INVERSE OPTIMAL CONTROL FOR DETERMINISTIC CONTINUOUS-TIME NONLINEAR SYSTEMS

BY

MILES J. JOHNSON

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Timothy Bretl, Chair
Professor Bruce Conway
Professor Cedric Langbort
Professor Seth Hutchinson

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Abstract

Inverse optimal control is the problem of computing a cost function with respect to which observed state input trajectories are optimal. We present a new method of inverse optimal control based on minimizing the extent to which observed trajectories violate first-order necessary conditions for optimality. We consider continuous-time deterministic optimal control systems with a cost function that is a linear combination of known basis functions. We compare our approach with three prior methods of inverse optimal control. We demonstrate the performance of these methods by performing simulation experiments using a collection of nominal system models. We compare the robustness of these methods by analyzing how they perform under perturbations to the system. We consider two scenarios: one in which we exactly know the set of basis functions in the cost function, and another in which the true cost function contains an unknown perturbation. Results from simulation experiments show that our new method is computationally efficient relative to prior methods, performs similarly to prior approaches under large perturbations to the system, and better learns the true cost function under small perturbations. We then apply our method to three problems of interest in robotics. First, we apply inverse optimal control to learn the physical properties of an elastic rod. Second, we apply inverse optimal control to learn models of human walking paths. These models of human locomotion enable automation of mobile robots moving in a shared space with humans, and enable motion prediction of walking humans given partial trajectory observations. Finally, we apply inverse optimal control to develop a new method of learning from demonstration for quadrotor dynamic maneuvering. We compare and contrast our method with an existing state-of-the-art solution based on minimum-time optimal control, and show that our method can generalize to novel tasks and reject environmental disturbances.

To my family.

Acknowledgements

My work would not have been possible without the support of many people. First I would like to thank my advisor Timothy Bretl for teaching me so much about how to do research, how to be professional, and how to nourish a driving passion for excellence in everything I do. I would also like to thank my committee, all of whom have provided wonderful support through every phase of my time here. I would like to thank Dan Block for his seemingly effortless leadership in the mechatronics lab. It has been my pleasure to work near him and learn from him – I wish I could follow him around all day, every day. My research group, thank you. In particular, thank you Abdullah Akce for being a constant source of calm, patience, and efficiency. Or Dantsker, thank you for being a Motie (you have to read the book) – an inventive and fearless builder of everything, from research aircraft to hobby bicycles. Aadeel Akhtar, thank you for your boundless energy, critical eye, musical talent, and wonderful taste for movies and anime. Navid Aghasadeghi, thank you for always having a fresh, confident, and down-to-Earth perspective on everything. Andy Borum and Joseph DeGol, so many good things are ahead. Thank you Syed Bilal Mehdi for your constantly uplifting attitude in the lab. What fun we had! James Norton, thank you for your constant will to inspire scientific thinking around so many engineers. Bob Sandheinrich, Sophie Puydupin-Jamin, and Olaoluwa Adeniba, thank you for your fresh ideas, sparky attitude, and pepper soup! Aniket Aranake, thank you for being the best intern ever, and a continuing source of brilliant ideas and conversation. Zoe McCarthy and Cyrus Omar, thank you for reminding us how research should be done, with a smile, blazing curiosity, and fearless spirit. Aaron Becker, you have been a constant inspiration to myself and the lab, even after you graduated. Jonathan Ponniah, couldn't ask for a better partner on the wall. Colin Das, once a colleague, now a brother. Dennis Matthews, what can I say!? I hope we continue breaking the odds. Wow, what a group. Finally, I'd like to thank my family. Nothing would be possible without their amazing energy, love, support, and inspiration. Period.

Table of Contents

List of Tables

List of Figures

Chapter 1

Inverse Optimal Control

1.1 Introduction

In the problem of optimal control we are asked to find input and state trajectories that minimize a given cost function. In the problem of inverse optimal control we are asked to find a cost function with respect to which observed input and state trajectories are optimal. Methods of inverse optimal control are beginning to find widespread application in robotics. In this chapter, we consider this problem under deterministic continuous-time nonlinear systems and cost functions modeled by a linear combination of known basis functions. Three existing methods that solve this problem are the following:

- The max-margin inverse reinforcement learning method of Abbeel, et al. [2]. This method is motivated by the problem of efficiently automating vehicle navigation tasks that currently require human expert operation. This method works by trying to learn a cost function that, when minimized, yields a trajectory with similar features as the expert. This method recently contributed to a framework that enables autonomous helicopter aerobatic flight based on observations of human expert pilots.

- The maximum-margin planning method of Ratliff, et al. [3]. This method shares the motivation of Abbeel and Ng, and works by minimizing a regularized risk function using an incremental subgradient method. This method contributed to a framework that mimics human driving of an autonomous mobile robot in complex off-road terrain.

- The method of Mombaur, et al. that we will call bi-level inverse optimal control [1]. This work is motivated by the problem of generating humanoid robot behavior that is similar to natural human motion. This method works by minimizing the sum squared error between predicted and observed trajectories. This method is applied to develop a model of human goal-oriented locomotion in the plane (i.e. paths taken during goal-oriented walking tasks) using observations from motion capture, and implement the model on a humanoid robot.

Despite differences in how learning is performed, these methods exhibit common structure. Each method models the cost function as a linear combination of known basis functions, often referred to as *features*. Each method also contains an inner loop that computes a *predicted trajectory* by minimizing a candidate cost function. In other words, each method solves a forward optimal control problem repeatedly in an inner loop. These methods also yield nominal convergence results. In the work of Abbeel and Ng, after a finite number of iterations, the method returns a cost function with respect to which at least one predicted trajectory performs as well as the observation with known margin. In the work of Ratliff, et al., the method converges linearly to a region around the true cost function, and then only sub-linearly to the true cost function. The method of Mombaur, et al., uses derivative-free optimization methods that generally do not have a complete convergence theory for non-convex objectives. These properties combined with the complexity of solving forward optimal control problems form a computational bottleneck.

We develop an approach that does not solve a forward optimal control problem repeatedly in an inner loop. Our method is inspired by ideas from inverse optimization in [5], making the assumption that observations may arise from a system that is only *approximately optimal*. We define how optimal a trajectory is based on how closely it satisfies necessary conditions for optimal control. This assumption allows us to define residual functions based on these necessary conditions. As a result, the inverse optimal control problem reduces to minimizing these residual functions in order to recover the parameters that govern the cost function. As we will show, this approach reduces to solving a matrix Riccati differential equation followed by one least-squares minimization.

It is unclear at this point how these methods compare in terms of prediction accuracy, computational complexity and robustness to system perturbations. We compare these methods using three example systems: (1) linear quadratic regulation, (2) quadratic regulation of a kinematic unicycle, and (3) characterization of a planar elastica. We compare the robustness of these methods by analyzing how they perform under perturbations to the system. To this purpose, we consider two scenarios: one in which we exactly know the set of basis functions in the cost function, and another in which the true cost function contains an unknown perturbation. Results from simulation experiments show that our new method is more computationally efficient than prior methods, performs similarly to prior approaches under large perturbations to the system, and better learns the true cost function under small perturbations.

The rest of the paper proceeds as follows. In Section 1.2 we formally describe the class of continuous-time, deterministic, nonlinear systems we consider, and the associated inverse optimal control problem. In Section 2.1 we describe the existing methods of inverse optimal control with which we compare our new method [1–3]. In Section 2.2 we develop the new method based on necessary conditions for optimal control. In Section 2.3 we describe the simulation experiments we use to explore the behavior of the methods and their robustness with respect to uncertainty. In Section 2.4 we present experimental results and discussion.

1.2 Inverse Optimal Control: Problem Statement

Consider the following class of optimal control problems

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \int_{t_0}^{t_f} c^T \phi[t, x(t), u(t)] dt \\ \text{subject to} \quad & \dot{x}(t) = f[t, x(t), u(t)] \\ & x(0) = x_{start} \\ & x(t_f) = x_{goal} \end{aligned} \tag{1.1}$$

where $x(t) \in \mathcal{X} \subset \mathbb{R}^n$ is the state, $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ is the input, $\phi : \mathbb{R} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^k_+$ are known basis functions, and $c \in \mathbb{R}^k$ is an unknown parameter vector to be learned. We assume, without loss of generality, that $\|c\| \leq 1$.

We assume that the system equations

$$\dot{x}(t) = f[t, x(t), u(t)] \tag{1.2}$$

are *well posed*, that is, for every initial condition x_{start} and every admissible control $u(t)$, the system $\dot{x}(t) = f[x(t), u(t)]$ has a unique solution x on $t \in [0, t_f]$. This is satisfied, for example, when f is continuous in t and u and differentiable ($\mathcal{C}^1$) in x, f_x is continuous in t and u, and u is piecewise continuous as a function of t [6,7]. The objective basis function ϕ is assumed to be smooth in x and u. This problem also assumes there are no input and state constraints. These constraints are often important in practice, and will be the subject of future work.

The problem of *inverse optimal control* is to infer the unknown parameters with respect to which a given trajectory, the *observation*, is a local minimum to problem (1.1). This observed trajectory is denoted by

$$(x^*, u^*) = \{x^*(t), u^*(t) : t \in [0, t_f]\}\,. \tag{1.3}$$

For convenience, we will often drop the asterisk and refer to an optimal trajectory as (x, t). We also consider observing multiple trajectories, each local minima of problem (1.1) for different boundary conditions. We will refer to a set D of M observations as follows

$$D = \left\{\left(x^{*(i)}, u^{*(i)}\right)\right\} \quad \text{for } i = 1, \ldots, M \tag{1.4}$$

where each trajectory has boundary conditions

$$\left(x^{(i)}_{start}, x^{(i)}_{goal}\right) \quad \text{for } i = 1, \ldots, M. \tag{1.5}$$

An important quantity in the methods discussed in this chapter is the accumulated value of the unweighted basis functions along a trajectory. We will call this the *feature vector* of a trajectory $\mu(x, u)$, defined by

$$\mu(x, u) = \int_{t_0}^{t_f} \phi[t, x(t), u(t)]dt. \tag{1.6}$$

In practice, one would generally have sampled observations of the behavior of the system, but for the analysis in this chapter, we assume we have perfect observations of the continuous-time system trajectories.

We evaluate the solution to the inverse optimal control problem by computing the sum-squared difference between actual and recovered values of the cost function parameters c.

1.3 Related Work

The classical problem of inverse optimal control is to infer the class of objective functions that makes a *given* control policy optimal. This is in contrast to the *data-driven* formulation of the problem in which the control policy is not known and the cost function must be learned from observations of system behavior.

Solution methods for the control-theoretic inverse optimal control problem have been developed for linear systems with quadratic cost along with extensions to nonlinear and stochastic problems [8–28] These methods were first developed in the context of linear time-invariant regulation [8–18]. In particular, this early work focused on determining the class of quadratic cost functions that makes a given linear controller optimal. Extensions of this problem to nonlinear systems are given in [19–22]. Krstic and Tsiotras [21] use inverse optimal control to reconstruct optimal controllers from knowledge of a control Lyapunov function and a particular stabilizing control policy. Li, et al. [22] present an inverse optimal control method for nonlinear systems based on computing an approximate value function given a control policy. Work has also been presented for stochastic nonlinear systems [23, 24]. The H_∞ inverse optimal control problem was presented as the counterpart of the LQ inverse optimal control posed by Kalman [25–27]. An extension poses the problem of computing the plant (system model) whose H_2 or H_∞ controller is equal to a given controller [28].

The data-driven formulation of the classical problem does not assume a given control policy, but instead learns the objective function of a system given *observations* of its behavior. In this context, inverse optimal control is often used as a solution approach to the more general problem of *learning from demonstration*. This problem is often referred to as *imitation learning* or *apprenticeship learning*. The problem of learning from demonstration is to derive a control policy (a mapping from states to actions) from examples, or demonstrations, provided by a teacher. Demonstrations are typically considered to be sequences of state-action pairs recorded during the teacher's demonstration.

There are generally two methods of approach. One approach is to learn a map from states to actions using *classification or regression* [29–43]. Classification techniques range from k-nearest-neighbors to sequencing motion primitives using neural networks. Regression techniques range from those that use lazy learning, where function approximation does not occur until a current query point is given, to full off-line function approximation using neural networks. Also included in this category are works that learn a plan of sequenced logical actions that bring the system from initial to goal state. For further details on these methods, see the survey by Argall, et al. [44].

The second general approach is to learn a cost function with respect to which observed input and state trajectories are (approximately) optimal, i.e. inverse optimal control [1–3, 45–61]. In particular, these methods have focused on finite-dimensional optimization problems, and stochastic optimal control problems.

In the context of finite parameter optimization, Keshavarz, et al. [45] develop an inverse optimization method that learns the value function of a discrete-time stochastic control system given observations. These ideas were extended to learn a cost function for a deterministic discrete-time system in Puydupin-Jamin, et al. [46]. Similarly, Terekhov, et al. [47, 48] and Park, et al. [49] develop an inverse optimization method for deterministic finite-dimensional optimization problems with additive cost functions and linear constraints. Our work takes inspiration from these methods of inverse optimization, applying the concepts of necessary and sufficient conditions for optimality to continuous-time optimal control.

A variety of methods were developed in the context of stochastic optimal control problems, in particular, Markov decision processes [2, 51–60]. Ng and Russell [51] developed a method for stationary Markov decision processes based on linear programming. The method of Abbeel and Ng [2] extends that work by finding a cost function with respect to which the expert's cost is less than those of predicted trajectories by a margin. Later work by Abbeel, et al. [52–54] simultaneously learns the system dynamics along specific trajectories of interest. The method developed in Ramachandran, et al. [56] takes a Bayesian approach and assumes that actions are distributed proportional to the future expected reward. The method developed in Ziebart, et al. [57,58] works by computing a probability distribution over all possible paths that matches features along the observed trajectory. Many distributions satisfy this constraint of matching features, and the principle of maximum entropy is used to resolve this ambiguity. Dvjijotham and Todorov [59] develop a method of inverse optimal control for linearly-solvable stochastic optimal control problems. Their method takes advantage of the fact that, for the class of system model they consider, the Hamilton-Jacobi-Bellman equation gives an explicit formula for the cost function once the value function is known. Aghasadeghi and Bretl [60] develop a method of inverse optimal control that uses path integrals. The use of path integrals leads to a distribution over all possible paths, and the problem is then one of maximizing the likelihood of observations.

