrix} 0 \\ -F_{X_b}^{max}/2 \end{bmatrix} \leq \begin{bmatrix} F_{X_b} \\ F_{Z_b} \end{bmatrix} \leq \begin{bmatrix} F_{X_b}^{max} \\ F_{X_b}^{max}/2 \end{bmatrix}, \tag{5.14}$$

where $F_{X_b}^{max}$ is 13 N and mg is 7.0 N.

5.3.3 NTG Setup

For our system we will choose as outputs $z_1 = x(t)$, $z_2 = z(t)$, $z_3 = \theta(t)$ in solving the problem posed in equation (5.8).

By choosing this parameterization, the equality constraint in equation (5.4) will need to be satisfied over the entire trajectory. In the case of only forward-flight, it would be possible to choose a parameterization that contains no equality constraints.

The optimal control problem is set up in NTG code by parameterizing the three position states (x, z, θ), each with 8 B-spline coefficients. Over the receding horizon time intervals, 21 collocation points were used with horizon lengths of 2.0 seconds. Collocation points specify the locations in time where the differential equations and any constraints must be satisfied, up to some tolerance.

Timing and Optimization Formulation

The choice of $x(0)$ for the optimization is dictated by the choice of timing scheme. We use two different strategies, corresponding to options 3 and 4 in Section 5.3.1, for choosing these initial constraints.

In some applications of receding horizon, run-times are insignificant compared to the dynamics of the system. This is not the case on most aerial platforms with current computing power. On our hardware we were able to achieve run-times between $0.1s$ and $0.3s$ in most cases; we ordinarily run linear controllers at

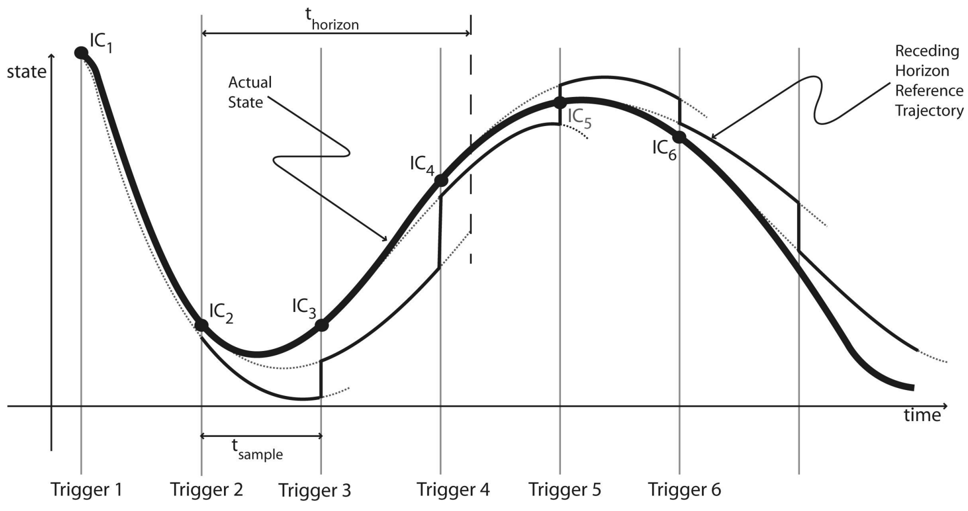

Figure 5.18: Illustration of timing scheme without prediction.

a minimum of $50Hz$. Because of this, the discussion in Section 5.3.1 is crucial.

Option 3: No Prediction

The first scheme for choosing the initial constraints in the state is the simplest, as it involves no model prediction. Whenever a computation is triggered, the current state of the system is given as the initial constraint on the state trajectory for the optimization problem. By the time the computation is finished t_{sample} seconds later, however, the idea is that the system has changed significantly. To attempt to use a valid control, we simply discard the first t_{sample} seconds of the trajectory, hoping that the resulting start point will coincide roughly with where we were in the previous trajectory. Fig. 5.18 shows graphically how this process works on one of the states. In this case, the controls corresponding to the line labeled "Receding Horizon Reference Trajectory" are applied to the system. Note that the figure exaggerates certain things for illustration. For example, the horizon length $t_{horizon}$ is in reality much longer than t_{sample}.

In our implementation, t_{sample} can either be set to some constant, or the computations can be run as "fast as possible", meaning a new computation is triggered immediately after the last one has finished. In this case, t_{sample} varies with the

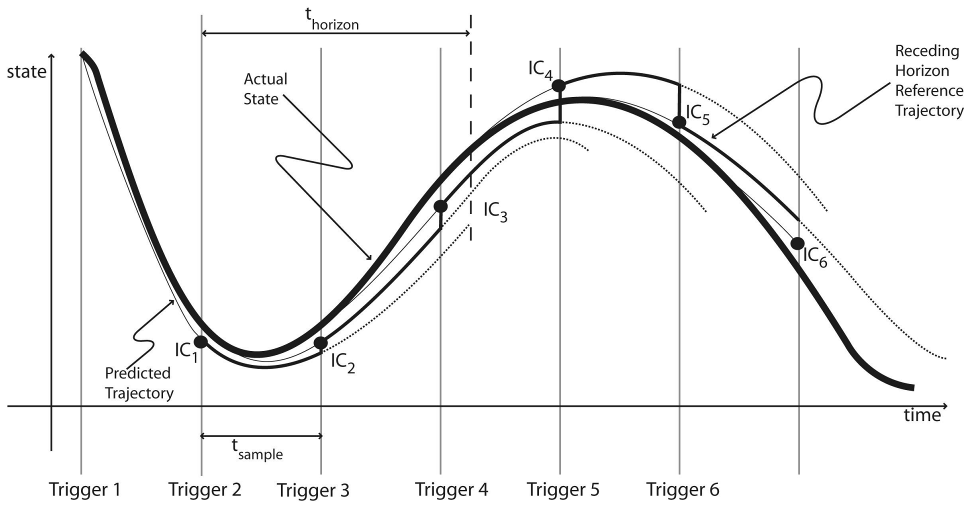

Figure 5.19: Illustration of timing scheme with prediction.

runtime.

Option 4: With Prediction

The second scheme we examine attempts to minimize discontinuities by using prediction. When a computation is triggered, the current state of the fan is first used as the initial condition for a simulation in which the control trajectory of the previous computation is used as input. This simulation is run for some amount of time t_{sim}. If a If fixed period is being used, t_{sim} is simply equal to the t_{sample}. if a “fast as possible” rule is used, t_{sim} is taken as an average of the past n runtimes. After the simulation is completed, the final values are passed as the initial constraints to the optimization. The resulting trajectory is output from the beginning. Figure 5.19 shows this process graphically. Again, the controls corresponding to the line labeled “Receding Horizon Reference Trajectory” are applied to the system.

Further Considerations

As with any timing scheme, there are necessarily discontinuities in the resulting control due to model mismatch and a non-zero sampling period. Early experience showed that some effort in minimizing these jumps was worthwhile. Accordingly, we use an inequality constraint on each optimization to achieve smoother control signals, $|u_{k+1}(0) - u_k(t_{sample})| < a$ for some a. If a fixed period is used, t_{sample} is simply equal to the period. If a "fast as possible" rule is used, t_{sample} is taken as an average of the past n run-times. This approach is compatible with both timing schemes discussed above. Graphically, control trajectories always start near the previous trajectory.

Another consideration involves non-convergent trajectory computations. Unfortunately, not all trajectory computations are guaranteed to converge. Each computation is given the last computed trajectory as an initial guess, which is sometimes not good enough. Also, some combinations of initial constraints and cost function are simply degenerate. If a computation returns certain signs of failure, the last good trajectory is simply continued and another computation is triggered. This will certainly fail if non-convergence happens frequently or repeatedly, as it has the effect of greatly increasing the sample time. In practice, this has not been a problem. The issue of non-convergence and state constraints will be discussed further in the results Section 5.3.4.

A characteristic of the spline representation used to solve the optimal control problem is that, between enforcement points, the values of the states, their derivatives, and the controls may not be consistent with the equations of motion for the system. Because of this, a point on the trajectory is, in general, not suitable as an initial equality constraint for a successive computation. Nevertheless, experience showed us that some sort of effort in minimizing large jumps in at least the forces is worthwhile. To deal with this, we introduce a degree of freedom on the accelerations by eliminating their initial constraints. We are most interested in minimizing jumps in the controls, so we enforce an inequality constraint $|u_{k+1}(0) - u_k(t_{sample})| < a$ for some a. If a fixed period is used, t_{sample} is simply equal to the period. If a "fast as possible" rule is used, t_{sample} is taken as an average of the past n run-times. This approach is compatible with both timing schemes discussed above; graphically, control trajectories always start near the

horizon	predict	no predict
$1.0s$	$0.4s$	$0.15s$
$1.5s$	$0.5s$	$0.2s$
$2.0s$	$0.65s$	$0.3s$
$2.5s$	$0.6s$	$0.4s$
$3.0s$	$0.5s$	$0.4s$

Table 5.1: Maximum acceptable periods as determined in simulation

previous trajectory.

5.3.4 Results

Timing Method Selection

We investigate through simulation and experimental testing the timing method and horizon length to use for our results. Table 5.3.4 shows results of identifying the highest acceptable periods for different combinations of timing mode, horizon length. The simulation allows us to explore many different configurations without fear of damaging the hardware. The test used for these results was a 20 m step in x, a fairly demanding request which puts the fan into a forward-flight state to test out the full features of the model. We were unable to design a gain-scheduled LQR controller which could perform this maneuver in an acceptable fashion. Acceptable results were chosen as stable and with few qualitative differences from the best results.

Next, Table 5.2 shows horizon lengths and timing methods that were acceptable on real experiment. One difference from these runs for the simulation was that on real experiment we used a smaller step of $5m$ in x in order to prevent damage to the apparatus. Another difference from the simulation is that the fixed period chosen is only a lower bound on the actual period. The majority of calculations remain below the fixed period in all the runs, but there are still some which exceed the value due to limited computing power. The prediction timing method produced

horizon	predict	no predict
$1.0s$	$0.0s$	$0.2s$
$2.0s$	$0.1s$	$0.2s$
$3.0s$	$0.2s$	$0.2s$

Table 5.2: Maximum acceptable periods as determined on the real experiment

larger run-times on average, and appeared more sensitive to the model used in NTG. The no prediction timing method with a $2s$ horizon running in "fast as possible" mode was chosen for the example test cases shown in this section. A thorough investigation of appropriate horizon times for the Caltech Ducted Fan can be found in Dunbar *et al.* [26].

Ducted Fan Flight Test Results

In this section we present the result of commanding a large change in the equilibrium of the system using the cost and constraints in equation (5.13). This aggressive command results a highly nonlinear motion of the system.

The two test cases that are investigated are aggressive maneuvering, using a series of step commands, and operation of the ducted fan near a state constraint. The desired commands to the experiment are input with joysticks. They are set up so that the user can change in real-time the x and z equilibrium positions of the experiment.

The first test case is an 11 m step command in x followed by an -11 m step. Figure 5.20 shows an animation of the translation and rotation of the ducted fan as well as the angle of the thrust vectoring bucket and the force being applied on the system. The commanded forces are depicted in Figure 5.21. The RHC at t_k is denoted by a dotted line. The insert picture illustrates with a solid line the portion (t_{sample}) of the RHC control that is being commanded to the experiment. The allowable jump in the control at t_{k+1}, given by $|u_{k+1}(0) - u_k(t_{sample})|$, was bounded by .25 N. Figure 5.22 illustrates that the system quickly responds to the

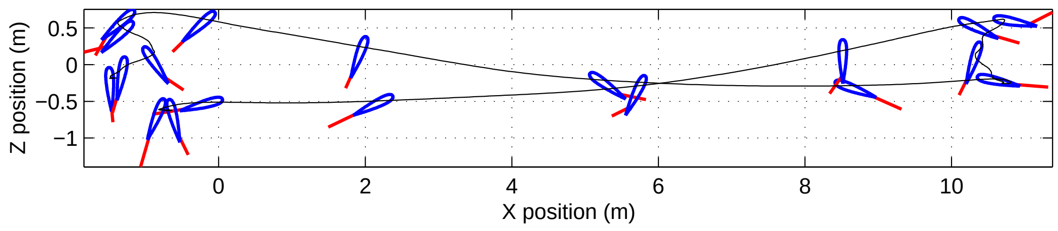

Figure 5.20: Plot depicting the actual attitude and position of the ducted fan throughout both step commands.