Learning from demonstration methods are applied in three different areas. First, learning from demonstration has been applied as a *method of data-driven automation* [2,3,34,40,43,51–55,57,59]. In this case, the purpose is to automate a task currently performed by humans. Tasks of interest include bipedal walking, navigation of aircraft, operation of agricultural and construction vehicles. Second, learning from demonstration methods have been applied to *cognitive and neural modeling* [1,22,38,41,42,47–50,58]. Instead of automation, these applications aim to understand how to quantitatively model a system in a manner that captures or explains system behavior. For example, [58] explain why taxi drives make specific route choices, and [41] predict future trajectories taken by human pedestrians in crowds. Third, learning from demonstration methods have been applied to *system identification* of deformable objects [61]. In this context, the system under consideration is a mechanical structure with static configurations that lie in local energy minima. In Javdani, et al. [61], a method is developed that learns elastic stiffness parameters of objects such as surgical suture, rope, and hair.

1.4 Applications of Inverse Optimal Control

In this dissertation, we compare our new method of inverse optimal control for deterministic nolinear systems with three prior methods of inverse optimal control, and show simulation results on three canonical systems (Chapter 2). We apply our method of inverse optimal control for deterministic nonlinear systems to three problems in robot motion planning: (a) determining the physical parameters of a Kirchoff elastic rod, (b) modeling human goal-oriented locomotion, and (c) learning a feedback controller for dynamic quadrotor flight maneuvers. Chapter 3 presents the problem and solution for the elastic rod. Chapter 4 similarly handles the problem of modeling human locomotion. In Chapter 5 we show how our method of inverse optimal control can be applied to learn feedback control policies for quadrotor dynamic maneuvering. For these applications we show both simulation and hardware experimental results demonstrating the robustness of our approach under various system perturbations such as unknown cost function, system dynamics, and noisy sampled-data observations.

Chapter 2

A Comparison of Inverse Optimal Control Methods

In this chapter we formally describe the three prior methods of inverse optimal control with which we compare the new method developed in Section 2.2. In their original form, the method of Abbeel and Ng, and the method of Ratliff, et al. were developed in the context of Markov decision processes. The general structure and theoretical guarantees of the methods apply with slight modification to the deterministic continuous-time class of problems we consider in this chapter, specified in Equation (1.1).

2.1 Three Prior Methods of Inverse Optimal Control

2.1.1 Method of Mombaur, et al.

In Mombaur, Truong, and Laumond (2010) [1] inverse optimal control is used to generate humanoid robot behavior that is similar to natural human motion. The framework of inverse optimal control is used to understand and identify the underlying optimality criteria of biological motions based on measurements. The solution of this problem yields optimal control models that generate natural humanoid robot motion.

This method works by searching for the cost function parameter c that minimizes the sum-squared error between predicted and observed trajectories. This method has two main components. In the upper-level, a derivative-free optimization technique is used to search for the cost function parameter c. In the lower-level, a numerical optimal control method is used to solve the forward optimal control problem (1.1) for a candidate value of c. We will now discuss the two levels in detail.

The objective of the upper-lever derivative-free optimization is given by the following

$$\underset{c}{\text{minimize}} \int_{t_0}^{t_f} \|[x^c(t); u^c(t)] - [x^*(t); u^*(t)]\|^2 dt \tag{2.1}$$

where $[x^*(t); u^*(t)]$ is the vector concatenation of the state and input of the observed trajectory at time t, and $[x^c(t); u^c(t)]$ is the solution to the forward problem (1.1), given the parameter vector c. The typical starting point for derivative-free methods is the Nelder-Mead simplex algorithm. For example, the `fminsearch` function in Matlab, or the `NMinmimze` function in Mathematica can be used. Higher performance algorithms are discussed in [1] that improve computational running time. For our baseline analysis in this chapter, however, we use the Matlab fminsearch implementation of the Nelder-Mead algorithm. The Nelder-Mead algorithm is a heuristic search method that compares objective function values at vertices of a simplex over the space of parameters c. At each iteration, the worst vertex in the simplex is replaced by a new test point. The new test point is derived by reflecting the worst vertex through the centroid of the simplex. If the value of this new point is a new minimum, then the simplex is expanded in the direction of the reflection. If the value of the new point is a new maximum, the simplex is contracted. Each new test point is a new value of the cost function parameter vector c. These iterations constitute the top-level of the method of Mombaur, et al.

Upon selecting the new point, the lower-level proceeds by solving (1.1) for the current value of c to generate the predicted trajectory (x^c, u^c). Given this trajectory, the objective function can be evaluated, and the method continues. The solution of (1.1) is obtained using a numerical optimal control solver such as direct multiple shooting or collocation. In this chapter, we use the recently developed pseudospectral optimal control package GPOPS [62] to solve the forward problem (1.1). This method terminates when both the objective function (sum-squared error between predicted and observed trajectories) and the variable of optimization (the unknown weight vector c) do not change more than ϵ from one iteration to the next. See Figure 2.1 for a summary of this method.

This method is easily extended for the case where multiple trajectories are observed, where each trajectory has different initial and goal conditions.

```
procedure MethodOfMombaur($x^*, u^*, c_0$)
    $\widehat{c} \leftarrow$ DerivativeFreeOptimization($J, c_0$)
    return $\widehat{c}$
end procedure

function J(c)
    $(x, u) \leftarrow$ SolveForwardProblem($c$)
    return $\int_{t_0}^{t_f} \|[x(t); u(t)] - [x^*(t); u^*(t)]\|^2 dt$
end function
```

Figure 2.1: Overview of the method of Mombaur, et al. [1].

Given M observed trajectories,

$$D = \left\{ \left(x^{(i)}, u^{(i)} \right) \right\} \quad \text{for } i = 1, \ldots, M \tag{2.2}$$

The upper-level objective function becomes

$$\underset{c}{\text{minimize}} \sum_{i=1}^{M} \int_{t_0}^{t_f} \|[x^{c(i)}(t); u^{c(i)}(t)] - [x^{*(i)}(t); u^{*(i)}(t)]\|^2 dt \tag{2.3}$$

where $[x^{c(i)}(t); u^{c(i)}(t)]$ is the solution to the forward problem (1.1) for boundary conditions $(x_{start}^{(i)}, x_{goal}^{(i)})$ and cost function parameterized by the candidate c.

We note that Mombaur, et al. originally specify the upper-level objective as follows

$$\underset{c}{\text{minimize}} \sum_{j=1}^{m} \|[x^c(t_j); u^c(t_j)] - [x^*(t_j); u^*(t_j)]\|^2 \tag{2.4}$$

where t_j are sample times along the trajectory from t_0 to t_f. Given a uniform grid with small sample period, this is equivalent to an approximation of the L^2 norm between the predicted and observed trajectories.

2.1.2 Method of Abbeel and Ng

The method of Abbeel and Ng [2] was originally developed for infinite-horizon Markov decision processes with discounted reward. In this section, we adapt this method to solve the deterministic continuous-time nonlinear inverse optimal control problem defined in Section 1.2. We are given an observation (x^*, u^*) that is assumed to be a local minima of problem (1.1) with cost function parameterized by some $c = c^*$. The goal of this method is to find a control policy that yields a feature vector close to that of the observation. Recall that the observed feature vector is given by

$$\mu(x^*, u^*) = \int_{t_0}^{t_f} \phi[t, x^*(t), u^*(t)]dt \tag{2.5}$$

This is the deterministic analog of the *feature expectations* defined by Abbeel and Ng [2].

The method is initialized by selecting a random cost function parameter vector $c^{(0)}$ and solving the forward problem (1.1) to obtain an initial predicted trajectory $(x^{(0)}, u^{(0)})$ and associated feature vector $\mu^{(0)}$. On the i-th iteration, solve the following quadratic program:

$$\begin{aligned} \underset{c^{(i)}, m^{(i)}}{\text{maximize}} \quad & m^{(i)} \\ \text{subject to} \quad & (c^{(i)})^T \mu^* \leq (c^{(i)})^T \mu^{(j)} - m^{(i)} \\ & \qquad \text{for} \quad j = 0, \ldots, i-1 \\ & \|c\| \leq 1 \end{aligned} \tag{2.6}$$

where $m^{(i)}$ is the margin on the i-th iteration. If $m^{(i)} < \epsilon$, then terminate. Otherwise, given $c^{(i)}$, solve the forward optimal control problem, Equation (1.1), with $c = c^{(i)}$ to obtain the predicted trajectory $(x^{(i)}, u^{(i)})$ and associated feature vector $\mu^{(i)}$. Set $i = i + 1$ and repeat. A summary of this method is shown in Figure 2.2.

Upon termination, this algorithm returns a set of policies Π. In the stochastic system case, one could then form a mixture of these policies to produce a new policy that produces feature expectations closest to those of the observed trajectory. In the deterministic case, this mixing concept still holds, although it has little practice use. Instead, note that upon termination, there exists at least one policy in Π that results in a feature vector that differs from the expert's by no more than ϵ.

We will now adapt the theoretical results from [2] for the deterministic, continuous-time case. First, we will introduce some notation. Given a set of policies Π, let $M(\Pi)$ denote the convex hull of the set of feature vectors

```
procedure MethodOfAbbeel($x^*, u^*$)
    $c_0 \leftarrow$ RandomVector
    $(x^{(0)}, u^{(0)}) \leftarrow$ SolveForwardProb($c_0$)
    $\mu^{(0)} \leftarrow$ FeatureVector($x^{(0)}, u^{(0)}$)
    $i \leftarrow 0$
    repeat
        $m^{(i+1)}, c^{(i+1)} \leftarrow$ QuadProgram (problem (2.6))
        $(x^{(i+1)}, u^{(i+1)}) \leftarrow$ SolveForwardProb($c^{(i)}$)
        $\mu^{(i+1)} \leftarrow$ FeatureVector($x^{(i)}, u^{(i)}$)
        $i \leftarrow i + 1$
    until $m^{(i)} \le \epsilon$
    return $c^{(j)}$    for $j = 1, \ldots, i$
end procedure
```

Figure 2.2: Overview of the method of Abbeel and Ng [2].

attained by the policies in Π,

$$M(\Pi) = Co\left\{\mu(\pi) : \pi \in \Pi\right\}. \tag{2.7}$$

Another important concept that we use is that the feature vectors are bounded, $\phi : \mathbb{R} \times \mathcal{X} \times \mathcal{U} \to [0, \phi_{max}]^k$ for some finite ϕ_{max}. This upper bound may depend on x_{start}, x_{goal}, t_f, and c for a particular problem. For example, this upper bound is easy to compute in the case of linear quadratic regulation, while Lyapunov function analysis can be performed for nonlinear systems. The following Lemma is adapted from Lemma 3 in [2] and establishes improvement in a single iteration of the max margin algorithm.

Lemma 1. *Consider problem 1.1, and a set of policies Π, and the convex hull of feature vectors $M(\Pi)$. Consider the case $\mu^* \in M$. Consider a feature vector $\bar{\mu}^{(i)} \in M$. Let $\pi^{(i+1)}$ be the optimal policy for the cost function defined by $c = (\mu^* - \bar{\mu}^{(i)})$. Define the projection of μ^* onto the line through $\bar{\mu}^{(i)}$ and $\mu^{(i+1)}$, denoted by $\tilde{\mu}^{(i+1)}$, as follows*

$$\tilde{\mu}^{(i+1)} = \frac{(\mu^* - \bar{\mu}^{(i)}) \cdot (\mu^{(i+1)} - \bar{\mu}^{(i)})}{\|\mu^{(i+1)} - \bar{\mu}^{(i)}\|^2}\left(\mu^{(i+1)} - \bar{\mu}^{(i)}\right) + \bar{\mu}^{(i)}. \tag{2.8}$$

Then

$$\frac{\|\mu^* - \tilde{\mu}^{(i+1)}\|}{\|\mu^* - \bar{\mu}^{(i)}\|} \le \frac{k}{\sqrt{k^2 + \|\mu^* - \bar{\mu}^{(i)}\|^2/(\phi_{max}t_f)^2}} \tag{2.9}$$

Proof. For simplicity of notation, let $\bar{\mu}(i) = 0$ (shift coordinates so that it coincides with the origin). The proof develops just as in [2].

$$\frac{(\tilde{\mu}^{(i+1)} - \mu^*) \cdot (\tilde{\mu}^{(i+1)} - \mu^*)}{\mu^* \cdot \mu^*}$$

$$= \frac{\mu^{(i+1)} \cdot \mu^{(i+1)} - \frac{(\mu^{(i+1)} \cdot \mu^*)^2}{\mu^* \cdot \mu^*}}{\mu^{(i+1)} \cdot \mu^{(i+1)}} \tag{2.10}$$

$$\leq \frac{\mu^{(i+1)} \cdot \mu^{(i+1)} - 2\mu^* \cdot \mu^{(i+1)} + \mu^* \cdot \mu^*}{\mu^{(i+1)} \cdot \mu^{(i+1)}} \tag{2.11}$$

$$\leq \frac{(\mu^{(i+1)} - \mu^*) \cdot (\mu^{(i+1)} - \mu^*)}{(\mu^{(i+1)} - \mu^*) \cdot (\mu^{(i+1)} - \mu^*) + \mu^* \cdot \mu^*} \tag{2.12}$$

$$\leq \frac{k^2 (\phi_{max} t_f)^2}{k^2 (\phi_{max} t_f)^2 + \mu^* \cdot \mu^*} \tag{2.13}$$

The preceding steps are described as follows. The definition of $\tilde{\mu}^{(i+1)}$ was used in step (2.10). The fact $(\mu^{(i+1)} \cdot \mu^* - \mu^* \cdot \mu^*)^2 \geq 0$ was used in step (2.11). The fact $\mu^{(i+1)} \cdot \mu^* \geq \mu^* \cdot \mu^*$ was used in step (2.12). Finally, the last step (2.13) used the fact that all of the feature vectors involved lie in M and, further, that all feature vectors are bounded and lie in $[0, \phi_{max} t_f]^k$.

The following Theorem provides a bound on the number of iterations required to achieve a desired margin ϵ. This Theorem is adapted from Theorem 1 in [2].

Theorem 2 (Theorem 1 in [2])**.** *The maximum margin inverse optimal control algorithm will terminate with $w^{(i)} \leq \epsilon$ after at most*

$$n = O\left(\frac{k}{\epsilon^2 / (\phi_{max} t_f)^2} \log \frac{k}{\epsilon / (\phi_{max} t_f)} \right)$$

iterations.

Proof. Given a $\bar{\mu}^{(i)}$, Lemma 1 constructs a point $\tilde{\mu}^{(i+1)} \in M^{(i+1)}$ that is closer to μ^* by a factor given by Eq. (2.9). If $\bar{\mu}^{(i)}$ is such that $\|\mu^* - \bar{\mu}^{(i)}\| \geq \epsilon$, then using Lemma 1 yields

$$\frac{\|\mu^* - \tilde{\mu}^{(i+1)}\|}{\|\mu^* - \bar{\mu}^{(i)}\|} \leq \frac{k}{\sqrt{k^2 + \epsilon^2 / (\phi_{max} t_f)^2}}$$

Now, the algorithm sets $\tilde{\mu}^{(i+1)} = \arg\min_{\mu \in M^{(i+1)}} \|\mu^* - \mu\|$, therefore

$$\frac{t^{(i+1)}}{t^{(i)}} \leq \frac{k}{\sqrt{k^2 + \epsilon^2/(\phi_{max} t_f)^2}}$$

Since the maximum distance between vectors in M is $\sqrt{k}(\phi_{max} t_f)$,

$$t^{(i)} \leq \left(\frac{\sqrt{k}}{\sqrt{k + \epsilon^2/(\phi_{max} t_f)^2}} \right)^i \sqrt{k}(\phi_{max} t_f) \tag{2.14}$$

So $t^{(i)} \leq \epsilon$ if

$$\begin{aligned} i &\geq \log \frac{\sqrt{k}(\phi_{max} t_f)}{\epsilon} \Big/ \log \frac{\sqrt{k + \epsilon^2/(\phi_{max} t_f)^2}}{\sqrt{k}} \\ &= O\left(\frac{k}{\epsilon^2/(\phi_{max} t_f)^2} \log \frac{k(\phi_{max} t_f)}{\epsilon} \right) \end{aligned} \tag{2.15}$$

2.1.3 Method of Ratliff, et al.

The maximum margin planning method of Ratliff, et al. [3] is an inverse optimal control method that tries to learn a cost function for which the expert policy has lower expected cost than every alternative policy by a margin that scales with the *loss* of that policy. This is formalized using ideas from maximum margin structured classification. In this section, we show the development of this method applied to the deterministic continuous-time nonlinear problem defined in Section 1.2.

We are given a set of M observations D

$$D = \left\{ \left(x^{(i)}, u^{(i)} \right) \right\} \quad \text{for } i = 1, \ldots, M \tag{2.16}$$

that are assumed to be local minima of problem (1.1) given corresponding boundary conditions $\left\{ x^{(i)}_{start}, x^{(i)}_{goal} \right\}$ and cost functions parameterized by the same (unknown) $c = c^*$. The notion and effect of the *loss* of a policy is captured by the following set of constraints

$$\forall \pi \in G \quad c^T \mu^{*(i)} \leq c^T \mu(\pi) - L(\pi) \tag{2.17}$$

where G denotes the space of feasible policies, L denotes a *loss function* that defines the closeness of two policies, $\mu^{*(i)}$ is the feature vector of the i-th observation, and

$$\mu(\pi) = \int_{t_0}^{t_f} \phi[t, x^{\pi}(t), u^{\pi}(t)]dt \tag{2.18}$$

is the feature vector of the trajectory that results from executing policy π. Typical loss functions are 0 near observed state-input trajectories, and increase gradually to 1 away from the observed trajectory. These constraints will be satisfied for all $\pi \in G$ if the single constraint holds for the policy that minimizes the right hand side expression. That is, for observation i, all the constraints are satisfied if

$$c^T \mu^{*(i)} \leq \min_{\pi \in G} \left(c^T \mu(\pi) - L(\pi) \right) \tag{2.19}$$

These constraints are nonlinear, but convex in c. This method now finds the smallest weight vector c for which the constraints are satisfied. Since there may not be a parameter vector c that exactly satisfies the constraint, slack variables ζ are introduced that allow constraint violations. These criteria result in the following convex optimization problem

$$\begin{aligned} \underset{c,\zeta}{\text{minimize}} \quad & \frac{1}{M}\sum_{i=1}^{M} \zeta_i + \frac{\lambda}{2}\|c\|^2 \\ \text{subject to} \quad & c^T \mu^{*(i)} \leq \min_{\pi in G} \left\{ c^T \mu(\pi) - L(\pi) \right\} + \zeta_i \\ & \qquad \text{for each } i \end{aligned} \tag{2.20}$$

where $\lambda \geq 0$ is a constant that trades off between the penalizing constraint violations and a desire for small weight vectors. Since the slack variables are in the objective and thus driven to be as small as possible, they will equal the constraint violation at the minimizer.

$$\zeta_i = c^T \mu^{*(i)} - \min_{\pi \in U} \left\{ c^T \mu(\pi) - L(\pi) \right\} \tag{2.21}$$

We can use this to pull the constraint into the objective function to obtain:

$$J(c) = \lambda\|c\|^2 + \frac{1}{M}\sum_{i=1}^{M} \left(c^T \mu^{*(i)} - \min_{\pi \in U} \left\{ c^T \mu(\pi) - L(\pi) \right\} \right) \tag{2.22}$$

Instead of directly solving this convex program, this method utilizes a iterative subgradient technique. The subgradient of a convex function f at a point x is any vector g such that

$$\forall x' \in X \quad f(x') \geq f(x) + g^T(x' - x) \tag{2.23}$$

In general, there are a continuum of subgradients, and at points where f is differentiable, the gradient is the unique subgradient. We want the subgradient of $J(c)$ with respect to c. The only nontrivial term for which we need to compute the subgradient is $-min_{\pi \in U} \left\{ c^T \mu(\pi) - L(\pi) \right\}$. To solve this, note that this term is a convex (but nondifferentiable) function. The subgradient is the gradient of the one function forming the active surface at the value of c. That is, we solve the forward problem at the given value c and obtain the feature vector μ^c associated with the solution. We can now write the subgradient $g(c)$ of $J(c)$ as:

$$g(c) = \frac{1}{N} \sum_{j=1}^{N} \left\{ \mu^{*(j)} - \mu^{(j)} \right\} + \lambda c \tag{2.24}$$

where $\mu^{(j)}$ represents the solution to $\arg\min_{\mu} \left(c^T \mu + L(\mu) \right)$, i.e. the solution to the forward optimal control problem (1.1) for the j-th boundary conditions and with cost function augmented by the loss function. An overview of this method is shown in Figure 2.3.

The theoretical guarantees of this approach are reproduced from [3]

Theorem 3. *Let $\{\alpha_i\}$ be chosen as $\alpha_i = \frac{1}{\lambda}$. Assume that for a particular radius R around the true minimum, $\forall c \ \|g\| \leq C$. Then the algorithm converges at a linear rate to a region around the minimum cost bounded by $\|J - J^*\| \leq \sqrt{\frac{\alpha C^2}{\lambda}} \leq \frac{C}{\lambda}$.*

Proof. By the strong convexity of $J(c)$ and Proposition 2.4 of (Nedic and Bertsekas, 2000)

$$\|c_{i+1} - c^*\|^2 \leq (1 - \alpha\lambda)^{i+1} \|c_0 - c^*\|^2 + \frac{\alpha C^2}{\lambda} \tag{2.25}$$

$$\rightarrow \frac{\alpha C^2}{\lambda} \leq \frac{C^2}{\lambda^2} \quad \text{as } i \rightarrow \infty \tag{2.26}$$

```
procedure MethodOfRatliff($x^{*(i)}, u^{*(i)}, \{\alpha\}, \lambda, N$)
    $j \leftarrow 1$
    $c \leftarrow$ RandomVector
    while $j \leq N$ do
        for $i = 1$ to $M$ do
            $(x^{c(i)}, u^{c(i)}) \leftarrow$ SolveForwardProb($c$)
            $\mu^{c(i)} \leftarrow$ FeatureVector($x^{c(i)}, u^{c(i)}$)
        end for
        $g \leftarrow$ Subgradient($c, x^{c(i)}, u^{c(i)}$)
        $c \leftarrow c - \alpha_j g$
        Project $c$ on to any additional constraints.
    end while
    return $c$
end procedure
```

Figure 2.3: Overview of the maximum margin planning method of Ratliff, et al. [3].

This Theorem specifies that for constant step size α, linear convergence to a neighborhood of the minimum cost is achieved. However, [3] also show that for a diminishing step size $\alpha_j = 1/j$, this method will converge to the minimum, but only at a sub-linear rate. In other words, it is expected that this method will make good improvement in a few iterations, but then can slow down. We also note that this method requires as input a variety of additional parameters that, in general, must be tuned for each problem. In particular, the choice of step size has an important affect on the rate of convergence of the method.

2.2 A New Method Based on Necessary Conditions for Optimality

The three methods described in the previous section exhibit common structure. In particular, each method solves a forward optimal control problem repeatedly in an inner loop. They do this in order to compare the observed trajectory (or feature vectors) with predicted trajectories given a candidate cost function. In this section, we derive another approach inspired by recent work in inverse convex optimization by Keshavarz, et al. [5]. The key idea in our approach is that we assume that the observations are perfect measurements of the system evolution, and that the expert is only *approximately optimal*, where we define what it means to be approximately optimal below. Under this new set of assumptions, we can immediately say how optimal the agent is by looking at how well the demonstration trajectory satisfies the necessary conditions for optimal control. To do this, we use the necessary conditions to define a set of residual functions. The inverse optimal control problem is then solved by minimizing these residual functions over the unknown parameters. In the remainder of this section, we will describe these different stages in detail.

2.2.1 Residual Function Formulation

Consider a trajectory (x, u) of the system given in Equation (1.2). The minimum principle gives us necessary conditions for (x, u) to be a local minimum of Eq. (1.1) [63,64]. The following theorem states these necessary conditions.

Theorem 4 (Free endpoint, fixed final time)**.** *If* (x, u) *is both regular and a local optimum, then there exists a* costate trajectory

$$p : \mathbb{R} \to \mathbb{R}^n$$

such that x *and* p *are a solution of*

$$\begin{aligned} \dot{x}(t) &= \nabla_p H\left[x(t), u(t), p(t)\right] & x(0) &= x_0 \\ \dot{p}(t) &= -\nabla_x H\left[x(t), u(t), p(t)\right] & p(t_f) &= 0 \end{aligned}$$

and the Hamiltonian $H\left[x(t), u(t), p(t)\right]$ *has a local minimum as a function of* $u(t)$ *at* $u(t) = u(t)$ *for* $t \in [t_0, t_f]$*, where*

$$H\left[x(t), u(t), p(t)\right] = c^T \phi\left(t, x(t), u(t)\right) + p(t)^T f\left(t, x(t), u(t)\right)$$

We apply these necessary conditions to our problem (1.1) to obtain

$$\begin{aligned} -\dot{p}(t)^T &= c^T \nabla_x \phi\left[t, x(t), u(t)\right] + p(t)^T \nabla_x f\left[t, x(t), u(t)\right] \\ p(t_f) &= 0 \end{aligned}$$

and

$$0 = c^T \nabla_u \phi\left[t, x(t), u(t)\right] + p(t)^T \nabla_u f\left[t, x(t), u(t)\right]$$

We now consider that we are given an observation (x, u) that may be thought of as the solution to problem (1.1) for some unknown value of c, although this need not hold. We now form residual equations from the necessary conditions. Let

$$z(t) = \begin{bmatrix} c \\ p(t) \end{bmatrix} \in \mathbb{R}^{k+n} \qquad\qquad v(t) = \dot{p}(t) \in \mathbb{R}^n$$

The residual function $r[z(t), v(t)]$ is then defined as

$$\begin{aligned} r[z(t), v(t)] &= \begin{bmatrix} \nabla_x \phi \Big|_{(x,u)}^T & \nabla_x f \Big|_{(x,u)}^T \\ \nabla_u \phi \Big|_{(x,u)}^T & \nabla_u f \Big|_{(x,u)}^T \end{bmatrix} z(t) + \begin{bmatrix} I \\ 0 \end{bmatrix} v(t) \\ &= F(t)z(t) + G(t)v(t) \end{aligned} \tag{2.27}$$

where we have just rearranged the necessary conditions. The notation $(\cdot)\Big|_{(x,u)}$ is shorthand for evaluating the particular function along the trajectory given in the observation, for example

$$\nabla_x \phi \Big|_{(x,u)} \equiv \nabla_x \phi \left[t, x(t), u(t)\right]. \tag{2.28}$$

We say the observation (x, u) is *approximately optimal* for some c and $p(t)$ (i.e. for some $z(t), v(t)$) if $r[z(t), v(t)]$ is close to zero, where we will formalize what it means to be close to zero in Section 2.2.2.

This formulation can also be extended to handle multiple observations. Consider M trajectories that may have different boundary conditions but have the same fixed final time t_f

$$\left\{ \left(x^{(i)}, u^{(i)} \right) \right\} \quad i = 1, \ldots, M \tag{2.29}$$

with each $(x^{(i)}, u^{(i)}) = \left\{ x^{(i)}(t), u^{(i)}(t) : t \in [0, t_f] \right\}$. The vector of unknown parameters $z(t)$ is extended to include the M unknown costates and $v(t)$ is extended to include the time derivatives of the costates

$$z(t) = \begin{bmatrix} c \\ p^{(1)}(t) \\ \vdots \\ p^{(M)}(t) \end{bmatrix} \qquad v(t) = \begin{bmatrix} \dot{p}^{(1)}(t) \\ \vdots \\ \dot{p}^{(M)}(t) \end{bmatrix}$$

Let the following matrices be defined for each trajectory (for $i = 1 \dots M$)

$$\bar{A}^{(i)}(t) = \begin{bmatrix} \nabla_x \phi \Big|^T_{\left(x^{(i)},u^{(i)}\right)} \\ \nabla_u \phi \Big|^T_{\left(x^{(i)},u^{(i)}\right)} \end{bmatrix} \quad \in \mathbb{R}^{(n+m)\times k} \tag{2.30}$$

$$\bar{B}^{(i)}(t) = \begin{bmatrix} \nabla_x f \Big|^T_{\left(x^{(i)},u^{(i)}\right)} \\ \nabla_u f \Big|^T_{\left(x^{(i)},u^{(i)}\right)} \end{bmatrix} \quad \in \mathbb{R}^{(n+m)\times n} \tag{2.31}$$

$$\bar{C}^{(i)}(t) = \begin{bmatrix} I \\ 0 \end{bmatrix} \quad \in \mathbb{R}^{(n+m)\times n} \tag{2.32}$$

The residual function considering all M trajectories can now be written as follows

$$\begin{aligned} r[z(t), v(t)] &= \begin{bmatrix} \bar{A}^1 & \bar{B}^1 & \dots & & \\ \bar{A}^2 & 0 & \bar{B}^2 & \dots & \\ & & \vdots & \ddots & \\ \bar{A}^M & \dots & & & \bar{B}^M \end{bmatrix} z(t) \\ &\quad + \begin{bmatrix} \bar{C}^1 & 0 & \cdots & \\ 0 & \bar{C}^2 & & \\ \vdots & & \ddots & \\ & & & \bar{C}^M \end{bmatrix} v(t) \\ &= F(t)z(t) + G(t)v(t). \end{aligned} \tag{2.33}$$

The particular structure of this residual function will play an important role in Section 2.4.4 where we discuss how the complexity of this approach scales with the number of observed trajectories.