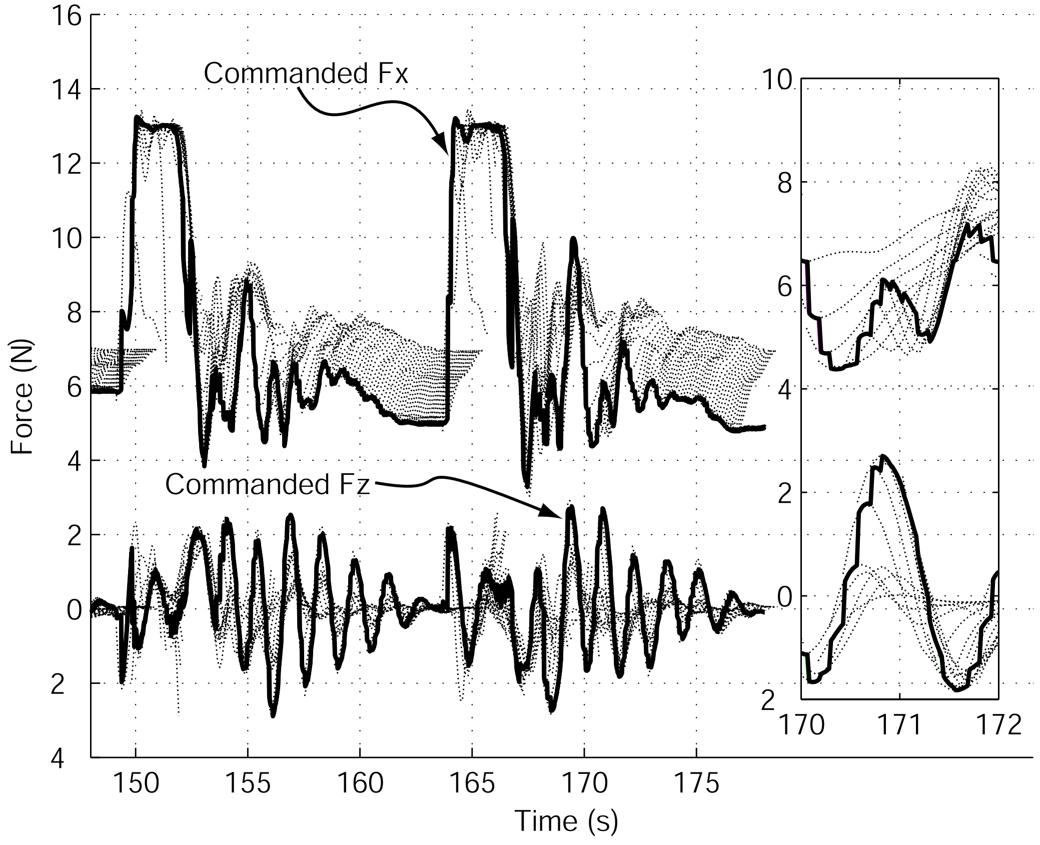

Figure 5.21: The commanded forces in the body frame. There is a nonlinear transformation between F_{X_B}, F_{Z_B} and commanded current to the motor and the thrust vector bucket angle δ_τ.

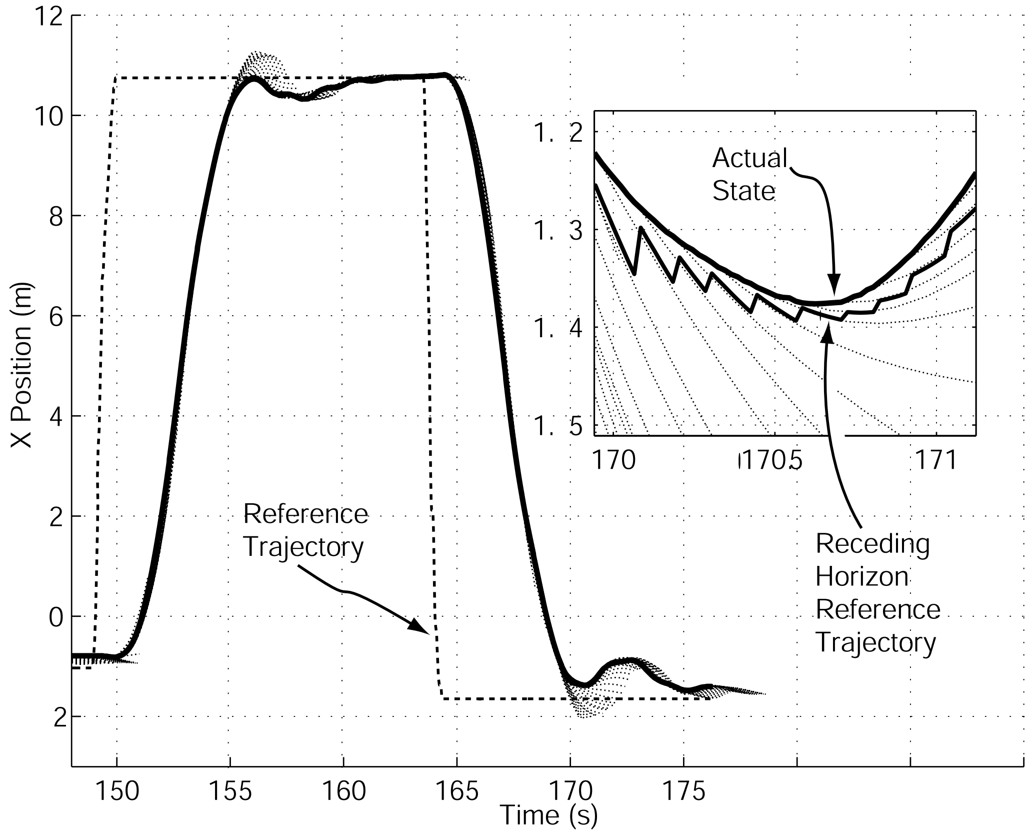

Figure 5.22: Horizontal position of the ducted fan

step commands and then settles at the desired commanded location. The receding horizon reference trajectory in Figure 5.22 is the predicted state resulting from the control applied in Figure 5.21 applied over each t_{sample}. Moving in and out of stall on several occasions, the attitude of the ducted fan changes significantly over the course of the run, as shown in Figure 5.23. The velocity of the ducted fan is depicted in Figure 5.24. The RHC strategy provides very aggressive and responsive flight qualities. Each RHC trajectory in this run converged to an optimal solution. The computation times for each trajectory are shown in Figure 5.25. The largest computation times occur when the system is far from the commanded equilibrium.