2.2.2 Residual Optimization

We now solve for the unknown parameters $z(t)$ and $v(t)$ by minimizing the residual functions. In other words, we solve the following problem

$$\begin{aligned} \underset{z(t),v(t)}{\text{minimize}} \quad & \int_{t_0}^{t_f} \|r[z(t), v(t)]\|^2 dt \\ \text{subject to} \quad & \dot{z}(t) = \begin{bmatrix} 0 \\ I \end{bmatrix} v(t) \\ & z(0) = z_0 \qquad \text{(unknown)} \end{aligned} \tag{2.34}$$

where

$$\|r[z(t), v(t)]\|^2 = z^T F^T F z + v^T G^T G v + z^T F^T G v \tag{2.35}$$

where the argument t has been dropped for convenience. If $z(0)$ were known, this would be a standard LQR problem (with cross terms)

$$\begin{aligned} \min_{z(t),v(t)} \quad & \int_{t_0}^{t_f} \left\{ z^T Q z + v^T R v + z^T S v \right\} dt \\ \text{subject to} \quad & \dot{z}(t) = Az + Bv \\ & z(0) = z_0 \end{aligned} \tag{2.36}$$

where

$$\begin{aligned} A(t) &= 0 & B(t) &= \begin{bmatrix} 0 \\ I \end{bmatrix} \\ Q(t) &= F(t)^T F(t) & R(t) &= G(t)^T G(t) \\ S(t) &= F(t)^T G(t). \end{aligned}$$

Solving this LQR problem yields the linear control policy and quadratic value function

$$v(t) = K(t)z(t) \qquad\qquad V(z_0) = z_0^T P(0) z_0$$

where

$$K(t) = -(G(t)^T G(t))^{-1} \left(G(t)^T F(t) + B(t)^T P(t) \right)$$

and where $P(t)$ represents the solution to the LQR Riccati equation. We complete our solution for $z(t)$ by solving the following problem

$$\underset{z_0}{\text{minimize}} \quad z_0^T P(0) z_0.$$

Without normalization, this quadratic program is satisfied by the trivial solution $z_0 = 0$. Normalization is performed by using prior knowledge about the problem domain. For example, when the forward optimal control problem has a quadratic cost function, one can often assume that one of the weights is equal to 1. Throughout all of our simulation experiments described below, we employ this method of normalization.

2.3 Simulation Experiments

To evaluate the performance of the three recent inverse optimal control methods described in Section 2.1 and the new method introduced in Section 2.2, we perform numerical simulations in which we observe optimal trajectories of three different systems and learn the objective function for each system. For each system, we collect the optimal trajectories by simulating the system acting under the optimal control policy for particular boundary conditions and fixed terminal time. We collect simulations for 50 random boundary conditions.

2.3.1 Unknown Basis Functions

To evaluate the robustness of the four methods, we perform the following perturbation to the inverse optimal control problem. Up to this point we have considered the true cost function to be perfectly modeled by the weighted combination of known basis functions

$$J(u) = \int_{t_0}^{t_f} c^T \phi[t, x(t), u(t)] dt. \tag{2.37}$$

In our perturbed problem, we perturb the true cost function such that the model given by the weighted combination of basis functions is only an approximation to the true cost function. In particular, we set the true cost function to be

$$J(u) = \int_{t_0}^{t_f} c^T \phi[t, x(t), u(t)] + d^T \rho[x(t), u(t)] dt. \tag{2.38}$$

where $\rho : \mathcal{X} \times \mathcal{U} \to [0,1]^l$ are perturbation basis functions and $d \in \mathbb{R}^l$ are perturbation weights such that $\|d\| < \epsilon$ for some $\epsilon > 0$. In particular, we model a general perturbation with a linear combination of k-th order multivariate Fourier basis functions. The multivariate basis functions are defined as

$$\rho_i[z(t)] = \begin{cases} 1 & \text{if } i = 0 \\ 1 + \cos(2\pi a^i \cdot z) & \text{for odd } i \\ 1 + \sin(2\pi a^i \cdot z) & \text{for even } i \end{cases} \tag{2.39}$$

for $i = 1, \dots l$, where $a^i = [a_1, \dots, a_{n+m}]$, each $a_j \in [0, \dots, l]$. Here z is the concatenation of the state and input vectors at time t, $z(t) = [x(t), u(t)]$. A particular set of basis functions is formed by systematically varying the elements in each a^i. Note that we limit the values that a^i take by assuming only one nonzero element for each i. Note that in the case of the planar elastica, described below, the perturbation basis functions have the same form as the primary cost basis functions. In this case, we perturb the system by simply including higher order terms.

2.3.2 Three Example Systems

The three systems we use are (a) linear quadratic regulation, (b) regulation of a kinematic unicycle, (c) characterizing the planar elastica. We now describe the forward optimal control problem of each of these systems.

Linear Quadratic Regulation

In our first system, we consider a linear system with quadratic cost

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \int_{t_0}^{t_f} x^T Q x + u^T R u \\ \text{subject to} \quad & \dot{x}(t) = A(t)x(t) + B(t)u(t), \\ & x(0) = x_{start} \\ & x(t_f) = \text{Free}, \end{aligned} \tag{2.40}$$

where states are denoted by $x(t) \in \mathbb{R}^n$ and control inputs are denoted by $u(t) \in \mathbb{R}^m$. We simulate 50 instances (trials) of this LQR problem using state dimensions $n = 5$ and $m = 3$, but using different dynamics, initial conditions and cost functions. The dynamic matrices $A(t)$ and $B(t)$ are assumed time-invariant, with elements drawn from a $N(0, 1)$ Gaussian distribution for each trial. The resulting matrices A are scaled such that $|\lambda_{max}(A)| < 1$, and controllability of the systems are verified manually. The initial state of the system x_0 for each trial is drawn from a $N(0, 5)$, and the final time $t_f = 10$ is fixed for all trials. Moreover, for each trial we select cost matrices Q and R, with diagonal elements generated according to the uniform distributions of $U[0, 1]$ and $U[\epsilon, 1]$ respectively, to obtain nonnegative-definite and positive-definite matrices Q and R. Solving the LQR problem (2.40) for these values results in 50 examples of regulating a linear system from a random initial condition to the origin.

Before discussing the full set of simulation results, we will walk through a simple example of applying our method of inverse optimal control to a two-dimensional LQR system. Consider, for example, the following optimal control problem

$$\underset{x,u}{\text{minimize}} \quad \int_{t_0}^{t_f} c_1 x_1(t)^2 + c_2 x_2(t)^2 + c_3 u(t)^2 dt \tag{2.41}$$

$$\text{subject to} \quad \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \tag{2.42}$$

$$x(t_0) = x_{start} \tag{2.43}$$

$$x(t_f) \in \mathbb{R}^n \tag{2.44}$$

The necessary conditions of optimal control for this problem can be written as follows

$$0 = \dot{p}^*(t) + \begin{bmatrix} 2x_1^*(t) & 0 & 0 \\ 0 & 2x_2^*(t) & 0 \end{bmatrix} c + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} p^*(t) \tag{2.45}$$

$$0 = \begin{bmatrix} 0 & 0 & 2u(t) \end{bmatrix} c + \begin{bmatrix} 0 & 1 \end{bmatrix} p^*(t) \tag{2.46}$$

These conditions are approximately satisfied if the following residual function is near zero given an exemplar trajectory (x, u)

$$r\left(z(t), v(t)\right) = \begin{bmatrix} 2x_1(t) & 0 & 0 & 0 & 0 \\ 0 & 2x_2(t) & 0 & 1 & 0 \\ 0 & 0 & 2u(t) & 0 & 1 \end{bmatrix} z(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} v(t) \tag{2.47}$$

Now, Figure 2.4 shows three separate trajectories, shown in red, green, and blue curves. These three trajectories arise from minimizing three respective cost functions that are shown next to the curves in the figure. Note that each trajectory shares the same initial condition, shown by a green circle, but each trajectory terminates at a slightly different terminal point. Figure 2.5 shows the result of applying each method of inverse optimal control to learn the unknown cost function weights c_1, c_2, c_3. Right away, certain properties of each method stand out: the method of Mombaur, et al. exhibits the most iterations (shown in green curves), but achieves very good trajectory prediction upon termination (final predicted trajectory shown in red). The maximum margin planning method (by Ratliff, et al.) is very fast in that it requires very few iterations to get a good answer, however it terminates before refining the solution. Our new method is not iterative, and, since the observed exemplar trajectories were not corrupted by noise, perfectly recovers the unknown cost function.

Quadratic Regulation of the Kinematic Unicycle

As our second test system, we consider quadratic regulation of the kinematic unicycle

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \int_{t_0}^{t_f} x^T Q x + u^T R u dt \qquad (2.48) \\ \text{subject to } \dot{x}(t) &= \begin{bmatrix} \cos x_3(t) \\ \sin x_3(t) \\ u(t) \end{bmatrix}, \\ x(0) &= x_{start} \\ x(t_f) &= \text{free}, \end{aligned}$$

where states are denoted by $x(t) \in \mathbb{R}^3$ (with $x_i(t)$ representing the i-th element of the vector $x(t)$), and control inputs are denoted by $u(t) \in \mathbb{R}$. We

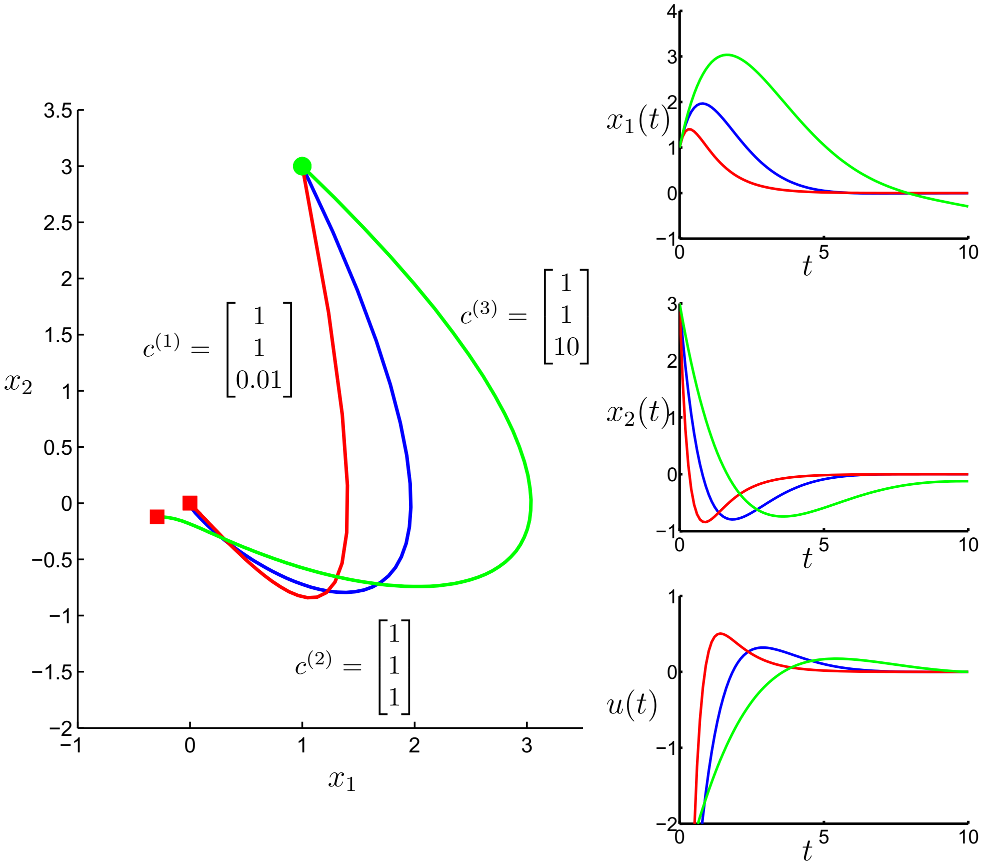
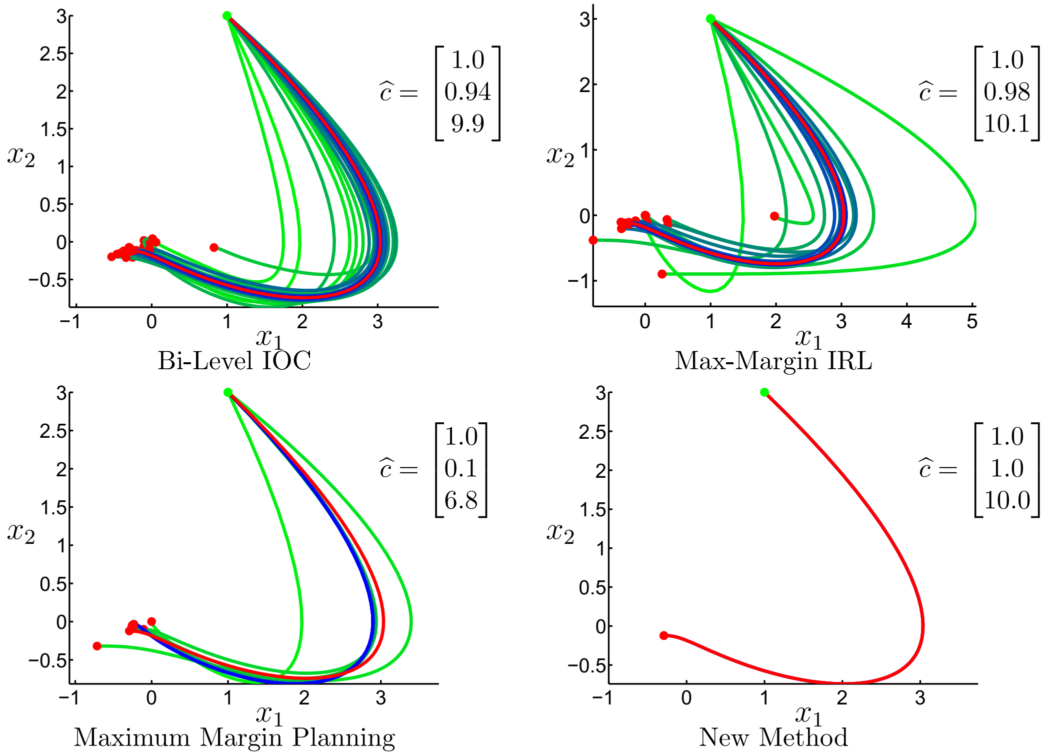


Figure 2.4: LQR example walk through, part 1. This figure shows three separate trajectories, shown in red, green, and blue curves. These three trajectories arise from minimizing three respective cost functions that are shown next to the curves in the figure. Note that each trajectory shares the same initial condition, shown by a green circle, but each trajectory terminates at a slightly different terminal point.