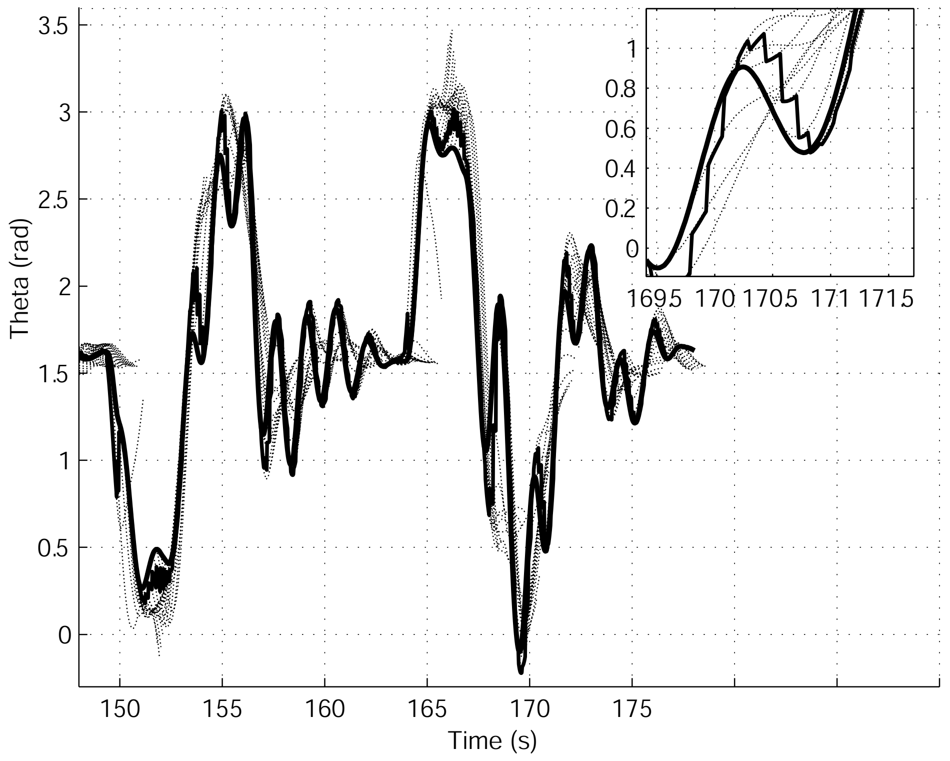

Figure 5.23: The attitude of the ducted fan

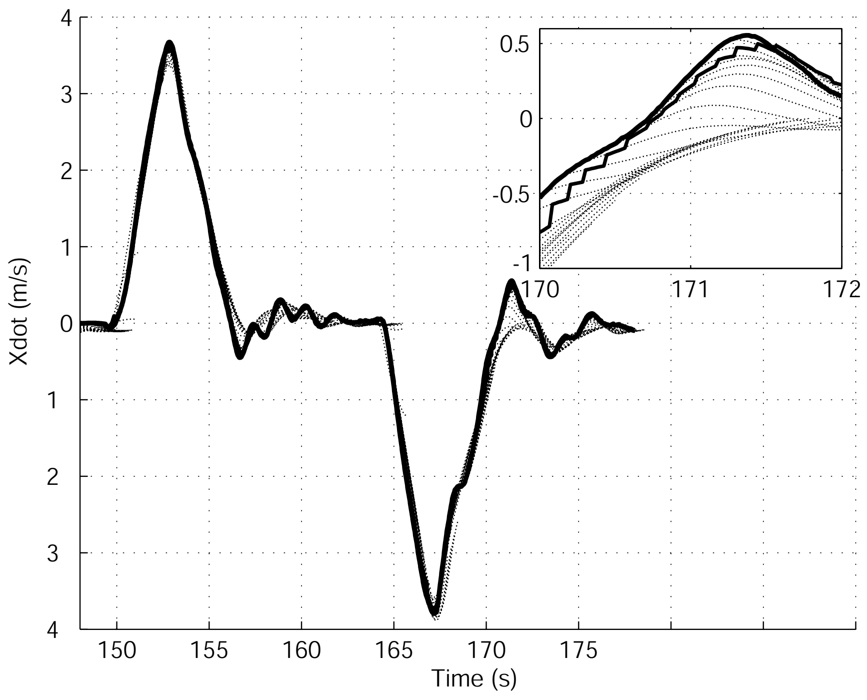

Figure 5.24: The spatial horizontal velocity of the ducted fan

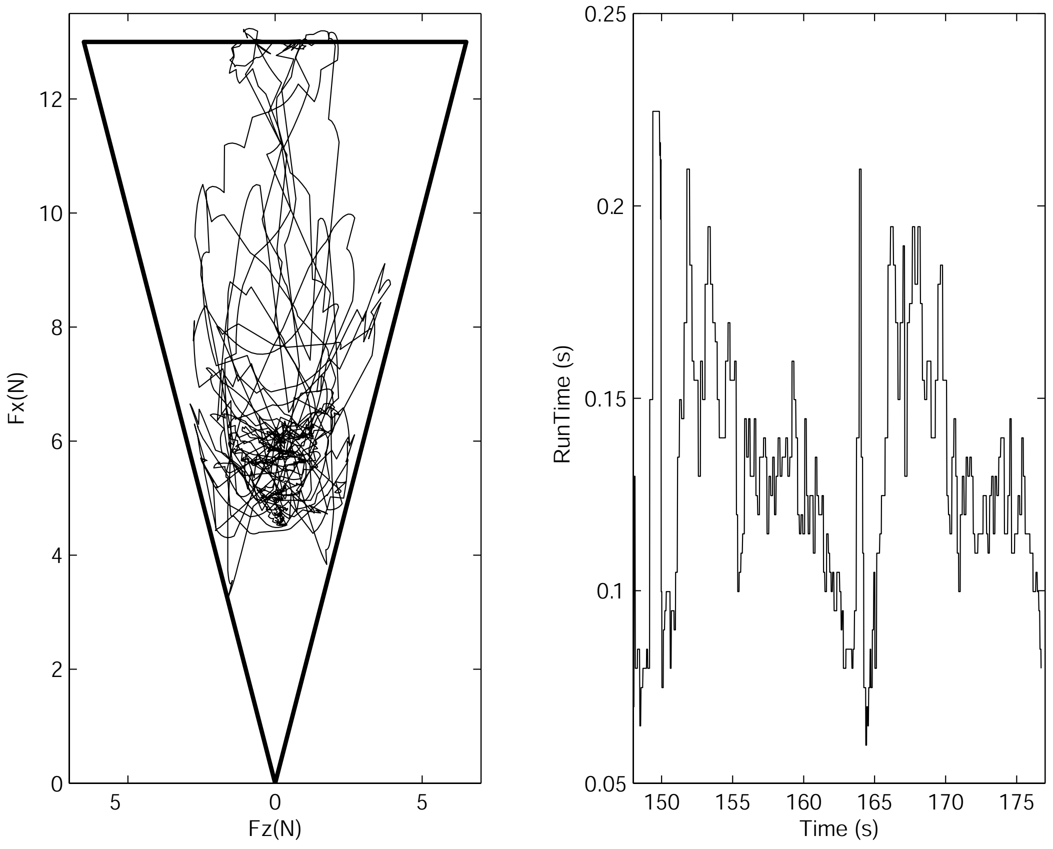

Figure 5.25: Force constraints and computation times.

NTG is started with an initial guess. However, the RHC solution at t_{k+1} uses the solution from t_k as the initial guess. The forces in are acceptably with the prescribed bounds as shown in Figure 5.25. The density of the collocation points determines how well the controls stay within the prescribed bounds.

The second test case is used to illustrate the inherent difficulties with state constraints. Figure 5.26 shows part of a run where only the z equilibrium is position is changed. In this case, the z state constraint is set be less than the maximum vertical travel of the experiment. Due to friction, there are significant model differences between NTG and the real system in the vertical direction. This difference in the model exhibits a weakness in our RHC strategy to model uncertainty. Between the times 115.5 to 117.1 s and 120.4 to 121.7 s NTG does not provide a feasible trajectory since we are starting in a region of infeasibility. Figure 5.26 shows the system recovers but hits the soft state constraint again. Finally, the joystick is moved and the system moves away from the constraint. The are many strategies that one could adopt to mitigate this problem. One could use a barrier function, change the state constraint in real-time so that the RHC problem is feasible (this would only work if the system does not exhibit strong non-minimum phase characteristics), or be conservative on the state constraints in order to mitigate our problem. A hard constraint was created by placing a block of wood below the ducted fan stand counterweight so that the ducted fan could come close but not violate the trajectory constraint. The ducted fan lightly bounced off the hard constraint and remained stable with all convergent trajectories.

5.3.5 Conclusion

The results presented in this section demonstrated the potential of real-time receding horizon control for constrained systems with fast dynamics. Real-time RHC control represents a revolutionary alternative to the traditional linear or nonlinear controller design with many benefits.

First, in most cases, a global system model and objective function are easier to obtain than a traditional linear or nonlinear controller that works globally. For a complex nonlinear system, classical controller design techniques would include inflexible methods such as gain scheduling. In comparison, given an accurate nonlinear model and adequately defined objective function, real-time RHC could provide a global optimal control that is elegant and flexible. For example, RHC can be easily reconfigured by changing the model (a reconfigurable UAV with a swing wing, or payload variation, etc.)

Second, real-time RHC can provide optimal control solution, even for systems with complex constraints such as actuator saturation, operational limits, terrain avoidance, etc. In contrast, it is extremely difficult to design a classic controller for constrained systems.

Third, with accurate modeling and precise objective definition, system perfor-

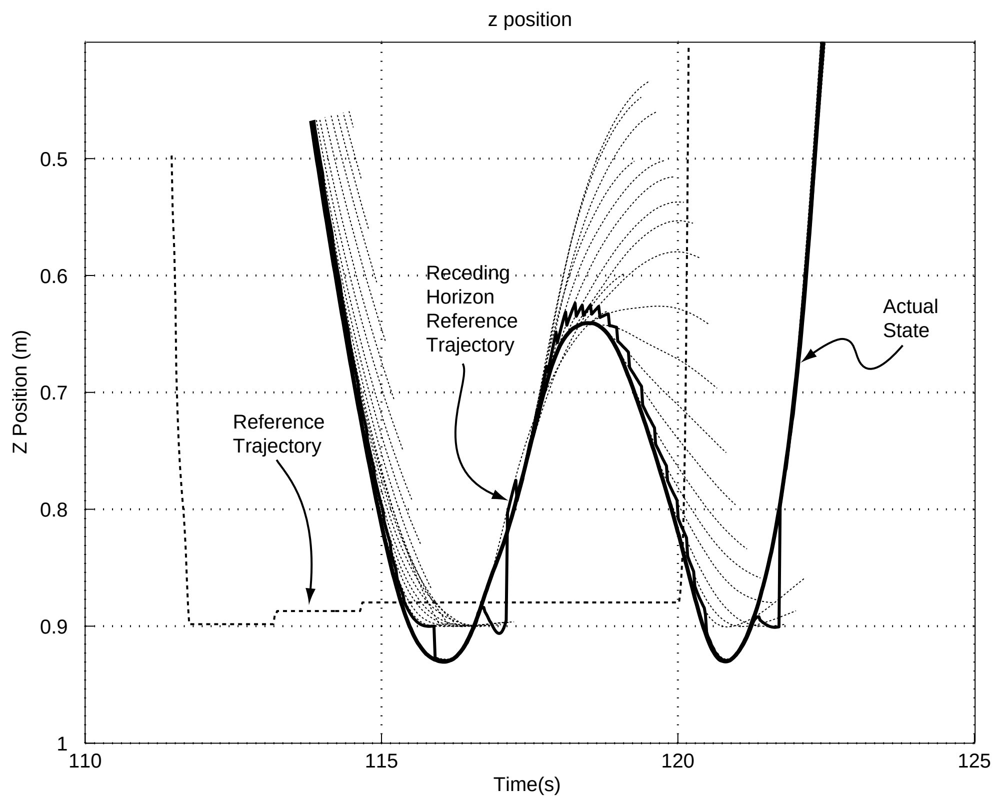

Figure 5.26: Vertical position of the ducted fan

mance could be far more superior than classic linear or nonlinear controller can achieve, particularly for very aggressive maneuvering that pushes the constraint boundaries.

Fourth, in many cases, real-time RHC eliminates the necessity of both inner loops and outer loops that is common in classic tracking and stability control design. Instead, trajectory generation and robust control are performed in a single integrated design with potentially better performance and higher bandwidth.

The theoretical discussion in Section 5.3.1 provided a framework for qualitative evaluation of different timing schemes to compensate for computational delays. The feasibility of two timing methods are verified through simulation, while the non-predictive timing scheme was deployed in a ducted fan experiment.

In thischapter, we investigated implementing both a two-degree-of-freedom design as well a receding horizon design on the Caltech Ducted Fan experiment. The advantages and disadvantages of the two-degree of freedom design were the following.

Advantages:

1. More robust to model uncertainty
2. Works with state constraints
3. Flexibility in objectives

Disadvantages:

1. System reaction latency due to trajectory computation time
2. Linear controller design is complex and may not be compatible with the trajectory generation problem
3. Allowances have to be made for tracking controller

The advantages and disadvantages of the receding horizon design are the following.

Advantages:

1. Quick response to commands

2. Tolerable to reconfiguration

3. Globally stabilizing

Disadvantages:

1. Sensitive to modeling

2. Care must be taken with state constraints

Future research includes extending RHC for nonlinear optimization at the mission level. Merits of different timing methods are to be examined through rigorous mathematical investigation and numerical simulation. One area of active research interests is to keep the current state updated for the optimization routine. We hope our computational experience will guide theoretical developments in the future.

Chapter 6

Micro-satellite Formation Flying

The research in this chapter has been the result of joint research with Nicolas Petit. A preliminary version of this material has appeared in [73].

Nonlinear station-keeping and reorientation control of a cluster of fully actuated, low-thrust, micro-satellites is considered in this chapter. We propose a very general optimization based control methodology to solve constrained trajectory generation problems, which will enable mircosatellites to autonomously perform station-keeping and reorientation maneuvers. Performance is reported for a typical micro-satellite formation flying space mission, using the Nonlinear Trajectory Generation software package.

6.1 Introduction

Several proposed earth orbiting demonstration space missions plan to utilize formations of cooperating, fully-actuated, micro-satellites to perform the function of a single complex satellite. The Air Force space based radar system called TechSat21 [100] is a prime example of such a mission. Burns [14] provides an overview of the TechSat21 mission.

One challenge of these missions is the formation control of the micro-satellites to meet a unified objective. Two typical formation control objectives are the following:

1. Station-keeping: A distributed array of small micro-satellite apertures will collaborate to form a much larger aperture than that possible with a single satellite.

2. Reconfiguration and Deconfiguration: Distribution of micro-satellite aperture can be dynamically reconfigured or deconfigured to meet changes in imaging or mission requirements.

Under the classical gravitational potential assumption, a standard approach to the formation control problem is to linearize the dynamics of the satellites around some reference orbit. In the case that the reference orbit is circular with no perturbation forces, the linearized equations of motion are commonly referred to as the Clohessy-Wiltshire equations [21]. When the correct initial conditions are chosen, the relative positions of the micro-satellites are periodic. By positioning the satellites at different phases along these periodic solutions, a sparse aperture can be created for imaging. Ideally, if satellite positioning is acceptable for imaging, no fuel would be used by taking advantage of the natural dynamics of the vehicles. Yeh *et al.* in [113, 114] provided insight into the control of satellite formations using Hills equations.

However, most micro-satellite missions will be subject to various perturbation forces. The second zonal harmonic of the non spherical Earth (J_2) is a dominant perturbation for the orbits under consideration in this work and cannot be neglected. The J_2 perturbation acts differentially on each satellite and induces secular motion between the micro-satellites in the formation. Sedwick *et al.* [92] derived an analytic expression for exact cancellation of differential J_2 for a micro-satellite formation in a polar, circular orbit. Schwieghart *et al.* [91] extended these results to non-polar orbits. The appropriate choice of initial conditions for a micro-satellite can also mitigate the differential effect of J_2, see Schaub *et al.* [89], Vadali *et al.* [101] and Koon *et al.* [53].

Our approach is based on using optimal control to actively control the sparse aperture of the micro-satellites formation. This has the advantage over existing techniques in that geometric formation constraints can be satisfied for arbitrary orbits. Past work has shown that optimal control has proven relevant to formation flying in the absence of J_2.

Kumar *et al.* [54] used optimal control to solve for the relative motion of two satellites in very low Earth orbits. As a result, Kumar kept the satellites constrained to a box, accounting for a generic differential drag perturbation. Kong [52] provided criterion for optimal trajectories for spacecraft interferometry. Inalhan *et al.* [45] considered the micro-satellite reconfiguration as a distributed and hierarchical control problem. Carpenter [15] also considered a decentralized formation control problem.

In this paper we will explicitly take the J_2 effect into account and considers a centralized optimal control formulation for a micro-satellite formation. The optimal control problem is then solved using the Nonlinear Trajectory Generation (NTG) software package described in Chapter 3.

Two strategies will be considered. First, the station keeping control of three satellites: minimize fuel subject to some nonlinear communication and imaging trajectory constraint. Second, the reconfiguration control of three micro-satellites: minimize fuel subject to final time formation constraints.