Bi-Level IOC

Max-Margin IRL

Maximum Margin Planning

New Method

Figure 2.5: LQR example walkthrough, part 2. This figure shows the result of applying each method of inverse optimal control to learn the unknown cost function weights c_1, c_2, c_3. See the text for further discussion of the behavior of each method.

simulate 50 instances (trials) of this problem using different initial conditions and different cost functions. The initial state of the system x_0 for each trial is drawn from a $N(0,5)$, and the final time $t_f = 10$ is fixed for all trials. Moreover, for each trial we select cost matrices Q and R, with diagonal elements generated according to the uniform distributions of $U[0,1]$ and $U[\epsilon,1]$ respectively, to obtain a nonnegative-definite and positive-definite matrices Q and R. Solving the optimal control problem (2.48) with these values results in 50 examples of regulating a kinematic unicycle from a random initial condition to the origin.

Characterizing the Planar Elastica

Consider a planar, variable-stiffness, elastica, i.e. a perfectly elastic wire confined to motion in a plane. Such a restriction is realized in practice by considering a wide elastic strip, each end of which is kept perpendicular to the plane of motion. One end is held fixed while a robot manipulator holds the other end. Let $(x_1(t), x_2(t)) \in \mathbb{R}^2$ be the curve traced out by this elastica, as a function of its arc-length $t \in [0,1]$. Let $x_3(t) \in \mathbf{S}^1$ be the tangent angle to this curve. The curvature of the elastica is denoted by $u(t)$. Without loss of generality, we will always assume $x(0) = 0$. In steady-state, for given boundary conditions $x(1) = x_{goal}$, $x(t)$ is a locally optimal solution to the following optimal control problem

$$\begin{aligned} \underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^1 c^T \sigma(t) u(t)^2 dt \\ \text{subject to} \quad & \dot{x}(t) = \begin{bmatrix} \cos x_3(t) \\ \sin x_3(t) \\ u(t) \end{bmatrix} \\ & x(0) = 0 \\ & x(1) = b, \end{aligned} \tag{2.49}$$

where $\sigma(t)$ represents a Fourier basis using a finite collection of terms from the Fourier series as basis functions

$$\sigma(t) = \begin{bmatrix} 1 \\ 1 + sin\left(2\pi t\right) \\ \vdots \\ 1 + sin\left(\frac{k-1}{2} 2\pi t\right) \\ 1 + cos\left(2\pi t\right) \\ \vdots \\ 1 + cos\left(\frac{k-1}{2} 2\pi t\right) \end{bmatrix} \in [0,2]^k. \tag{2.50}$$

In this model, the cost function can be considered the energy of the system – in this case the bending energy of the object. In the experiments we perform, in each trial we randomly select boundary condition x_f, and randomly choose cost function parameters c. We generate random boundary conditions by sampling configurations such that the initial costate of the trajectories are uniformly distributed. We randomly choose c by sampling the uniform distribution $U[0,1]$ for each component, and normalize the parameter vector such that the first element is 1.

Another recent work models deformable one-dimensional objects such as surgical suture, rope, and hair [61]. In this recent work, exhaustive search is used to find the unknown parameters that minimize the squared error between the observed and predicted state trajectories, where the predicted state trajectories are derived by finding local minima of the energy of the object. In other words, this method, when adapted to deterministic continuous-time problems, is analogous to the method of Mombaur, et al. [1].

2.4 Results and Discussion

2.4.1 Perfect Observations with Known Basis Functions

In this set of experiments, each algorithm was given one perfect observation of an optimal trajectory and learned the unknown cost function parameters c. After learning the cost function, predicted trajectories are computed. This allows us to compute other statistics such as the error in total cost, error

Table 2.1: Results for perfect observations with known basis functions.

		Mombaur	Abbeel	Ratliff	New
LQR	computation (s)	280	68	117	4
	forward problems	129	28	48	0
	parameter error	7.03e-2	1.71e-1	6.99e-1	6.35e-8
	feature error	2.30e-3	3.07e-3	1.15e-1	2.81e-9
	trajectory error	1.36e-5	1.04e-4	2.64e-2	1.04e-16
Unicycle	computation (s)	448	63	280	2
	forward problems	133	20	100	0
	parameter error	3.27-2	5.12e-1	5.23e-1	2.54e-5
	feature error	3.53e-3	1.69e-2	1.42e-2	1.03e-5
	trajectory error	1.55e-5	1.12e-3	4.64e-3	8.09e-10
Elastica	computation (s)	428	60	43	3
	forward problems	301	31	4	0
	parameter error	1.78e-1	1.28e+0	1.18e+0	2.96e-7
	feature error	6.28e-3	9.11e-3	2.31e-2	3.44e-3
	trajectory error	6.22e-4	6.55e-4	3.22e-3	3.38e-4

in feature vectors, and sum squared error between observed and predicted trajectories.

Table 2.1 shows results averaged over 50 trials with randomly selected boundary conditions in each trial. These results are consistent with what we expect from the theoretical analysis of each algorithm. In the method of Mombaur, the sum-squared error between predicted and observed trajectories converges near zero as the number of iterations increases. However the inferred cost function parameters are not learned perfectly. Similarly, upon termination of the methods of Abbeel and Ratliff, the error between predicted and observed feature vectors is small, but the cost function parameters are not learned perfectly.

The new method developed in this chapter also performs as expected – learning the unknown parameters perfectly (within the accuracy and precision tolerances of ODE and least squares solvers). In the case of the elastica, the new method learns the unknown cost parameters c to high precision, but shows relatively less precision for the feature vector and trajectory errors. This is due to the numerical forward problem solver being attracted to local minima. In other words, despite having a more precise learned cost function than the other methods, the predicted trajectory is very similar to those predicted by the other methods.

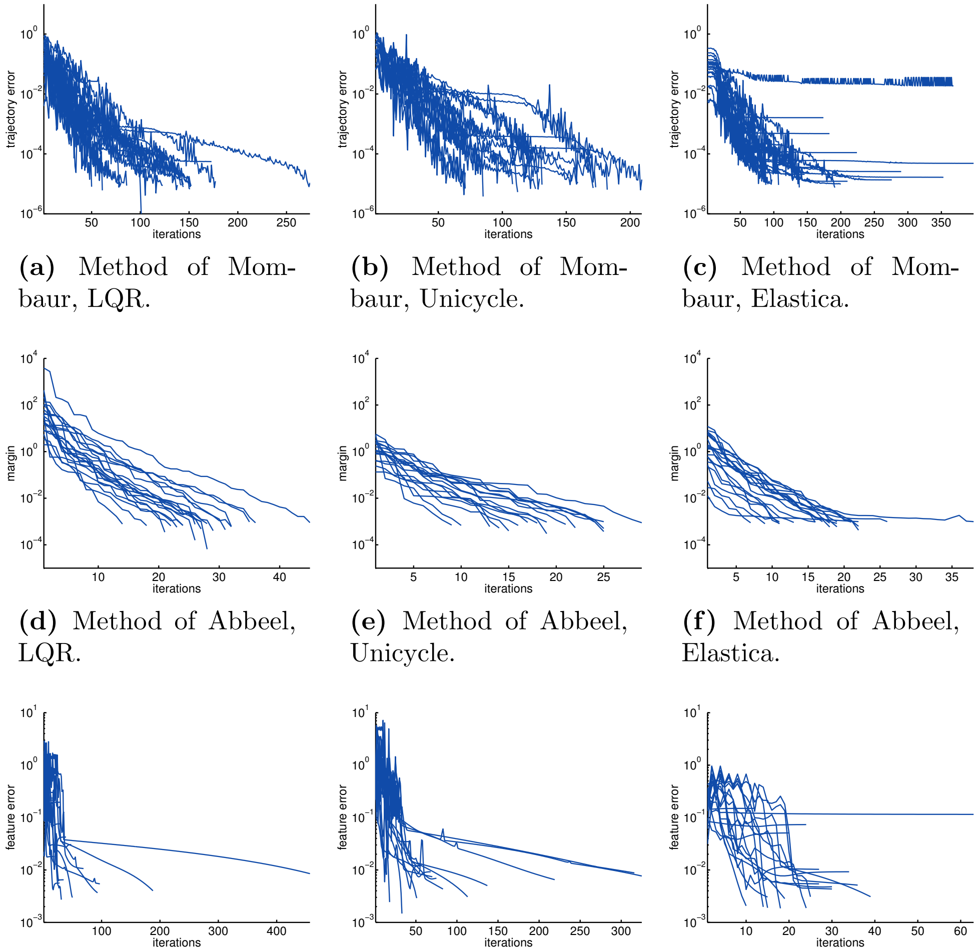
Figure 2.6 shows the convergence of all trials for each of the iterative inverse optimal control methods and each system. This figure also shows that in the method of Mombaur and Ratliff, there are a few trajectories that were problematic for these methods. The occurrence of this issue for the method of Ratliff in each of the three systems is likely due to the fact that the step size sequence in that algorithm is very important in determining the speed of convergence of the algorithm (also see discussion in Section 2.1.3).

2.4.2 Perfect Observations with Perturbed Cost

In this set of experiments, the true cost function consists of a linear combination of known basis functions plus a bounded deterministic perturbation (see Section 2.3.1). For each system, one particular set of boundary conditions was selected, and observations of optimal trajectories are gathered for a range of perturbation magnitudes. Figure 2.7 shows the performance of each method over varying magnitude perturbations. These results generally show:

- All of the methods learn cost functions that are able to approximate the observation in terms of feature vector and trajectory errors.
- The performance of the iterative methods remains close to the results obtained with known basis functions for small perturbations, and then degrades at larger perturbations,
- The performance of our new method (KKT) is linearly proportional to the magnitude of perturbation.

Note that in the case of the elastica, all of the methods, including the new approach based on necessary conditions flattens out at small perturbations. This is due to the forward solver getting stuck in local minima. This is supported by the fact that in the case of perfect observations with completely known basis functions, the new method learns the unknown cost function parameters to a higher precision than is reflected in the feature and trajectory errors (see Section 2.4.1.



(a) Method of Mombaur, LQR.

(b) Method of Mombaur, Unicycle.

(c) Method of Mombaur, Elastica.

(d) Method of Abbeel, LQR.

(e) Method of Abbeel, Unicycle.

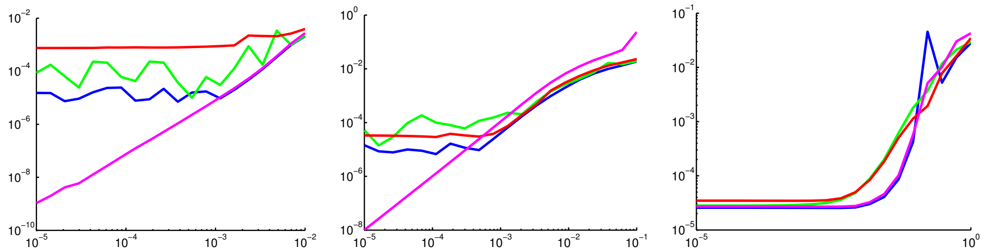
(f) Method of Abbeel, Elastica.

(g) Method of Ratliff, LQR.

(h) Method of Ratliff, Unicycle.

(i) Method of Ratliff, Elastica.

Figure 2.6: This figure shows the behavior of each of the iterative methods. For the method of Mombaur, this figure shows the evolution of the trajectory error for each trial. For the method of Abbeel, this figure shows the evolution of the margin for each trial. For the method of Ratliff, we show the evolution of the feature vector error for each trial.



(a) LQR system, Trajectory error.

(b) Unicycle system, Trajectory error.

(c) Elastica system, Trajectory error.

Figure 2.7: This figure shows how the feature vector and trajectory errors change for varying magnitude perturbations. Blue: Mombaur, Green: Abbeel, Red: Ratliff, Magenta: new method.

2.4.3 Inaccurate Model and Sampled Observations

In our baseline comparison of our new method of inverse optimal control with three existing methods, we considered perturbations in the structure of the cost function. In other words, under this perturbation the known cost basis functions are an approximation of the true cost, and no parameter vector c can perfectly reproduce observed behavior. Thus, a cost perturbation is a direct way to begin understanding how robust these methods are to cost function inaccuracy. We will now consider two other types of system perturbation. First, we will consider inaccurate system dynamics models. This is important in practice when an approximate model is all that is available. Second, we will consider the case when the observed trajectories consist of noisy sampled measurements. The analysis of these types of perturbation will help us understand how robust inverse optimal control methods are to deterministic structural uncertainties and additive stochastic noise.

Inaccurate Dyamics

In practice, the system being studied is not perfectly known or is modeled using simplified or approximate equations of motion. In this case, our new method of inverse optimal control will clearly be affected because it depends on explicit partial derivatives of the dynamics with respect to the state and control vectors. To study the behavior of our method under this type of perturbation, we will consider a set of simulation experiments in which the true dynamics are not available to the IOC methods. In particular we will consider the following perturbations for each of the three example systems in the baseline comparison.

1. **Linear Quadratic Regulation:** In this system, the nominal dynamics are given by $\dot{x} = Ax + Bu$ where $A, B \sim N(0,1)$. We will model the unknown true dynamics by $\bar{A} = A + A_p$ where $A_p \sim N(0,1)$ and is scaled such that $|\lambda_{max}(A_p)| < \epsilon$ for a range of small $\epsilon > 0$.

2. **Kinematic Unicycle:** The nominal dynamics for this system are given by

$$\dot{x} = \begin{bmatrix} \cos x_3 \\ \sin x_3 \\ u \end{bmatrix}$$

We will model the unknown true dynamics by

$$\dot{x} = \begin{bmatrix} s\cos x_3 \\ s\sin x_3 \\ u \end{bmatrix}$$

where $s = 1 + \epsilon$, and $\epsilon \sim U(0, \delta)$, for a range of small $\delta > 0$.

3. **Planar Elastic Rod:** The nominal system was defined as a unit length rod. We will model the unknown true system with an additive length perturbation of $\epsilon \sim U(0, \delta$ for a range of small $\delta > 0$.

Figure 2.8 shows how the performance of each inverse optimal control algorithm change under perturbations of varying magnitude.

Noisy Sampled-Data Observations

In practice, one will often only have access to noisy observations of the system. In this case, one way to improve the accuracy of the learned cost function is to average over multiple observations. This is handled naturally by the multiple-observation formulations for the methods of Mombaur, Ratliff, and the new method developed in this paper. The inverse optimal control problem we consider requires a continuous and differentiable trajectory. Thus, one way to handle noisy observations is to consider sampled observations of the optimal trajectory that are perturbed by zero-mean Gaussian noise, and then interpolated using smooth cubic spline interpolation. The resulting observations will in general not be local extremals, but will be approximations.

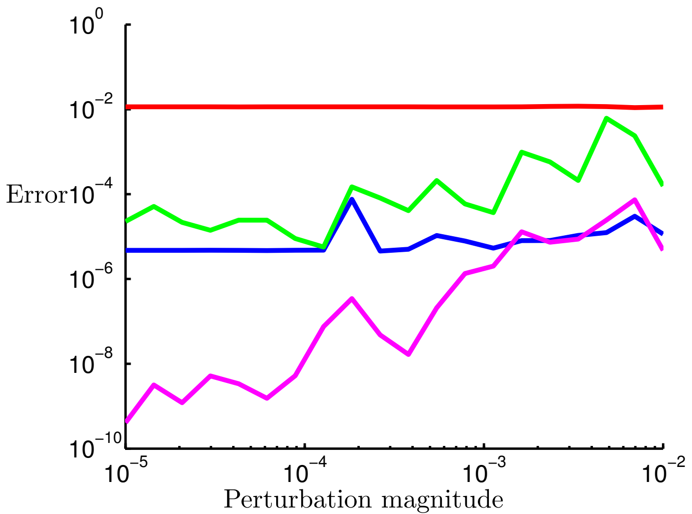
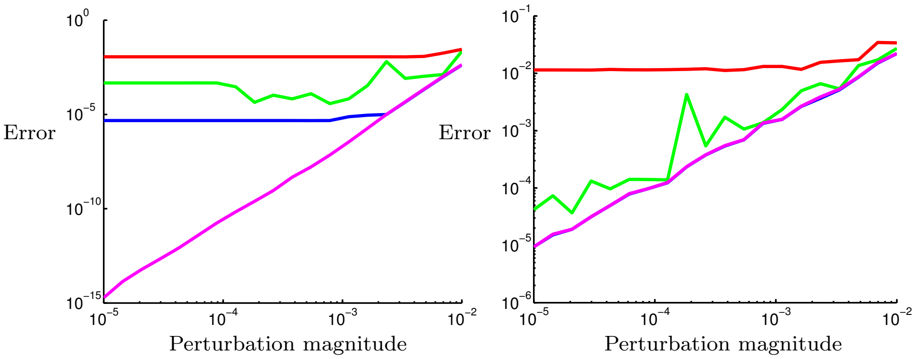


Figure 2.8: Comparison of methods under inaccurate system dynamics perturbation for the Example 1 system. This figure shows the average error in trajectory prediction for varying magnitude perturbation of the underlying system dynamics.

In particular, we perform a set of simulation experiments in which we construct observations as follows. First, we begin with a continuous time local extremal trajectory (x^*, u^*). We then sample this trajectory with a fixed sample period h, yielding a collection of N state and control samples $\{(x_0, u_0), \ldots, (x_N, u_N)\}$ where each $(x_i^*, u_i^*) = (x(t_i)^*, u(t_i)^*)$ for $i = 1, \ldots, N$. Next, we add Gaussian noise to each sampled state and control to yield a sampled noisy observation $x_i = x_i^* + \eta_{x,i}$ where $\eta_{x,i} \sim N(0, \Sigma_x)$, and $u_i = u_i^* + \eta_{u,i}$ where $\eta_{u,i} \sim N(0, \Sigma_u)$, for diagonal and positive $\Sigma_x \in \mathbb{R}^n, \Sigma_u \in \mathbb{R}^m$. Figure 2.9(a) shows how the performance of each inverse optimal control algorithm changes for varying sample period and Figure 2.9(b) shows how performance changes for varying magnitudes of zero-mean additive Gaussian noise.

2.4.4 Complexity of the Approaches

An important property of the approaches is how they scale with the number of observations. In the methods of Mombaur and Ratliff, the complexity is roughly linear in the number of observations (recall that Abbeel's method was not posed for multiple observations). In other words, in each iteration, instead of solving one forward optimal control problem, these two methods solve N forward problems, one corresponding to the boundary conditions



(a) Sampled-data observations.

(b) Noisy sampled-data observations.

Figure 2.9: (a) Comparison of methods under sampled-data observations that are then converted to continuous-time observations using cubic spline interpolation. This figure shows how the error in the learned cost function parameter vector c changes for varying magnitude sample period (x-axis). (b) Comparison of methods under noisy sampled-data observations that are then converted to continuous-time observations using cubic spline interpolation. This figure shows how the error in the learned cost function parameter vector c changes for varying magnitude additive Gaussian noise (x-axis). The sample period is held fixed in this set of data at one percent the total time of the trajectories.

appropriate for each of the observed trajectories. In our new approach, the primary computation is the solution of a Riccati differential equation of dimension $(k + Nn) \times (k + Nn)$

$$\dot{P} - (PB + S)\, R^{-1}\, (PB + S)^{T} + Q = 0 \tag{2.51}$$

where A, B, Q, R, S were derived in Section 2.2.2. This matrix differential equation is, however, very sparse. The sparse structure of these matrices and the fact that P is symmetric allow us to partition P in the following way (where the diagonal and upper right parts of P are shown)

$$P = \begin{bmatrix} P_{11} & P_{12}^{(1)} & P_{12}^{(2)} & \cdots & P_{12}^{(N)} \\ \cdot & P_{22}^{(1)} & 0 & \cdots & 0 \\ \cdot & \cdot & P_{22}^{(2)} & 0 & \cdots \\ \vdots & \vdots & & \ddots & \vdots \\ \cdot & \cdots & & & P_{22}^{(N)} \end{bmatrix} \tag{2.52}$$

where $P_{11} \in \mathbb{R}^{k\times k}$, $P_{12}^{(i)} \in \mathbb{R}^{k\times n}$, and $P_{22}^{(i)} \in \mathbb{R}^{n\times n}$. After some algebra, the differential equations for these partioned matrices become

$$\begin{aligned}\dot{P}_{11} = \sum_{i=1}^{N} \Bigg\{ & P_{12}^{(i)} \left(P_{12}^{(i)}\right)^T + P_{12}^{(i)} \left(\nabla_x \phi^{(i)}\right)^T \\ & + \nabla_x \phi^{(i)} \left(P_{12}^{(i)}\right)^T - \nabla_u \phi^{(i)} \left(\nabla_u \phi^{(i)}\right)^T \Bigg\}\end{aligned} \tag{2.53}$$

$$\begin{aligned}\dot{P}_{12}^{(i)} &= P_{12}^{(i)} P_{22}^{(i)} + P_{12}^{(i)} \left(\nabla_x f^{(i)}\right)^T \\ &\quad + \nabla_x \phi^{(i)} P_{22}^{(i)} - \nabla_u \phi^{(i)} \left(\nabla_u f^{(i)}\right)^T\end{aligned} \tag{2.54}$$

$$\begin{aligned}\dot{P}_{22}^{(i)} &= P_{22}^{(i)} P_{22}^{(i)} + P_{22}^{(i)} \left(\nabla_x f^{(i)}\right)^T \\ &\quad + \nabla_x f^{(i)} P_{22}^{(i)} - \nabla_u f^{(i)} \left(\nabla_u f^{(i)}\right)^T\end{aligned} \tag{2.55}$$

For N observations, our new method involves solving N differential equations of size $n \times n$, N differential equations of size $k \times n$, and one differential equation of size $k \times k$. This shows that our new method grows linearly with the number of observations used to infer the unknown cost function.

Chapter 3

Calibration of the Kirchoff Elastic Rod

3.1 Introduction

Figure 3.1 shows a thin, flexible wire of fixed length that is held at each end by a robotic gripper. Such thin elastic rods have been of mathematical and engineering interest for centuries, beginning with the study of equilibrium shapes of planar rods by Euler and the Bernoullis [65]. Kirchhoff would later extended Euler's analysis to three-dimensional rods [66], while Max Born would be the first to show agreement between the theory and experiments [67]. Today, the equilibrium shapes of elastic rods and their stability have been studied extensively [68, 69]. Much work has also been done on the dynamics of elastic rods [70]. These theoretical investigations have found applications in a variety of environments, both natural and engineered. Examples of natural structures that can be modeled as elastic rods are human hair [71], twining plants [72], ripples in plant leaves [73], and DNA [74–76]. Engineered examples include electrical cables, wires, rope, deep-sea cables [77], flexible pipelines used in offshore drilling [78], carbon nanotubes [54, 79, 80] and graphene sheets [81, 82].

Despite this large collection of work on elastic rods, one seemingly fundamental problem remains challenging: Consider a thin elastic rod of fixed length that is held at each end by a robotic gripper. Given starting and goal equilibrium shapes of the elastic rod, find a path of each gripper that causes the rod to move between the two shapes while remaining in static equilibrium and avoiding self-collision. Equivalently, one can think of the problem as finding a path *of the rod* through its set of equilibrium shapes, beginning at the start configuration and ending at the goal configuration. This set of equilibrium shapes is the set of all configurations of the rod that would be in equilibrium if both ends of the rod were held fixed by the robotic grippers.

This problem is challenging for multiple reasons. First, an equilibrium

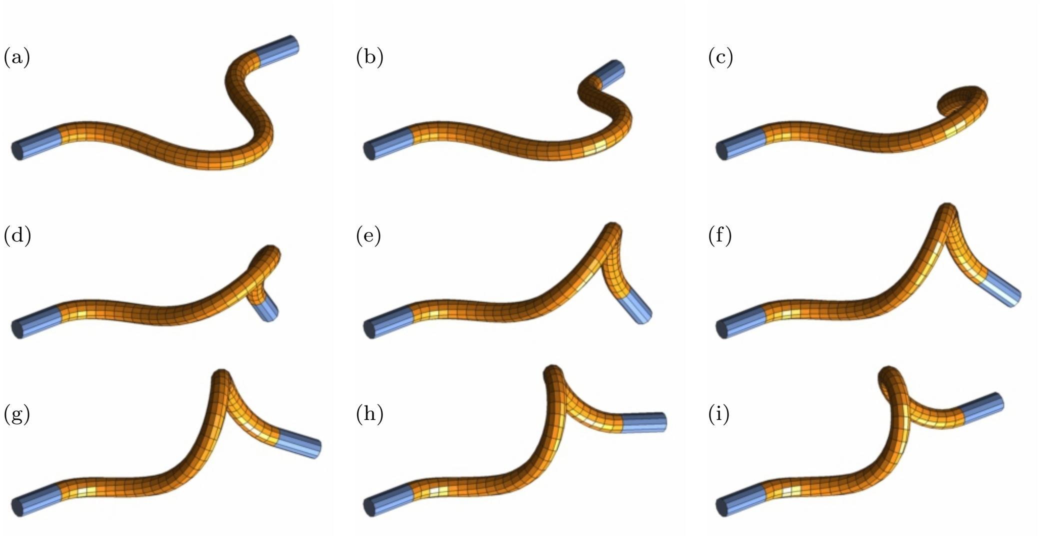


Figure 3.1: Quasi-static manipulation of an elastic rod (orange) by robotic grippers (blue). Notice that the grippers begin (frame a) and end (frame i) in the same position and orientation. This motion corresponds to a single straight-line path in the global coordinate chart derived in [4].

shape of the rod is a continuous map $q : [0,1] \to SE(3)$. Therefore, the configuration space is infinite dimensional. Second, configurations of the rod in general cannot be computed in closed form and must be approximated using numerical techniques. Lastly, specifying the position and orientation of the two robotic grippers does not uniquely determine the configuration of the rod. In Figure 3.1, note that configurations (a) and (i) have the same robotic gripper placements, but are different elements of the configuration space. Thus, simply moving the robotic grippers from their starting placement to their goal placement does not guarantee that the rod will move from its starting configuration to its goal configuration. For these reasons, previous literature addressing this problem suggests planning in the configuration space indirectly, by sampling displacements of the robotic grippers and numerically computing the resulting configuration of the elastic rod. This approach was developed in the seminal work of Lamiraux and Kavraki [83] and later applied to manipulation of “deformable linear objects” by Moll and Kavarki [84]. This previous work states that manipulation planning should be done in the configuration space of the elastic rod. However, the indirect method described above is ultimately used.

A novel approach to manipulation planning for an elastic rod was developed by Bretl and McCarthy [85], in which the rod was modeled as a

Kirchhoff elastic rod [86]. The main result in this work is that the configuration space of the rod is a smooth six dimensional manifold that can be parameterized by a single (global) coordinate chart. This was shown by formulating the problem of finding equilibrium shapes of the rod in an optimal control framework. Equilibrium configuration were shown to be local solutions to a geometric optimal control problem, with boundary conditions that vary with the position and orientation of each robotic gripper [86,87]. Coordinates for the configuration space of the rod (i.e., for *all* equilibrium shapes over *all* boundary conditions) are provided by the initial value of costates that arise in necessary and sufficient conditions for optimality. These coordinates describe all possible equilibrium configurations of the rod that can be achieved by moving the robotic grippers. This coordinate chart makes the seemingly difficult problem of manipulation planning easy to solve. This work is an extension of a similar manipulation planning method for planar elastic kinematic chains [85] and was implemented in hardware experiments for a planar elastic rod [88].

Although these experimental results are proof-of-concept, we are motivated by a variety of applications. Common manufacturing tasks that involve handing and assembly of deformable objects are fixturing of sheet metal [89–91], cutting and layup of composites [92,93], installation of a wire harness [94], and assembly of flexible circuit boards [95–98]. Medical procedures and equipment that could benefit from this work include automated knot tying and surgical suturing [99–103], retraction of tissue [104], and manipulation of flexible needles [105]. Other applications include cable routing [106], folding clothes [107, 108], and protein folding [109]. Finally, we are motivated by the link between manipulation of deformable objects and control of hyper-redundant [110] and continuum robots [111,112], as pointed out by Tanner [113].

Two main approaches to manipulation planning for deformable objects have been taken in previous literature. One relies on numerical simulations of the objects, while the other uses task-based decomposition. The first approach is considered by Moll and Kavraki in [84], in which they propose a sampling-based planning algorithm for quasi-static manipulation of an inextensible elastic rod by robotic grippers in a three-dimensional workspace. Equilibrium configurations of the rod are those that locally minimize the total elastic energy. The algorithm samples placements of the robotic grippers and then numerically approximates equilibrium configurations that satisfy these boundary conditions. The distance between configurations is measured by the integral of the sum-squared difference in curvature and torsion of the rod, and nearby configurations are connected by spherical interpolation of the gripper placement. The resulting path of the rod between nearby configurations is again approximated numerically. The choice of numerical method used has a significant impact on the performance of this approach. While Moll and Kavarki [84] used recursive subdivision, other potential methods include finite elements, finite differences, and discrete geometric models of elastic rods [114]. The second approach involves tasks that are topological rather than geometric. One such task is knot tying with rope, in which the sequence of crossings of the rope is much more important that the exact shape. Motion primitives can be designed to ensure that crossing operations are realizable by robotic grippers. Such primitives may rely on the rope begin placed on a table [103] or being held by fixtures [102]. Another example of a topological goal is folding of objects such paper [115] and clothes [107].