We will address the formation control problem in terms of the absolute reference frame. Using the absolute reference frame is a challenge since numerical computations must be done with a high degree of accuracy. Yet, this point of view simplifies the methodology. Optimal trajectories to as a reference trajectory may not be simple periodic trajectories. Instead, the trajectories would be expressed as a time varying curve, creating complicated expressions for the linearized dynamics. Finding the best trajectories for a *formation* of micro-satellites is a difficult task. The numerical implementation of optimal control for such a strategy would also be complex. Another disadvantage of using the linearization is that large reconfiguration maneuvers may be away from the region where a linearization is valid.

This chapter is organized as follows. Section 6.2 presents the formulation of the problem under consideration. Section 6.3 describes the costs and constraints in order to satisfy typical station keeping and reconfiguration requirements.

Numerical results are given in Section 6.4. Several trade studies are conducted and simulation results are also presented in this section. Finally, extensions to general classes of perturbations and conclusions are given in Section 6.5.

6.2 Problem Formulation

The inertial, orbital, and body are the three reference frames that will be used during our analysis. The superscripts I, O and B denote the inertial,orbit, and body frames, respectively. Figure 6.1 depicts the coordinate systems. For the inertial coordinate system, the X^I direction is toward the vernal equinox, the Y^I direction is along the equatorial axis and the Z^I points toward the north pole. Classically, the local coordinate system is chosen so that the X^O axis points up, the Y^O axis is parallel to the velocity vector and the Z^O axis is in the cross range direction. The body frame is fixed to the satellite and assumed to be aligned to the orbital frame. The motion of each fully actuated micro-satellite can be described in absolute coordinates. Including the J_2 perturbation, the dynamics are described by the following differential equations

$$\begin{aligned}
m\ddot{x}_i &= -\frac{\mu x_i}{|r_i|^3}\left(1 - J_2\frac{3}{2}\left(\frac{R_e}{|r_i|}\right)^2\left(5\frac{z_i^2}{|r_i|^2} - 1\right)\right) + u^I_{x_i} \\
m\ddot{y}_i &= -\frac{\mu y_i}{|r_i|^3}\left(1 - J_2\frac{3}{2}\left(\frac{R_e}{|r_i|}\right)^2\left(5\frac{z_i^2}{|r_i|^2} - 1\right)\right) + u^I_{y_i} \\
m\ddot{z}_i &= -\frac{\mu z_i}{|r_i|^3}\left(1 + J_2\frac{3}{2}\left(\frac{R_e}{|r_i|}\right)^2\left(3 - 5\frac{z_i^2}{|r_i|^2}\right)\right) + u^I_{z_i},
\end{aligned} \tag{6.1}$$

where x_i, y_i, and z_i are the coordinates of the absolute position of the ith micro-satellite $i \in \{1, 2, 3\}$ and $|r_i| = \sqrt{x_i^2 + y_i^2 + z_i^2}$. The gravitational constant is denoted by μ and the second zonal harmonic of the non-spherical earth effect by J_2.

The mass of each satellite is denoted by m and is considered constant (100 kg).

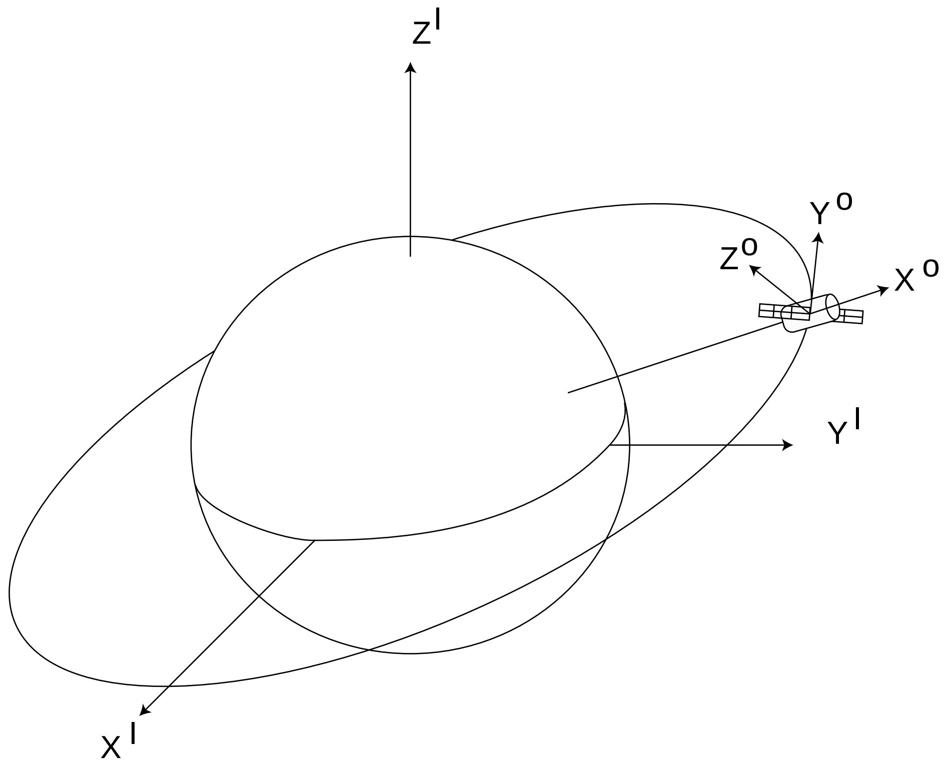

Figure 6.1: Orbit and inertial coordinate systems.

It is assumed that the moments of inertia are such that $(I_Z > I_X, I_Y)$ so that each micro-satellite is gravity gradient stabilized.

It is assumed that the body frame of the satellite is always aligned to the orbit frame as a result of passive attitude stabilization. Therefore, it is easy to find the transformation from the body frame to the inertial frame by the following

$$\begin{aligned} u^I &= T_{IB}u^B = T_{IO}u^B \\ T_{IO} &= \left[\left(\frac{q}{||q||}\right) \times \left(\frac{p \times q}{||p \times q||}\right), \quad \frac{p \times q}{||p \times q||}, \quad -\frac{q}{||q||} \right], \end{aligned}$$

where $u^I = (u^I_{x_i}, u^I_{y_i}, u^I_{z_i})^T$, $p = (\dot{x}, \dot{y}, \dot{z})^T$. T_{IO} is the transformation from the frame fixed to the orbit to the inertial frame and T_{IB} is the transformation from the body frame to the inertial frame. T_{IB} is a rotation matrix, so $||u^B_i|| = ||u^I_i||$ where $||\cdot||$ is the Euclidean norm. This particular point will be useful in the optimal problems formulations.

Note that the orbit rate is given by $\omega = T_{OI}\dot{T}_{IO}$. Since T_{OI} is a function of the inertial velocity, difficulties will arise when linearizing a micro-satellite with J_2 about an arbitrary trajectory in the orbit frame.

6.2.1 Micro-Satellite Formation Flying Requirements

Typical requirements for formation flying are given by Chien *et al.* in [20] and by Esper in [27]. All numerical calculations presented in this paper assume a semi-major axis (a) of 7138 km, which corresponds to an altitude of 800 km for a circular orbit. Eccentricities (e) between between 0 and 0.1 are addressed. In general, the technique presented here can be used for any desired orbit. The requirements on the control actuation and the ΔV are the following:

1. The thrust is considered continuous and is limited to $|30mN|$.
2. The integral of the absolute value of accelerations (ΔV), must not exceed $20m/s/year$.

6.2.2 Micro-Satellite Trajectory Generation

To solve the proposed optimal control problems we will use NTG. First, outputs must be found such that equations (6.1), can be mapped to a lower dimensional output space.

The problem of the outputs for this system is particularly easy to solve since, as with any fully actuated mechanical system, each micro-satellite is differentially flat. Namely, by choosing the configuration variables in (6.1) we can parameterize the complete state and inputs.

Numerical implementation. In NTG, a time scaling is required when working with the system dynamics in equations (6.1) to make the evaluation of the B-spline polynomials accurate. In order to not interfere with the absolute precision of the software package, a time scale was also applied. It turned out that the following scalings worked particularly well for our problem

$$t_s = \frac{t}{S_t}, \qquad q_s = \frac{q}{R_e}, \qquad m_s = \frac{m}{S_m}.$$

The time was scaled by S_t so that an orbit was approximately one time unit. The inertial positions q were scaled by the radius of the earth R_e and the mass m was scaled by S_m to unity.

6.3 Optimal Control Problem Formulation

Parameterizing the trajectory of the micro-satellites over large periods of time would require prohibitively many variables, particularly when using absolute coordinates, rendering real-time computation impossible. Therefore, in order to make the real-time computation tractable, we solve optimal control problems over a finite horizon $[0, T]$. We take T equal to the approximate period of the orbit (without control) and solve the optimal control problem for one orbit. Then we take the ending point of this optimal trajectory as a new starting point and solve the optimal control problem over the horizon $[T, 2T]$, *etc.* This receding horizon methodology, though sub-optimal when compared to the optimal control solution over the whole mission, is numerically tractable and very efficient. Furthermore, it may be necessary to adopt such a strategy, albeit not provably stable, to provide robustness in the presence of unmodeled dynamics and perturbations.

We will consider the three modes of operation as shown in Figure 6.2.

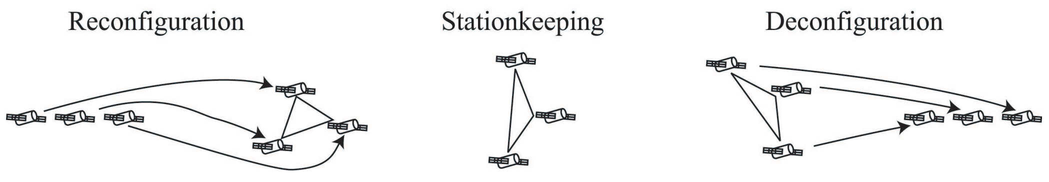

Figure 6.2: Micro-satelllite modes of operation: reconfiguration, station-keeping, and deconfiguration

6.3.1 Station-Keeping with Guaranteed Earth Coverage

The instantaneous fuel consumption of each micro-satellite can be represented as $|u^B_{x_i}|+|u^B_{y_i}|+|u^B_{z_i}|$. This non-differentiable function would make our numerical solver behave poorly. To overcome any trouble with the evaluation of the gradient of the cost, we substitute the quadratic cost $(u^B_{x_i})^2+(u^B_{y_i})^2+(u^B_{z_i})^2$. Though this does affect the formulation of the optimal control, the solution obtained provides a very low $|u^B_{x_i}|+|u^B_{y_i}|+|u^B_{z_i}|$ cost. Moreover, the mapping from u^B to u^I is such that $||u^B||=||u^I||$, which is very convenient for numerical resolution.

Let $T>0$ be the finite horizon over which we want to solve the optimal control problem. The positions of the three micro-satellites will be denoted by $q_1=(x_1,y_1,z_1)$, $q_2=(x_2,y_2,z_2)$, $q_3=(x_3,y_3,z_3)$ and the thrusts by $u^B_1 = T_{OI}(u^I_{x_1},u^I_{y_1},u^I_{z_1})$, $u^B_2=T_{OI}(u^I_{x_2},u^I_{y_2},u^I_{z_2})$, $u^B_3=T_{OI}(u^I_{x_3},u^I_{y_3},u^I_{z_3})$.

We will now cast the requirements for imaging and communications into nonlinear constraint. Determining these constraints are likely to be mission specific. The constraints chosen are purely for illustration to show that complicated, nonlinear constraints can be handled with our methodology.

The first constraint we will consider can be written as

$$\begin{aligned} &||q_i(t)-q_j(t)||\leq d,\\ &\qquad \forall t\in[0,T],\forall(i,j)\in\{1,2,3\}, i\neq j. \end{aligned} \tag{6.2}$$

We will interpret this as a communication constraints, that is, we desire the micro-satellites to stay close together so that communication between the micro-satellites is possible.

The second constraint we will consider is an imaging constraint. We will require that the area projected on the earth be above some threshold. For the sake of simplicity and computational efficiency, we chose not to compute the exact surface of the projection of the triangle defined by the three micro-satellites on the earth. Instead, we computed the projection on the earth as if the earth was locally a plane, which is a reasonable approximation for areas as small as 1000m^2.