3.2 Model

We refer to the object in Figure 3.1 as a *rod*. Assuming that it is thin, inextensible, and of unit length, we describe the shape of this rod by a continuous map $q\colon [0,1] \to G$, where $G = SE(3)$. As defined in Bretl and McCarthy [4], let L_q denote the left translation map $L_q : G \to G$. Let e denote the identity element of G, and let $\mathfrak{g} = T_eG$ and $\mathfrak{g}^* = T_e^*G$. Abbreviating $T_eL_q(\zeta) = q\zeta$ as usual for matrix Lie groups, we require this map to satisfy

$$\dot{q} = q(u_1X_1 + u_2X_2 + u_3X_3 + X_4) \tag{3.1}$$

for some $u\colon [0,1] \to U$, where $U = \mathbb{R}^3$ and

$$X_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad X_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad X_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$X_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad X_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad X_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

is a basis for $\mathfrak{g}$. Denote the dual basis for $\mathfrak{g}^*$ by $\{P_1, \ldots, P_6\}$. We refer to q and u together as $(q, u)\colon [0,1] \to G \times U$ or simply as (q, u). Each end of the rod is held by a robotic gripper. We ignore the structure of these grippers, and simply assume that they fix arbitrary $q(0)$ and $q(1)$. We further assume, without loss of generality, that $q(0) = e$. We denote the space of all possible $q(1)$ by $\mathcal{B} = G$. Finally, we assume that the rod is elastic in the sense of Kirchhoff [86], so has total elastic energy

$$\frac{1}{2}\int_0^1 \left(c_1 u_1^2 + c_2 u_2^2 + c_3 u_3^2\right) dt$$

for given constants $c_1, c_2, c_3 > 0$. For fixed endpoints, the rod will be motionless only if its shape locally minimizes the total elastic energy. In particular, we say that (q, u) is in static equilibrium if it is a local optimum of

$$\begin{aligned} \underset{q,u}{\text{minimize}} \quad & \frac{1}{2}\int_0^1 \left(c_1 u_1^2 + c_2 u_2^2 + c_3 u_3^2\right) dt \\ \text{subject to} \quad & \dot{q} = q(u_1 X_1 + u_2 X_2 + u_3 X_3 + X_4) \\ & q(0) = e, \qquad q(1) = b \end{aligned} \tag{3.2}$$

for some $b \in \mathcal{B}$.

The problem of *inverse optimal control* is to infer the unknown parameters with respect to which a given trajectory, the *observation*, is a local minimum to problem (3.2). This observed trajectory is denoted by

$$(q^*, u^*) = \{q^*(t), u^*(t) : t \in [0,1]\}. \tag{3.3}$$

3.2.1 Necessary Conditions for Static Equilibrium

The new method of inverse optimal control derived in Section 2.2 will be modified to handle the geometric optimal control problem 3.2. We have seen that if a Kirchhoff elastic rod is in static equilibrium, then its configuration (q, u) must be a local solution to the geometric optimal control problem (3.2). In this section, we apply necessary conditions for optimality to show that the set of all normal (q, u) is a smooth six-manifold that can be parameterized by a single chart. Coordinates for this chart are given by the open subset $\mathcal{A} \subset \mathbb{R}^6$ that is defined by (3.7) in the following theorem.

Theorem 5. *A trajectory* (q, u) *is normal with respect to* (3.2) *if and only if there exists* $\mu\colon [0,1] \to \mathfrak{g}^*$ *that satisfies*

$$\begin{aligned}
\dot{\mu}_1 &= u_3\mu_2 - u_2\mu_3 & \dot{\mu}_4 &= u_3\mu_5 - u_2\mu_6 \\
\dot{\mu}_2 &= \mu_6 + u_1\mu_3 - u_3\mu_1 & \dot{\mu}_5 &= u_1\mu_6 - u_3\mu_4 \\
\dot{\mu}_3 &= -\mu_5 + u_2\mu_1 - u_1\mu_2 & \dot{\mu}_6 &= u_2\mu_4 - u_1\mu_5,
\end{aligned} \tag{3.4}$$

$$\dot{q} = q(u_1X_1 + u_2X_2 + u_3X_3 + X_4), \tag{3.5}$$

$$u_i = c_i^{-1}\mu_i \quad \textit{for all } i \in \{1, 2, 3\}, \tag{3.6}$$

with initial conditions $q(0) = e$ *and* $\mu(0) = \sum_{i=1}^{6} a_iP_i$ *for some* $a \in \mathcal{A}$*, where*

$$\mathcal{A} = \left\{a \in \mathbb{R}^6 \colon (a_2, a_3, a_5, a_6) \neq (0, 0, 0, 0)\right\}. \tag{3.7}$$

These necessary conditions will be used to derive residual functions analogous to those derived in Section 2.2. We will again say that the system is approximately optimal when these residual functions are close to zero. The problem is now to derive an efficient solution to the minimization of the residual functions given the system equations and costate equations.

3.3 Simulation Experiments

3.3.1 Perfect Observations with Known Basis Functions

In this set of experiments, each algorithm was given one perfect observation of an optimal trajectory and learned the unknown cost function parameters c.

After learning the cost function, predicted trajectories are computed. This allows us to compute other statistics such as the error in total cost, error in feature vectors, and sum squared error between observed and predicted trajectories.

Table 3.1 shows results averaged over 50 trials with randomly selected boundary conditions in each trial. In the method of Mombaur, the sum-squared error between predicted and observed trajectories converges near zero as the number of iterations increases. However the inferred cost function parameters are not learned perfectly. Similarly, upon termination of the methods of Abbeel and Ratliff, the error between predicted and observed feature vectors is small, but the cost function parameters are not learned perfectly.

Table 3.1: Results for perfect observations with known basis functions.

System	Error Type	Mombaur	Abbeel	Ratliff	New
Elastic Rod	computation (s)	95	9	15	1
	forward problems	71	5	10	0
	parameter error	3.38e-2	8.92e-1	9.71e-1	3.96e-5
	feature error	6.77e-7	6.24e-3	4.48e-3	4.87e-7
	trajectory error	1.94e-5	7.95e-3	8.82e-3	6.14e-6

The new method developed in this paper also performs as expected – learning the unknown parameters perfectly (within the accuracy and precision tolerances of ODE and least squares solvers).

3.3.2 Perfect Observations with Perturbed Cost

In this set of experiments, the true cost function consists of a linear combination of known basis functions plus a bounded deterministic perturbation (see Section 2.3.1). For each system, one particular set of boundary conditions was selected, and observations of optimal trajectories are gathered for a range of perturbation magnitudes. Figure 2.7 shows the performance of each method over varying magnitude perturbations. These results generally show:

- All of the methods learn cost functions that are able to approximate the observation in terms of feature vector and trajectory errors.

- The performance of the iterative methods remains close to the results obtained with known basis functions for small perturbations, and then degrades at larger perturbations,
- The performance of our new method (KKT) continues to improve as the perturbation descreases, reflecting exact recovery of the cost function (to specified numerical method tolerances).

Note that in the case of the elastic rod, all of the methods, including our new approach, stop improving as the perturbation magnitude gets small. This trend occurs because the numerical method for solving the forward optimal control problem terminates before reaching the observed local minima under our standard convergence and tolerance parameters that are fixed for all experiments.

3.4 Hardware Experiments

In this section, we perform hardware experiments analogous to those performed in simulation in the previous section. We place the rod in a sequence of static equilibrium configurations, and use a camera motion tracking system to detect sampled locations along the length of the rod. We then use cubic spline interpolation to generate continuous observations. We manipulated a 33 cm long steel cable, with one end of the cable rigidly attached to the ground and the other end held by an Adept industrial robot arm, see Figure 3.2. The cross-section of the cable was approximately circular and was constant along the length of the rod. Therefore the bending stiffnesses c_2 and c_3 are approximately equal. Figure 3.3 shows the observations used in our inverse optimal control method.

Our observations of the rod begin as sparsely sampled-data observations of position markers along the rod, and orientation at the endpoints and at a subset of the positions along the rod. To generate continuous observations of the rod configuration, we separately interpolate the position and orientation of the rod sampled measurements. Interpolation of the position measurements is performed in a standard way using cubic smoothing splines. Interpolation of the orientation is spline technique on Lie Groups as developed in [116], and related to other interpolation techniques over rotations [117–119]. Figure 3.4 shows an example of the rod configuration

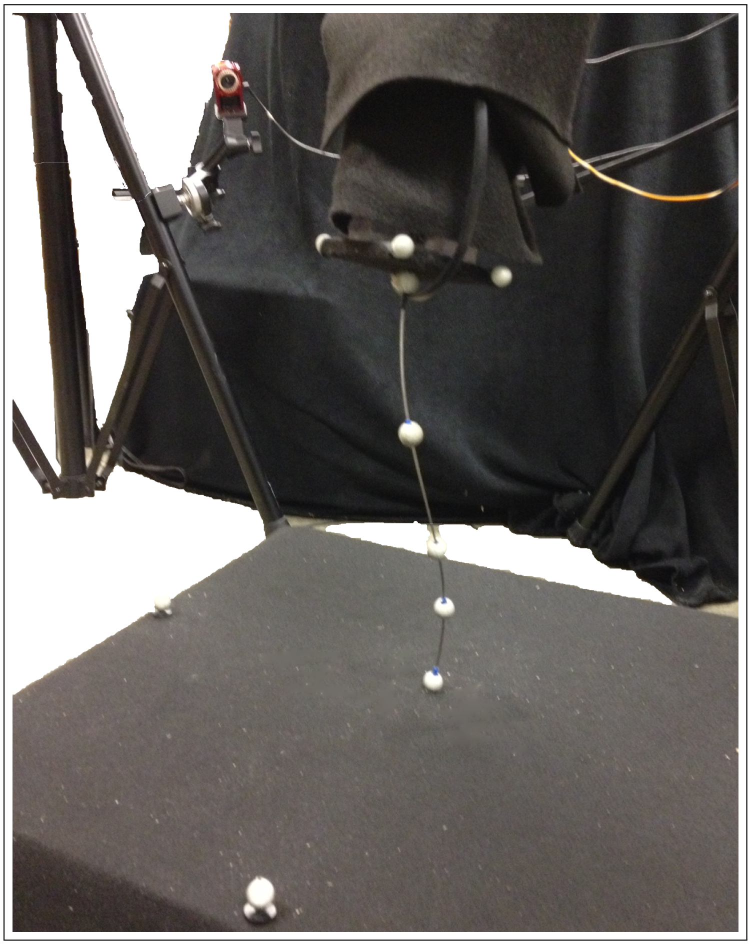
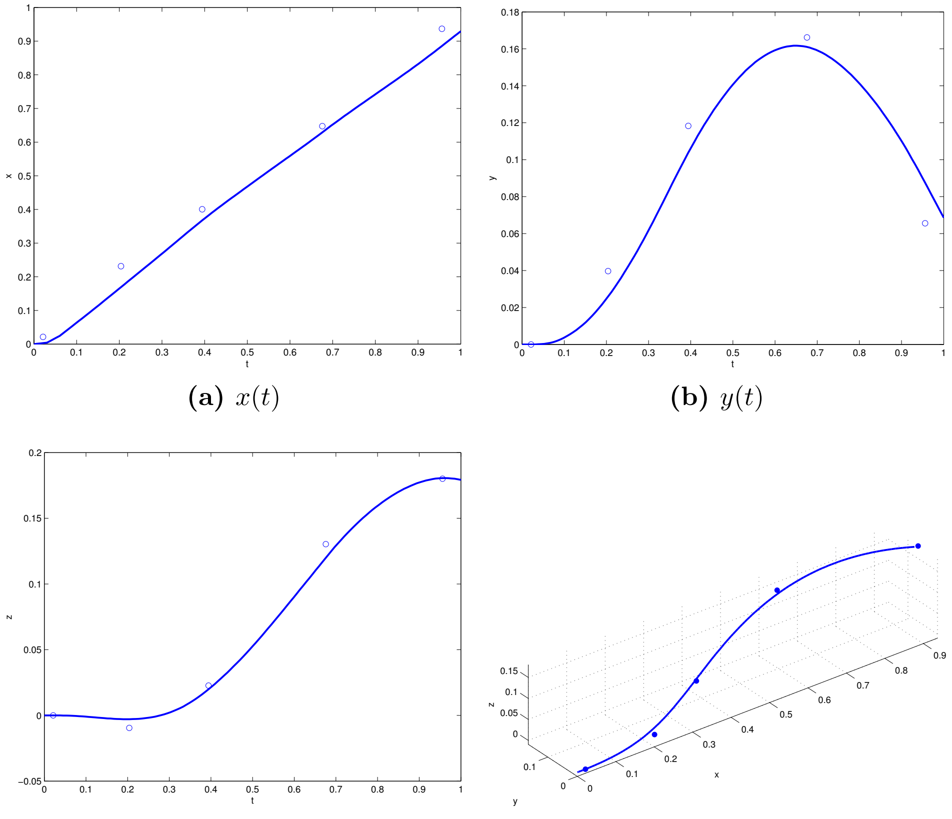


Figure 3.2: Hardware experimental setup for the 3D elastic rod. A steel cable is fixed at one end to a table and held at the other end by an Adept industrial robot.