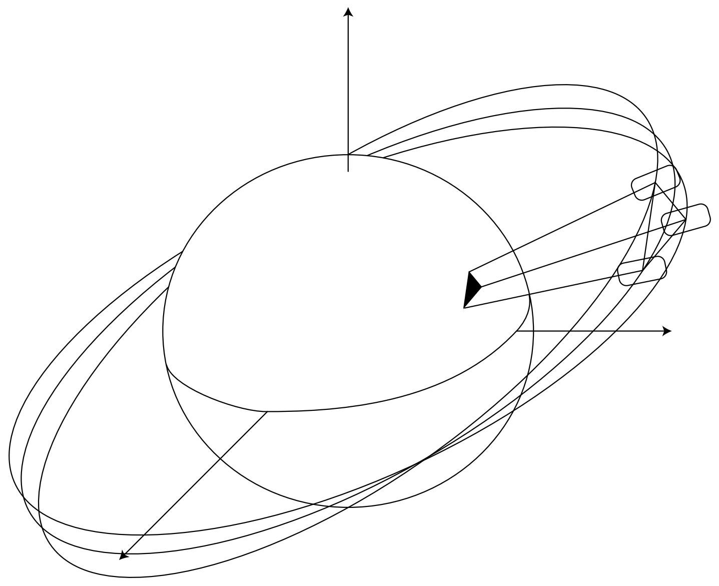

Figure 6.3: Station keeping with guaranteed earth coverage

This "projected" area is, up to an arbitrary choice of orientation,

$$A(t) = \frac{1}{2} n(t) \cdot m(t),$$

where

$$\begin{aligned} m(t) =& (q_1(t) + q_2(t) + q_3(t))/||q_1(t) + q_2(t) + q_3(t)|| \\ n(t) =& (q_1(t) - q_3(t)) \times (q_1(t) - q_2(t)) \end{aligned}$$

The projected area is depicted in Figure 6.3. The imaging constraint is

$$A(t) \geq S, \ \forall t \in [0, T]. \tag{6.3}$$

Finally, we solve the following optimal control problem:

Problem 1 (Nonlinear Station-keeping). Given initial states of the three micro-satellites q_1, q_2, q_3, p_1, p_2, p_3, find a trajectory that minimizes

$$J(u_1^B, u_2^B, u_3^B) = \int_0^T \left(||u_1^B||^2 + ||u_2^B||^2 + ||u_3^B||^2\right) dt \tag{6.4}$$

subject to the dynamics (6.1) and the constraints (6.2), (6.3).

6.3.2 Nonlinear Formation Reconfiguration and Deconfiguration

Given any initial position and velocity of each micro-satellite, we require the three micro-satellites to change their positions and velocities so that the following constraints are satisfied at the final time:

- the relative distances of the three micro-satellites must be less or equal to a prescribed value.
- the projected area on the earth must be no less than a certain value.

Mathematically, these requirements infer the following optimal control problem.

Problem 2 (Reconfiguration). Given initial values for the positions and velocities of the three micro-satellites q_1, q_2, q_3, p_1, p_2, p_3, we look for a minimum of

$$J(u_1^B, u_2^B, u_3^B) = \int_0^T \left(||u_1^B||^2 + ||u_2^B||^2 + ||u_3^B||^2\right) dt \tag{6.5}$$

subject to the dynamics (6.1) and the constraints

$$\begin{aligned} &||q_i(T) - q_j(T)|| \leq d_f \\ &A(T) \geq a_f. \end{aligned}$$

Moreover, we also solve the inverse problem. Starting from a given triangular configuration, we can compute the thrusts required to go to any prescribed positions and velocities. This can be very useful when imaging of the earth is not necessary. We can ask the micro-satellite to go and wait in a “parking” orbit where they do not burn any fuel. For example, if the satellites were to follow one another on the same free orbit, they would not burn fuel or pull apart due to J_2 differential perturbations. When imaging of the earth is necessary, we can reconfigure the micro-satellites into a triangular formation facing the earth and either drift or station keep the formation. Mathematically, these requirements infer the final optimal control problem.

Problem 3 (Deconfiguration). Given initial states of the three micro-satellites q_1, q_2, q_3, p_1, p_2, p_3, we look for a minimum of

$$J(u_1^B, u_2^B, u_3^B) = \int_0^T \left(||u_1^B||^2 + ||u_2^B||^2 + ||u_3^B||^2\right) dt \tag{6.6}$$

subject to the dynamics (6.1).

6.4 Numerical Results

In this section, we provide numerical solutions to the station-keeping, reconfiguration, and deconfiguration problems formulated in in the previous section.

6.4.1 Station-Keeping

The parameterization of the variables of the micro-satellite cluster was achieved by using 10 polynomials of order 9, with 4 regularity conditions at each knot point for each output. This makes a total of 486 coefficients. Seventy collocation points were used to enforce the constraints and evaluate the cost. Orbits without control were used as initial guesses.

The runs were done on a 600 MHz PC. For the station-keeping problem within one orbit, runs take about 120 seconds, while reconfiguration and deconfiguration runs less than one orbit take approximatively 5 seconds to run. The orbits are approximately 6000 sec. The large difference in computation times is due to the fact that we are enforcing difficult nonlinear trajectory constraints for the station-keeping problem and only a nonlinear final time constraint for the reorientation problem.

The initial starting state of each micro-satellites was propagated without control to provide the initial guess. Let S denote the desired projected surface area and d the maximum distance allowed between the micro-satellites.

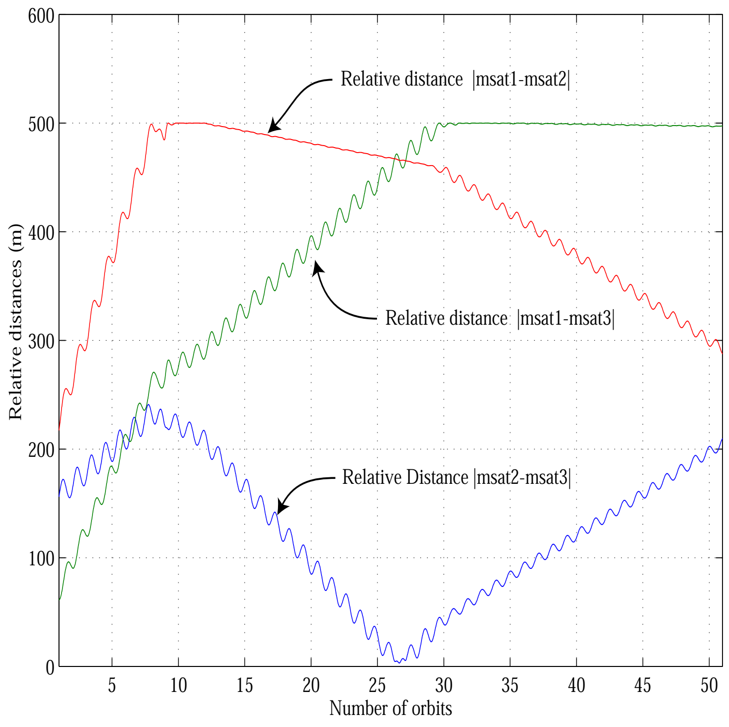

Figure 6.4: Station-keeping for three micro-satellites. Relative distances (m)

Results for the minimum projected surface area $S = 100$ m^2 and the maximum distance between the satellites $d = 500$ m are reported in Figure 6.4 and Figure 6.5. Figure 6.6 shows the ΔV used for the 50 orbits. Figure 6.7 depicts the difference in projected area with and without control.

In this case, the initial conditions were chosen by perturbing nominal values of orbital elements. The hypothesis is that up to first order the formation should not pull apart if the eccentricity and semi-major axis are chosen the same for all micro-satellites. For instance, we chose $a = 7138$ km, $e = 0.1$, $i = 45$ deg, $w = 2$ rad, $\Omega = 0.1$ rad, $M = 0.1$ rad where a is the semi-major axis, e the

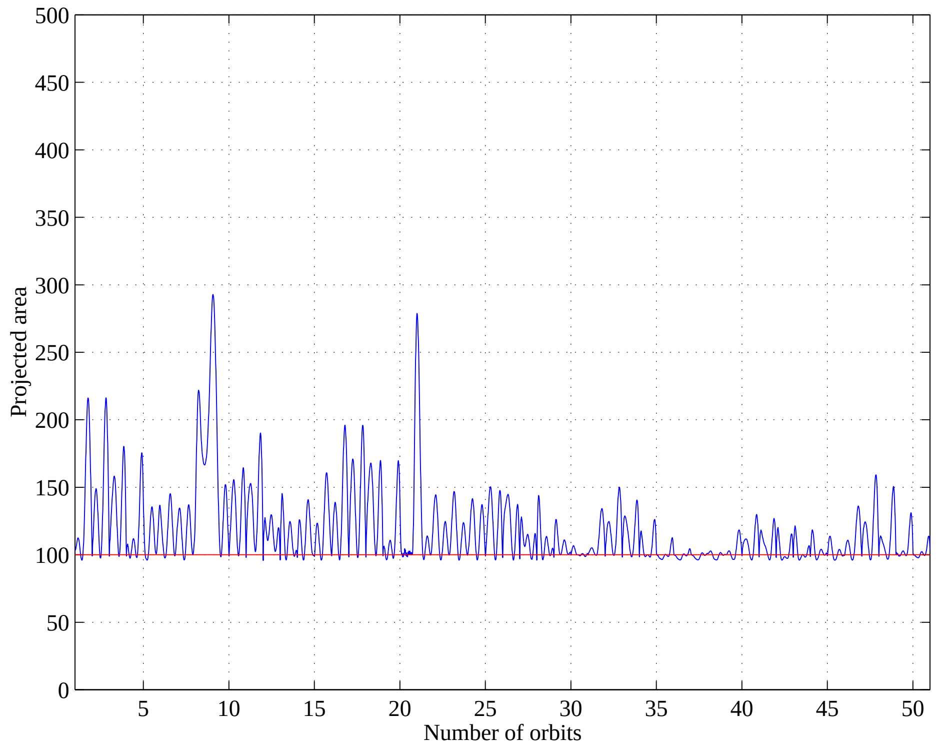

Figure 6.5: Station-keeping for three micro-satellites. Projected area (m^2).

eccentricity, i the inclination, w argument of periapsis, Ω the longitude of the ascending node, and M the mean anomaly, respectively. Then we perturbed this nominal set by $\Delta_1 = (0 \text{ km},\ 0,\ 0 \text{ deg},\ -1e-3 \text{ deg},\ 3.5e-4 \text{ deg},\ 0 \text{ deg})$, $\Delta_2 = (0 \text{ km},\ 0,\ 0 \text{ deg},\ 5e-4 \text{ deg},\ 0 \text{ deg},\ 0 \text{ deg})$ and $\Delta_3 = (0 \text{ km},\ 0,\ 0 \text{ deg},\ -1e-3 \text{ deg},\ -3.5e-4 \text{ deg},\ 0 \text{ deg})$ for satellite 1,2 and 3 respectively. There is no particular reason for choosing these initial conditions, except that they nominally satisfied the station-keeping constraints.

For this trajectory $i = 45$ deg, and the resulting $\Delta V = 10.4$ m/s/year.

Table 6.1 contains results of trade studies with eccentricity, inclination, projected surface area S and the maximum distance between satellites d. Many of these results meet a reasonable requirement of a $\Delta V \leq 20$ m/s/year. The 90 deg inclination seems easier to control. While the J_2 effect is more important than in the other cases, the differential J_2, which really matters, is lower. The controls in

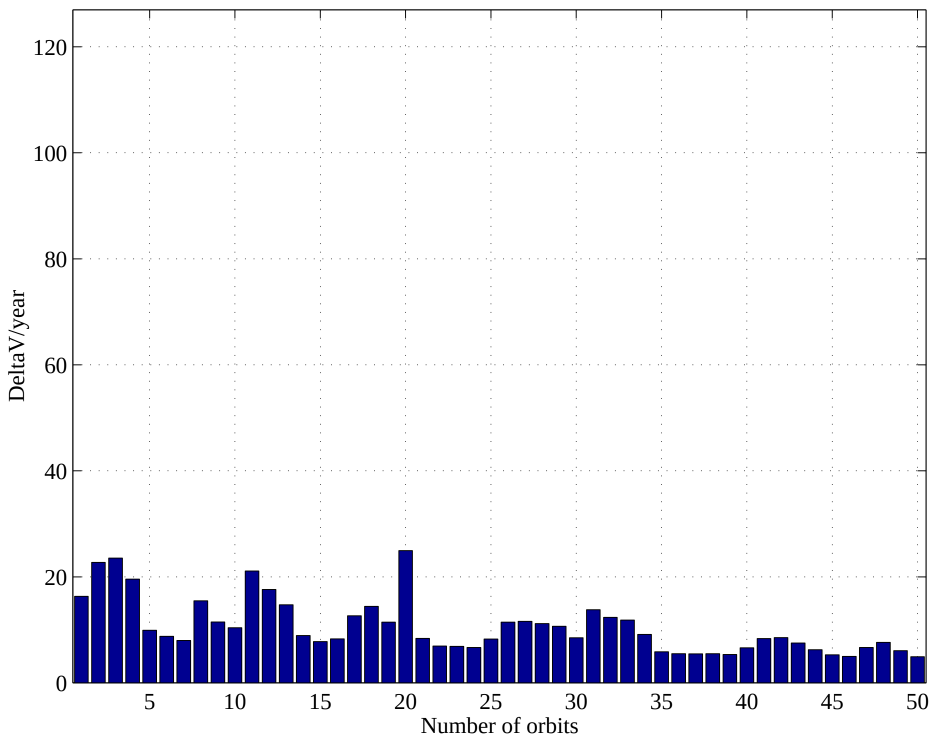

Figure 6.6: Station-keeping for three micro-satellites. ΔV (m/s)

the body frame were within ± 30 mN for all cases under consideration

6.4.2 Results to Problems 2 and 3: Nonlinear Reconfiguration and Deconfiguration

We choose to compute optimal reconfiguration within 2 orbits and studied various cases consistent with the station-keeping cases. A typical micro-satellite reconfiguration maneuver is depicted in Figure 6.8. While the cost of going into a triangular formation decreases with the size of the triangle (constraints are in fact weaker), the cost to come from a triangular formation to a given control-free trajectory increases (the configuration gets harder to recover).

The "parking" strategy seems relevant to useful for mission design. A typical deconfiguration maneuver is depicted in Figure 6.9. It can be seen in Table 6.2 that the ΔV cost of a typical "going out of formation", then "going into formation

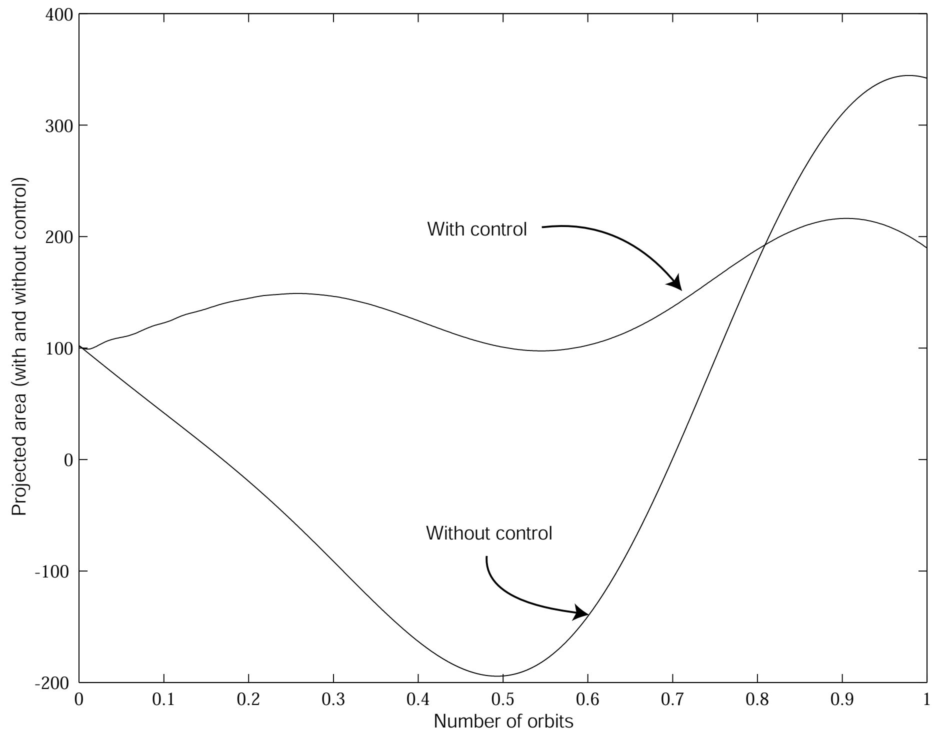

Figure 6.7: Projected area for station-keeping with and without control. Projected area (m^2). Without control, the projected area becomes singular.

again" is about 0.1 m/s.

The reconfiguration maneuver can be used for a variety of different mission requirements. Reconfiguration of a micro-satellite formation to view a specific region of the earth is one possibility. Another possible using of configuration is to move the formation in a configuration such that it can drift while imaging. When the formation drifts apart, the formation can be reconfigured to drift again.

6.5 Conclusion

In this chapter, NTG was employed and successfully solved the important problems of station-keeping, reconfiguration, and deconfiguration for micro-satellite formation flying.

Table 6.1: Station-keeping. Top: effect of S for a given d. Bottom: effect of e for a given S.

$i = 0$ deg	$S = 100$ m^2	$S = 200$ m^2	$S = 300$ m^2
$d \leq 500$ m	$\Delta V = 25.6$ m/s/year	$\Delta V = 47.8$ m/s/year	$\Delta V = 67.3$ m/s/year
$i = 45$ deg	$S = 100$ m^2	$S = 200$ m^2	$S = 300$ m^2
$d \leq 500$ m	$\Delta V = 10.4$ m/s/year	$\Delta V = 17.0$ m/s/year	$\Delta V = 26.8$ m/s/year
$i = 90$ deg	$S = 100$ m^2	$S = 200$ m^2	$S = 300$ m^2
$d \leq 500$ m	$\Delta V = 8.69$ m/s/year	$\Delta V = 21.4$ m/s/year	$\Delta V = 27.4$ m/s/year

$i = 0$ *deg*	$e = 0$	$e = 0.1$
$S \geq 100$ m^2, $d \leq 500$ m	$\Delta V = 25.6$ m/s/year	$\Delta V = 36.7$ m/s/year
$S \geq 100$ m^2, $d \leq 300$ m	$\Delta V = 34.2$ m/s/year	$\Delta V = 33.2$ m/s/year
$i = 45$ deg	$e = 0$	$e = 0.1$
$S \geq 100$ m^2, $d \leq 500$ m	$\Delta V = 10.4$ m/s/year	$\Delta V = 26.0$ m/s/year
$S \geq 100$ m^2, $d \leq 300$ m	$\Delta V = 37.0$ m/s/year	$\Delta V = 34.2$ m/s/year
$i = 90$ deg	$e = 0$	$e = 0.1$
$S \geq 100$ m^2, $d \leq 500$ m	$\Delta V = 8.69$ m/s/year	$\Delta V = 26.1$ m/s/year
$S \geq 100$ m^2, $d \leq 300$ m	$\Delta V = 21.7$ m/s/year	$\Delta V = 28.9$ m/s/year

Many perturbations were not taken into account in this work, such as solar pressure, aerodynamics drag, etc. Depending on the orbit, other perturbations (such as aerodynamic at very low earth orbits) may be dominant. The technique we presented may be generalized to include any perturbation that can be modeled as a function of the positions and their time derivatives, since the model remains *flat*.

The choice of initial conditions seems also a critical issue. Using tools from dynamical systems theory, Koon *et al.* [53] showed that some regions of space offer better initial conditions than others for the station-keeping problem. Starting

Table 6.2: Reconfiguration ΔV for various objectives.

Projected area objective (m^2)	150	200	300	400	500	600
Bound on relative distances (m)	150	200	300	400	500	600
Going in formation ΔV(m/s)	1.49e-1	1.09e-1	7.04e-2	4.25e-2	1.20e-2	9.44e-3
Going out of formation ΔV (m/s)	1.23e-2	1.02e-2	1.32e-2	2.62e-2	3.07e-2	4.89e-2

from these regions of space, we may expect even lower fuel consumptions with the same requirements.

The main result of this work is to report that it is possible to solve problems of engineering interest for micro-satellite formation flying missions by a trajectory generation approach. These trajectories can be computed on board in real-time using NTG.

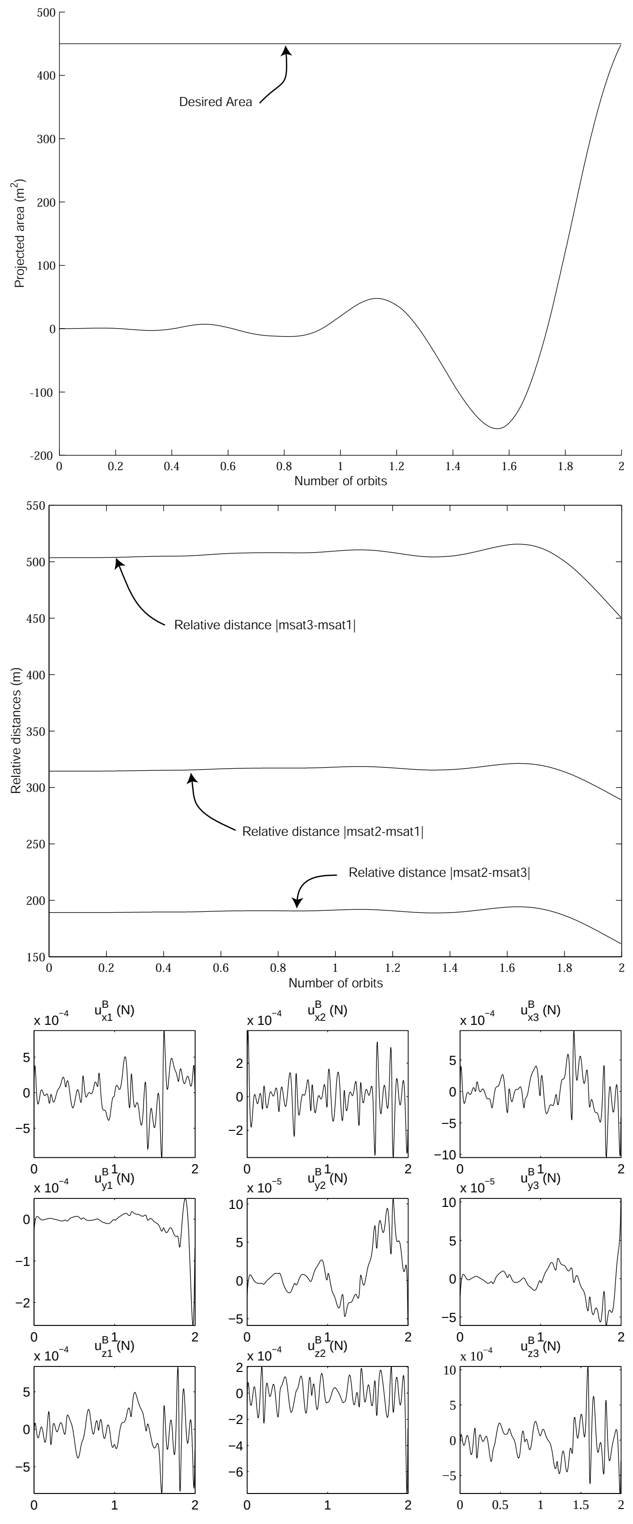

Figure 6.8: Reconfiguration. Going into a formation with a projected area of $450m^2$. Top: projected area versus time. Middle: relative distances versus time. Bottom: thrusts in the body frame, where each colunm is a different satellite.

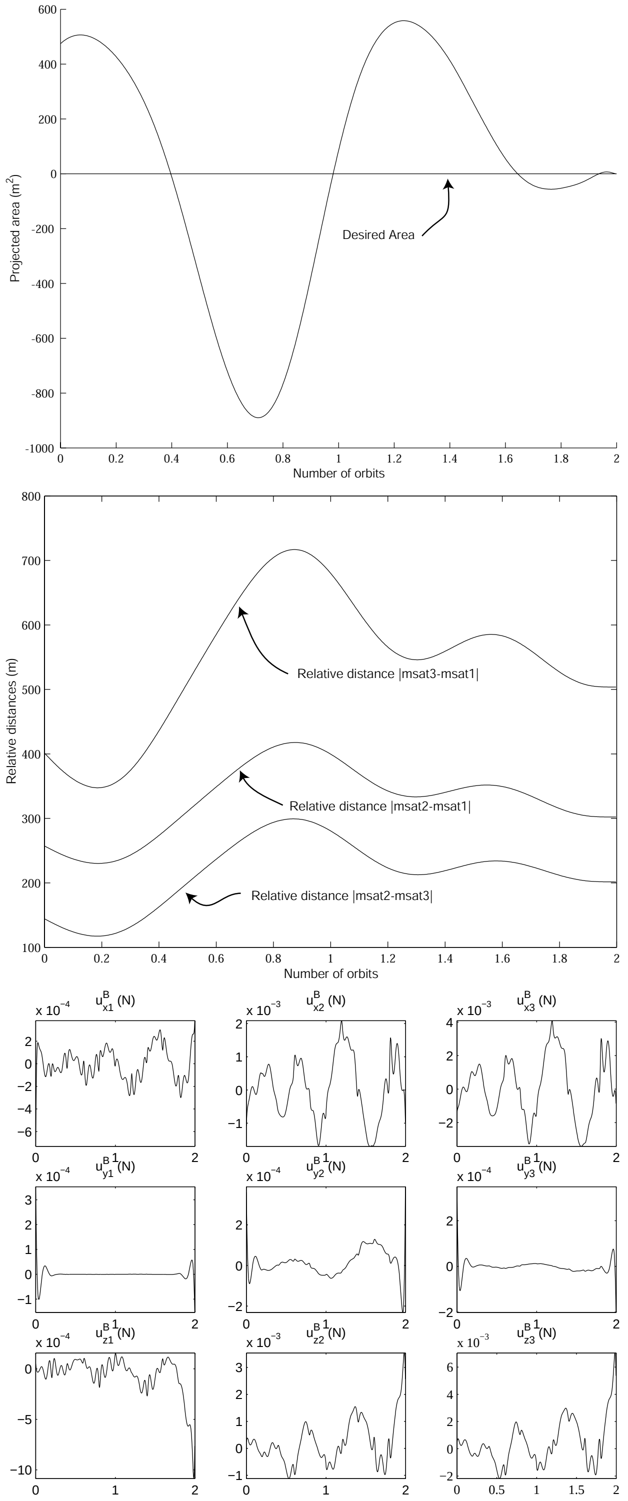

Figure 6.9: Deconfiguration. Going to a "parking" configuration. Top: projected area versus time. Middle: relative distances versus time. Bottom: thrusts in the body frame, where each column is a different satellite.

Chapter 7

Future Work

Differential Flatness The question of necessary and sufficient conditions for differential flatness remains an open research problem. Due to its complexity, we could not determine if the paper by Chetverikov in [19] proves the prized result or casts the problem into an equally difficult one. Finding approximate flat outputs is another area that is important to the improvement of NTG like algorithms.

Differential Flatness and VTOL aircraft Design Designing an aircraft that can hover and be efficient in forward flight is a difficult proposition. It appears that many VTOL aircraft have been designed without accounting for the complexity of the control problem. More research needs to be done to integrate the control system design with the overall design of the aircraft. Simple, differentially flat VTOL aircraft designs, such as the "Aeroranger" may make VTOL aircraft more reliable and efficient than helicopters.

Use of Feasible Nonlinear Programming Solver It was mentioned in Chapter 2, that NPSOL was an infeasible sequential quadratic programming method. In other words, NPSOL does not necessarily satisfy the nonlinear constraints, system or otherwise, until an optimal solution is reached. This is not an ideal situation for critical real-time trajectory generation problems in that a feasible solution is better than none at all. There are several feasible nonlinear programming solvers available. CFSQP is a feasible sequential quadratic programming solver that uses the active set method. The NTG software version 3.1 supports the CFSQP solver and well as NPSOL. A lack of time prevented us from doing significant testing with CFSQP, so it was not included in this thesis. A future research direction would be implement a feasible nonlinear programming solver in NTG and show that one can always obtain a solution, albeit not optimal. In fact, the trust region method KNITRO appears to be the next solver to be employed in the NTG algorithm. Not only does it have a feasible solution option, but it also has the capability to accept analytical second order information concerning the constraints and the objective. In the NTG performance evaluation in Chapter 4, we hypothesized that using second order information would increase the convergence rate for systems that had highly nonlinear constraints.

Combination of NTG and Indirect Methods The combination of NTG and indirect methods such as the trajectory morphing technique presented by Hauser *et al.* in [43] could be used to obtain very accurate solution to optimal control problems. Using the solution of a direct method to initialize an indirect method is advocated in von Stryk *et al.* [109].

Along the same lines, a topic of future research would be to use the information in the trajectory generation solution, such as the Lagrange multipliers, to find a compatible locally stabilizing controller. This is related to the neighboring extremal problem in Bryson and Ho [12]. Currently, we design the locally stabilizing controller off-line with no regard to the cost function in the trajectory generation problem.

Develop a Set Accepted Standards to Compare Optimal Control Transcription Techniques The Constrained and Unconstrained Testing Environment (CUTE) can be used by nonlinear programming solver developers to test their code. A similar environment to CUTE needs to be developed for the optimal control community. Accuracy of solution, computation time, convergence rates are just a few measures that would be used to test optimal control transcription techniques. A library of optimal control problems with a standard interface would also be necessary for the environment.

Generic Optimal Control Problem Solution There may be situation in which it is necessary solve a more general optimal control problem that cannot be cast into the cost of equation 2.2 and and the constraints in equation 2.3. The cost

$$\begin{aligned} J := \int_{t_0}^{t_f} & L(x(\tau - t_1), x(\tau - t_2), \ldots, u(\tau - t_1), u(\tau - t_2), \ldots, \tau) d\tau + \\ & \phi(x(t_1, t_2, \ldots,), u(t_1, t_2, \ldots), t_1, t_2, \ldots) \qquad t_i \in [t_0, t_f] \end{aligned} \tag{7.1}$$

and the constraints

$$lb \leq S(x(t-t_1)x(t-t_2), \ldots, u(t-t_1), u(t-t_2), \ldots, t) \leq ub \; \forall t \in [t_0, t_f] \; t_i \in [t_0, t_f] \tag{7.2}$$

may be applicable a distributed, multi-vehicle environment optimal control problem. For example, the correlation cost function

$$J_1 := \int_{t_0}^{t_f} x(\tau) x(\tau - t_1) d\tau$$

or the trajectory constraint

$$S_1 := \dot{x}^2(t_1) + \sin x(t - t_2) + t_3 = 0 \; \forall t \in [t_0, t_f] \; t_i \in [t_0, t_f]$$

cannot be implemented in the version of NTG used in this thesis.

New Applications of NTG The application of NTG to agile missiles and projectiles is an area of future research. See Milam *et. al* [72] for a solution to a missile problem using NTG. In addition, applying NTG to systems governed by Partial Differential Equations (PDE)'s is an area of future research. Petit *et. al* [80] has started this effort by extending NTG to use tensor product B-spline basis functions.

The field of physical based animation appears to be an area where NTG can contribute to state of the art. Witkin *et al.* [111], Z. Popovic *et al.* [85], J. Popovic *et al.* [84], Wu *et al.* [112], and Liu *et al.* [59] all use either direct collocation or multiple shooting to solve their problems. The animation community could all benefit from having a physical based animation capability built in their design software. Currently, the animation design software MAYA, Lightwave, and 3DStudioMax, have only an inverse kinematics capability. Game consoles could benefit from a program such as NTG built in the software to provide the user more realistic control and visual effects.

Bibliography

[1] *Uninhabited Air Vehicles: Enabling Science for Military Systems.* National Academy Press, 2000.

[2] S. K. Agrawal and N. Faiz. A new efficient method for optimization of a class of nonlinear systems without lagrange multipliers. *Journal of Optimization Theory and Applications*, 97(1):11–28, 1998.

[3] E. L. Allgower and K. Georg. *Numerical Continuation Methods.* Springer-Verlag, 1990.

[4] C. Atkeson. *Using Trajectory Optimizers to Speed Up Global Optimization in Dynamic Programming*, chapter 6. Morgan Kaufmann, 1994.

[5] J. T. Betts. *Practical Methods for Optimal COntrol Using Nonlinear Programming.* SIAM, 2001.

[6] J. T. Betts and J. M. Gablonsky. A comparison of Interior Point and SQP methods on optimal control problems. Technical report, March 2002. Phantom Works, Mathematics and Computing Technology.

[7] J.T. Betts. Survery of numerical methods for trajectory optimization. *AIAA J. Guidance and Control*, 21:193–207, 1998.

[8] R. Bhattacharya and G. J. Balas. Implementation of online control customization within the open control platform. In *Software Enabled Control: Information Technologies for Dynamical Systems.* IEEE Press, 2003. To appear.

[9] C. De Boor. *A Practical Guide to Splines*. Springer, New York, 1978.

[10] A. E. Bryson. *Dynamic optimization*. Addison Wesley, 1999.

[11] A. E. Bryson, W. F. Denham, and S. E. Dreyfus. Optimal programming problems with inequality constraints I: necessary conditions for extremal solutions. *AIAA Journal*, 1(11):2544–2550, November 1963.

[12] A. E. Bryson and Y. C. Ho. *Applied Optimal Control*. Ginn and Company, 1969.

[13] R. Bulirsch, F. Montrone, and H. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2):223–254, August 1991.

[14] R. Burns. TechSat21: Formation design, control, and simulation. In *Proceedings of IEEE Aerospace Conference*, pages 19–25, 2000.

[15] J. R. Carpenter. A preliminary investigation of decentralized control for satellite formations. In *Proceedings of IEEE Aerospace Conference*, pages 63–74, 2000.

[16] B. Charlet, J. Lévine, and R. Marino. On dynamic feedback linearization. *Systems and Control Letters*, 13:143 – 151, 1989.

[17] D. G. Chen and B. Paden. Stable inversion of nonlinear nonminimum-phase systems. *International Journal of Control*, 64(1):81–97, 1996.

[18] Y. Chen and J. Huang. A new computational approach to solving a class of optimal control problems. *International Journal of Control*, 58(6):1361–1383, 1993.

[19] V. N. Chetverikov. New flatness conditions for control systems. In *IFAC Symposium on Nonlinear Control Systems Design (NOLCOS)*, 2001.

[20] S. Chien and R. Sheerwood. The techsat-21 autonomous sciencecraft constellation. In *Proceedings the 6th International Symposium on Artificial Intelligence and Robotics and Automation in Space: i-SAIRAS 2001*, pages 384–388, 2001.

[21] W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 1960.

[22] B. A. Conway and K. M. Larson. Collocation vs differential inclusions. *J. Guidance, Control and Dynamics*, 21(545):780–785, 1998.

[23] J. E. Dennis, A. Booker, P. Frank, D. Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 1:1–13, 1999.

[24] S. Devasia. Approximated stable inversion for nonlinear systems with nonhyperbolic internal dynamics. *IEEE Trans. Automat. Contr.*, 44(7):1419–1425, Feb 1999.

[25] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *IEEE Trans. Automat. Contr.*, 42:930–943, July 1996.

[26] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray. Model predictive control of a thrust-vectored flight control experiment. In *15th IFAC World Congress*, 2002.

[27] J. Esper, S. Neeck, et al. Nano/Micro satellite constellations for Earth and space science. In *Proceedings of the 3rd IAA Symposium on Small Satellites for Earth Observation*, 2001.

[28] F. Fahroo and I. M. Ross. Second look at approximating differential inclusions. *J. Guidance, Control and Dynamics*, 24(1):131–133, 2001.

[29] N. Faiz, S. K. Agrawal, and R. M. Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *J. Guidance, Control and Dynamics*, 24(2):219–227, 2001.

[30] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control, Veldhoven*, 2002.

[31] R. Fletcher. *Practical Methods of Optimization.* 2nd. J. Wiley and Sons, 1987.

[32] M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.

[33] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems. *IEEE Trans. Auto. Cont.*, 44(5):928–937, 1999.

[34] R. Franz and J. Hauser. Optimization based parameter identification of the caltech ducted fan. In *Proc. American Control Conference*, 2003.

[35] R. Franz, M. B. Milam, and J. Hauser. Applied receding horizon control of the caltech ducted fan. In *Proc. American Control Conference*, 2002.

[36] E. Frazzoli, M. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Navigation*, 25(1):116–129, 2002.

[37] P. E. Gill, W. Murray, and M. A. Saunders. *Large-scale SQP Methods and their Application in Trajectory Optimization*, chapter 1. Birkhäuser Verlag, 1994.

[38] P. E. Gill, W. Murray, M. A. Saunders, and M. Wright. *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming.* Systems Optimization Laboratory, Stanford University, Stanford, CA 94305.

[39] P. E. Gill, W. Murray, M. A. Saunders, and M. Wright. *User's Guide to SNOPT 5.3: A Fortran Package for Large-Scale Nonlinear Programming*, December 1998.

[40] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Addison-Wesley, 1981.

[41] C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance and Control*, 10:338–342, 1987.

[42] C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance and Control*, 10:338–342, 1987.

[43] J. Hauser and D. G. Meyer. Trajectory morphing for nonlinear systems. In *American Control Conference*, 1998.

[44] D. Hsu, R. Kindel, J.C. Kindel, and S. Rock. *Randomized Kinodynamic Motion Planning with Moving Obstacles*, pages 247–261. Boston, 2001.

[45] G. Inalhan, J. Busse, and J. How. Precise formation flying control of multiple spacecraft using carrier-phase differential GPS. In *Proc. Guidance, Control and Navigation Conference*, number AAS 00-109, 2000.

[46] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 2nd edition, 1989.

[47] A. Jadbabaie and J. Hauser. On the stability of unconstrained receding horizon control with a general terminal cost. *Systems and Control Letters*. Submitted.

[48] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding-horizon control of nonlinear systems. *IEEE Trans. on Automatic Control*, 46:776–783, 2001.

[49] T. Kaliath. *Linear Systems*. Prentice-Hall, 1980.

[50] H. B. Keller. *Lectures on Numerical Methods in Bifurcation Problems*. Springer-Verlag, 1987.

[51] H. Khalil. *Nonlinear Systems*. Macmillan Publishing Co., New York, NY, 1992.

[52] E. M. C. Kong. Optimal trajectories and optimal design for separated spacecraft interferometry. Master's thesis, Massachusetts Institute of Technology, 1999.

[53] W. S. Koon, J. E. Marsden, J. Masdemont, and R. M. Murray. J_2 dynamics and formation flight. In *Proceedings of AIAA Guidance, Navigation, and Control Conference, Montreal, Quebec, Canada*, 2001.

[54] R. Kumar and H. Seywald. Fuel-optimal stationkeeping via differential inclusions. *Journal of Guidance, Control, and Dynamics*, 18(5):1156–1162, 1995.

[55] S. Lavalle and J. Kuffner. Randomized kinodynamic planning. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, volume 1, pages 473–479, 1999.

[56] C. Lawerence and A. Tits. Nonlinear equality constraints in feasible sequential programming. *Optimization Methods and Software*, 6:265–282, 1996.

[57] C. Lawrance, J. Zhou, and A. Tits. *User's guide for CFSQP Version 2.5.* Institute for Systems Research, University of Maryland, College Park, MD 20742.

[58] F. L. Lewis and V. L. Syrmos. *Optimal Control.* John Wiley and Sons, 1995.

[59] C. K. Liu and Z. Popovic. Synthesis of complex dynamic character motion from simple animations. In *Computer Graphics (Proceedings of the SIGGRAPH 2002)*, pages 209–218, 2002.

[60] R. Mahadevan, S. K. Agrawal, and F. J. Doyle III. Differential flatness based nonlinear predictive control of fed-batch bioreactors. *Control Engineering Practice*, 9:889–899, 2001.

[61] R. Mahadevan and F. J. Doyle III. Efficient optimization approaches to nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 00(2):1–19, 2002.

[62] P. Martin. Aircraft control using flatness. In *European Control Conference Proceedings*, pages 194–199, 1996.

[63] P. Martin. Aircraft control using flatness. In *Multiconference on Computational Engineering in Systems Applications*, pages 194–199, 1996.

[64] P. Martin, S. Devasia, and B. Paden. A different look at output tracking—Control of a VTOL aircraft. *Automatica*, 32(1):101–107, 1994.

[65] P. Martin, S. Devasia, and B. Paden. A different look at output tracking: control of a VTOL aircraft. *Automatica*, 32(1):101–107, 1996.

[66] P. Martin, R. M. Murray, and P. Rouchon. Flat systems, equivalence, and trajectory generation. Apr. 2003. CDS Technical Report.

[67] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814.

[68] G. H. McCall and editors J.A. Corder. *New World Vistas: Air and Space Power for the 21st Century*. United States Air Force, 1996.

[69] M. B. Milam, R. Franz, J. Hauser, and R. M. Murray. On the stability of unconstrained receding horizon control with a general terminal cost. *IEE Proceedings on Control Theory and Applications*, 2003. Submitted.

[70] M. B. Milam, R. Franz, and R. M. Murray. Real-time constrained trajectory generation applied to a flight control experiment. In *15th IFAC World Congress*, 2002.

[71] M. B. Milam and R. M. Murray. A testbed for nonlinear control techniques: The Caltech Ducted Fan. In *1999 IEEE Conference on Control Applications*, 1999.

[72] M. B. Milam and N. Petit. Constrained trajectory generation for a planar missile. Technical Report CIT/CDS 03-011, California Institute of Technology, May 2003.

[73] M. B. Milam, N. Petit, and R. M. Murray. Constrained trajectory generation for microsatellite formation flying. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pages 328–333, 2001.

[74] N. Nijmeijer and A. J. van der Scaft. *Nonlinear Dynamical Control Systems.* Springer Verlag, 1990.

[75] J. Oldenburg and W. Marquardt. Dynamic optimization based on higher order differential model representations. Technical report, October 1999. Technical Report LPT-1999-22.

[76] J. Oldenburg and W. Marquardt. Flatness and higher order differential model representations in dynamic optimization, 2002.

[77] H. Pesch. Real-time computation of feedback controls for constrained optimal control problems. Part 1: Neighboring extremals. *Optimal Control Applications and Methods*, 10:129–145, 1989.

[78] H. Pesch. Real-time computation of feedback controls for constrained optimal control problems. Part 2: A correction method based on neighboring extremals. *Optimal Control Applications and Methods*, 10:147–171, 1989.

[79] N. Petit, M. B. Milam, and R. M. Murray. Inversion based constrained trajectory optimization. In 5*th IFAC symposium on nonlinear control systems*, 2001.

[80] N. Petit, M. B. Milam, and R. M. Murray. A new computational method for optimal control of a class of constrainted systems. In *15th IFAC World Congress*, 2002.

[81] L. Piegl and W. Tiller. *The Nurbs Book.* Springer, New York, 1997.

[82] E. Polak. *Optimization - Algorithms and Consistent Approximations.* Springer-Verlag, 1998.

[83] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes.* Wiley-Interscience, 1962.

[84] J. Popović, S. M. Seitz, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulation. In *Computer Graphics (Proceedings of the SIGGRAPH 2000)*, pages 209–218, 2000.

[85] Z. Popović and A. Witkin. Physically based motion transformation. In *Computer Graphics (Proceedings of the SIGGRAPH 1999)*, pages 11–20, 1999.

[86] W. H. Press, B. P. Flanerry, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* New York:Cambridge University Press, 1986.

[87] M. Rathinam. *Differentially flat nonlinear control systems.* Ph.D. thesis, Cal. Inst. of Tech., 1997.

[88] A. Richards and J. How. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. In *Proc. American Control Conference*, 2003. Submitted.

[89] H. Schaub. Spacecraft formation flying control using mean orbital elements. *Journal of the Astronautical Sciences*, pages 69–87, 2001.

[90] A. Schwartz. *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems.* Ph.D. thesis, U.C. Berkeley, 1996.

[91] S. A Schwieghart and R.J. Sedwick. A perturbative analysis of geopotential disturbances for satellite formation flying. In *Proceedings of the 2001 IEEE Aerospace Conference*, pages 1862–1867, 2001.

[92] Y. Sedwick, D. Miller, and E. Kong. Mitigation of differential perturbations in formation flying satellite clusters. *Journal of Astronautical Sciences*, 47(3 and 4):309–331, 1999.

[93] R. Serban and L. R. Petzold. COOPT-a software package for optimal control of large-scale differential-algebraic equation systems. *Journal of Mathematics and Computers in Simulation*, 56(2):187–203, 2001.

[94] H. Seywald. Trajectory optimization based on differential inclusion. *J. Guidance, Control and Dynamics*, 17(3):480–487, 1994.

[95] H. Seywald and E. M. Cliff. Goddard problem in the presence of a dynamic pressure constraint. *J. Guidance, Control and Dynamics*, 16(4):776–781, 1993.

[96] L. Singh and J. Fuller. Trajectory generation for a uav in urban terrain, using nonlinear mpc. In *Proc. American Control Conference*, 2001.

[97] E. D. Sontag. *Mathematical Control Theory.* Springer, New York, 1998.

[98] M. C. Steinbach. Optimal motion design using inverse dynamics. Technical report, March 1997. Konrad-Zuse-Zentrum für Informationstechnik Berlin.

[99] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer, New York, 1991.

[100] TechSat 21. `http://www.vs.afrl.af.mil/vsd/techsat21/` .

[101] S. Vadali, H. Schauband, and K. Alfriend. Initial conditions and fuel-optimal control for formation flying of satellites. In *AIAA Guidance, Navigation and Control Conference*, August 1999.

[102] A. J. van der Schaft. Representing a nonlinear state space system as a set of higher-order differential equations in the inputs and outputs. *Systems and Control Letters*, 12:151–160, 1989.

[103] M. van Nieuwstadt. *Trajectory generation for nonlinear control systems.* Ph.D. thesis, Cal. Inst. of Tech., 1997.

[104] R. J. Vanderbei. *LOQO User's Manual - Version 4.05*, October 2000.

[105] S. A. Vavasis. *Nonlinear Optimization : Complexity Issues*. Number 8 in International series of monographs on computer science. Oxford University Press, 1991.

[106] T. Veeraklaew and S. K. Agrawal. New computational framework for trajectory optimization of higher-order dynamic systems. *J. Guidance, Control and Dynamics*, 24(2):228–236, 2001.

[107] A. Verma and J. Junkins. Inverse dynamics approach for real-time determination of feasible aircraft reference trajectories. In *Proc. AIAA Guidance, Control, and Navigation Conference*, pages 545–554, 1999.

[108] O. von Stryk. Numerical solution of optimal cntrol problems by direct collocation. *International Series of Numerical Mathematics*, 111:129–143, 1993.

[109] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.

[110] R. A. Waltz and J. Nocedal. *KNITRO User's Manual- Version 3.0*, April 2003.

[111] A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics (Proceedings of the SIGGRAPH 1988)*, pages 159–168, 1988.

[112] J. Wu and Z. Popovic. Realistic modeling of bird flight animations. In *Computer Graphics (Proceedings of the SIGGRAPH 2003)*, 2003. submitted.

[113] H. H. Yeh and A. Sparks. Geometry and control of satellite formations. In *Proceedings of the American Control Conference*, pages 384–388, 2000.

[114] H.H. Yeh, E. Nelson, and A. Sparks. Nonlinear tracking control for satellite formations. In *Proceedings of IEEE Conference on Decision and Control*, pages 328–333, 2000.