(a) $x(t)$

(b) $y(t)$

(c) $z(t)$

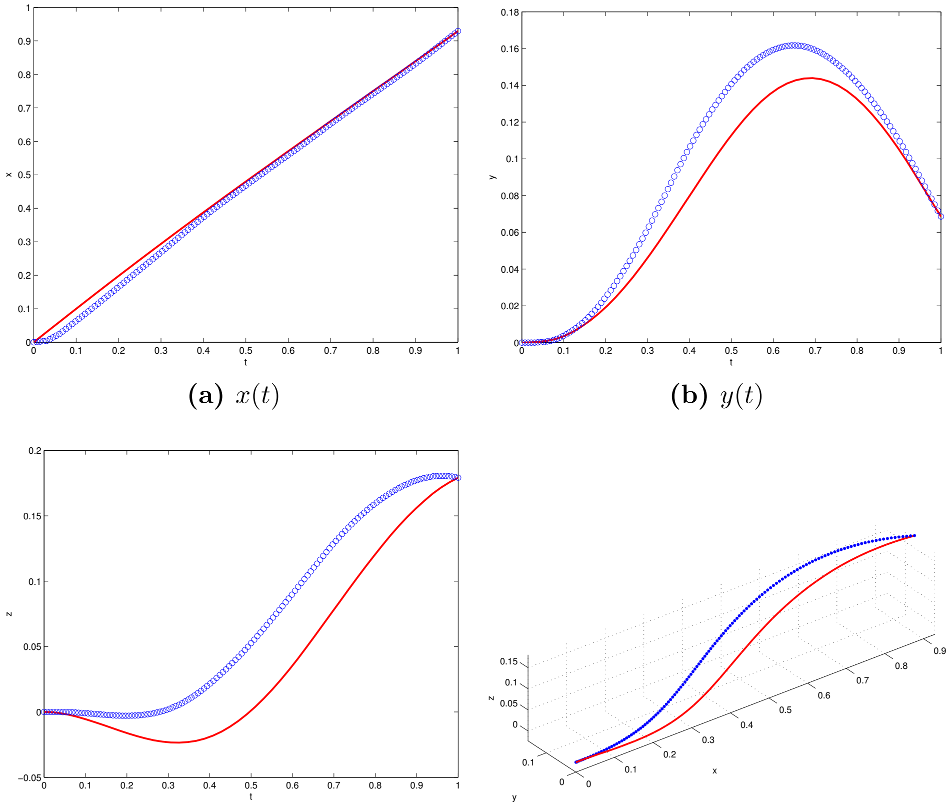
(d) 3D plot

Figure 3.3: Example observation of physical 3D elastic rod used for inverse optimal control. Here, t denotes the arc length parameter, and the length of the rod has been normalized to 1. The circles denote the raw measurements of position markers along the rod. The solid curve represents our spline interpolation of those markers.

predicted after learning the stiffness parameters of the rod using our inverse optimal control method. Table 3.2 shows a comparison of results for all of our implementations of inverse optimal control, where the results are averaged over two observations of the rod.

Comparison to Ideal Model

For an ideal elastic rod the bending stiffness is equal to the Young's modulus of the cable, E, times the area moment of inertia, I, of the cross-section of the rod. The torsional stiffness, c_1, is equal to the shear modulus of the rod, G, times the cross-sectional polar moment of inertia of the rod, J. The elastic potential energy of the rod can be normalized by the bending stiffness



(a) $x(t)$ **(b)** $y(t)$

(c) $z(t)$ **(d)** 3D plot

Figure 3.4: This figure shows the predicted configuration of the elastic rod after estimating the physical properties of the rod using inverse optimal control. The blue samples represent the interpolated measurement of the observed rod. The red curve shows the configuration of the rod given the initial conditions and learned physical properties – i.e. it is the solution of the forward optimal control problem after we learn the cost function via inverse optimal control. Errors here can be due to observation noise, model inaccuracy (e.g. we ignore gravity), and plastic deformation of the steel wire that we model as a perfectly elastic rod.

Table 3.2: Hardware results.

System	Error Type	Mombaur	Abbeel	Ratliff	New
Elastic Rod	computation (s)	424	132	93	1
	forward problems	26	15	13	0
	feature error	0.521	0.618	0.715	0.532
	trajectory error	11.4	12.3	13.6	12.5

	c_1	c_2	c_3
Theoretical model	0.77	1	1
Learned value from IOC	0.77	0.4	0.3

Table 3.3: Learned physical properties of the elastic rod from inverse optimal control. Errors here can be due to observation noise, model inaccuracy (e.g. we ignore gravity), and plastic deformation of the steel wire that we model as a perfectly elastic rod.

as follows:

$$\frac{1}{2}\int_0^1 \left(\frac{GJ}{EI} u_1^2 + u_2^2 + u_3^2 \right) dt \tag{3.8}$$

For circular cross sections, we have

$$I = \frac{\pi}{4} r^4 \qquad J = \frac{\pi}{2} r^4 \tag{3.9}$$

where r is the radius of the cross-section. We also have the following relationship between E and G [120]:

$$G = \frac{E}{2(1+\nu)} \tag{3.10}$$

where ν is the Poisson's ratio of the material. Therefore we have

$$\frac{GJ}{EI} = \frac{1}{1+\nu} \tag{3.11}$$

The Poisson's ratio of steel is approximately $\nu \approx 0.3$ [120].

One source of error is in the assumptions made when deriving the stiffnesses c_1, c_2, and c_3. The steel cable consists of multiple smaller steel cables braided together, so the cross-section is not exactly circular. This problem does not arise with the planar rod, as the elastic energy can be normalized by the bending stiffness and then no stiffnesses appear in the normalized elastic energy. Other sources of error include uncertainty in the length of the rod, uncertainty in the position and orientation of each endpoint, uncertainty in the motion capture data, and non-constant stiffnesses along the rod. Also, since we assumed that the rod had a naturally straight shape, uncertainty in the unstressed shape of the rod could have contributed to the error. Finally, we note that a portion of the error can be attributed to the fact that the weight of the rod due to gravity was ignored in this analysis.

In future work, the weight of the rod due to gravity could be added into the analysis, although this complicates the application of Lie-Poisson reduction. With gravity, the Hamiltonian is no longer left-invariant.

Chapter 4

Modeling Human Locomotion

4.1 Introduction

Human locomotion is studied from many different perspectives. In this application of inverse optimal control, we will study the natural high-level trajectories that humans take as they walk from an initial rest position to a given target position and orientation. That is, we are not concerned with the biomechanical modeling of joint actuation and kinematics, and only with the selection and execution of trajectories in the plane. In recent work, Mombaur, et al. [1] derived a system model that captures relevant dynamics and investigates a particular cost function that is able to approximate human walking trajectories. In our own previous work [46] we developed a method of discrete-time inverse optimal control and validated the approach using a discrete-time unicycle model of human locomotion. Experimental data came from a database of human walking trajectories provided to us by Gustavo Arechavaleta, Jean-Paul Laumond, Halim Hicheur, and Alain Berthoz [121]. In summary, subjects were asked to walk in a gymnasium from a starting point to a final destination represented by a porch. The starting point was always the same, but the final position and final orientation of the porch were varying. The subjects were asked to walk from one point to another freely, without time or velocity constraints, and the trajectories were recorded using motion capture technology. An example of a subset of eight observed trajectories for one subject is presented in Figure 4.1.

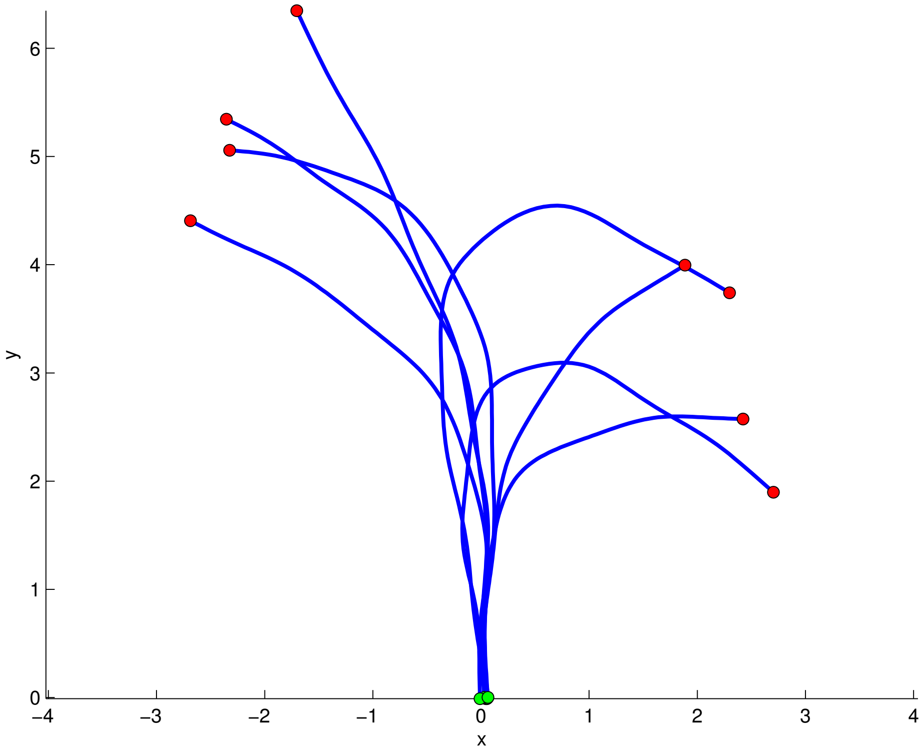


Figure 4.1: This figure shows eight examples of human walking trajectories captured by a motion capture system capable of tracking the human subjects' torsos. The subjects were asked to start at a fixed initial condition and walk freely to a final destination designated by a gate that the subjects should walk through. Here, green circles represent the starting position, and red circles represent the terminal position that typically coincided with a non-zero terminal velocity.

4.2 Optimal Control Model of Human Locomotion Paths

We will consider the following locomotion model that was first described in [1]. The system dynamics are given as follows

$$\begin{aligned}
\dot{x} &= v_1 \cos\theta - v_2 \sin\theta \\
\dot{y} &= v_1 \sin\theta + v_2 \cos\theta \\
\dot{\theta} &= \omega \\
\dot{v}_1 &= u_1 \\
\dot{v}_2 &= u_2 \\
\dot{\omega} &= u_3
\end{aligned} \tag{4.1}$$

where x, y, θ denote the position and orientation of the system in the plane, v_1, v_2 denote the forward and sideward velocities in the body-fixed reference frame, and ω denotes the angular velocity of the system. The inputs u_1, u_2, u_3 represent forward, sideward, and rotational accelerations, respectively. The cost function is modeled as a linear combination of basis functions that penalize time, input energy, and squared-error between body orientation and direction to the goal. This cost function is given by

$$J(x(t), u(t)) = \int_{t_0}^{t_f} c^T \phi[t, x(t), u(t)] dt \tag{4.2}$$

where the basis functions are

$$\phi[t, x(t), u(t)] = \begin{bmatrix} 1 \\ u_1(t)^2 \\ u_2(t)^2 \\ u_3(t)^2 \\ \psi[x(t), x_{goal}]^2 \end{bmatrix} \tag{4.3}$$

and

$$\psi[x(t), x_{goal}] = \arctan\left(\frac{y_f - y(t)}{x_f - x(t)}\right) - \theta(t) \tag{4.4}$$

where x_f, y_f are the position coordinates of the goal configuration x_{goal}. The problem of inverse optimal control is now to learn the values of c that make

Table 4.1: Results from human walking data experiments.

System	Error Type	Mombaur	Abbeel	Ratliff	New
Locomotion	computation (s)	138	64	41	2
	forward problems	58	31	27	0
	feature error	0.186	0.109	0.853	0.716
	trajectory error	0.135	0.102	0.761	0.358

observations of human walking trajectories local minima of problem 1.1 with the cost function and system dynamics defined in this section.

4.3 Experimental Results

We now apply our new method of inverse optimal control, as well as the three existing methods for comparison, to the model of human locomotion defined by equations (4.1) through (4.3). Experimental data came from a subset of the database of human walking trajectories provided to us by Gustavo Arechavaleta, Jean-Paul Laumond, Halim Hicheur, and Alain Berthoz [121]. For this subset of trajectories, shown in Figure 4.1, we perform inverse optimal control independently on each single trajectory. We then perform inverse optimal control given the set of all trajectories as observations, and learn one cost function that best models the set of observations. Results from our new method of inverse optimal control are shown in Figures 4.2 and 4.3 that show trajectories produced by solving the forward optimal control problem using the cost function learned by our inverse optimal control algorithm. In Figure 4.2, we performed inverse optimal control independently on each observed trajectory. In Figure 4.3 we used all eight observations to compute one cost function. One can see that better results are obtained when using multiple observations to recover the cost function. These results suggest that more observations used to compute the basis weights c, the better the cost function will predict observed trajectories.

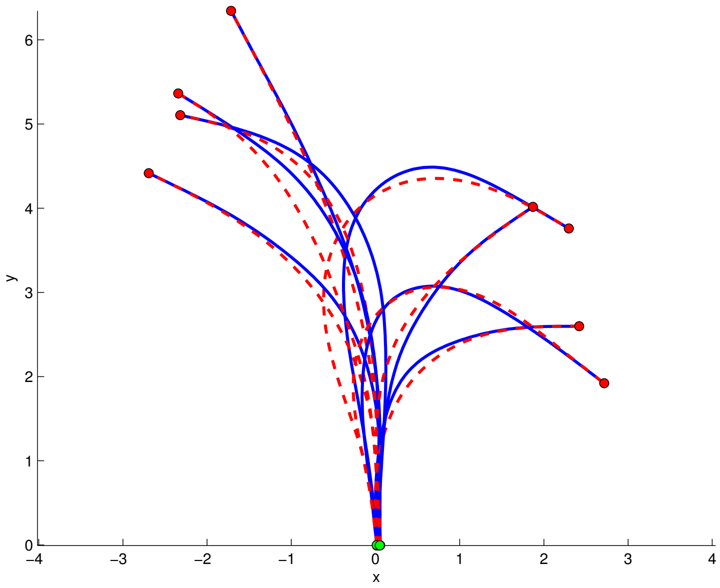


Figure 4.2: In this figure, observed and predicted trajectories are projected on the x-y plane. Blue curves represent observed trajectories obtained from experimental data. Red dashed curves show predicted results obtained when using individual observations to recover the value of parameter c independently for each trajectory.

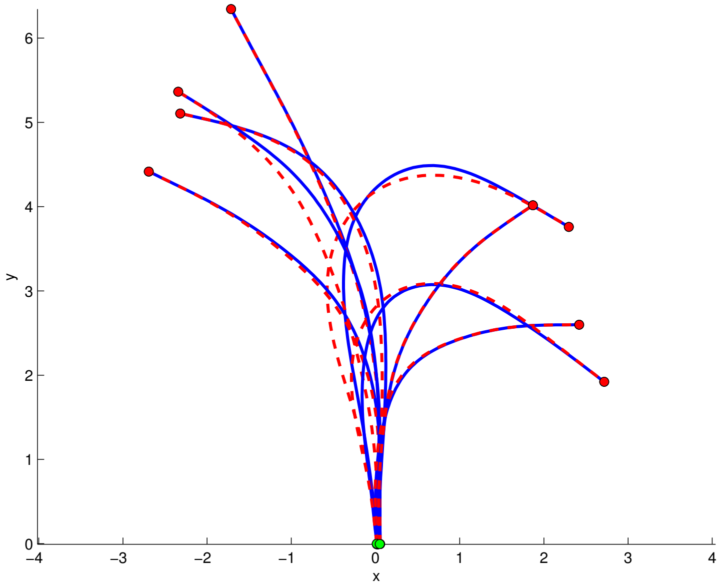


Figure 4.3: In this figure, observed and predicted trajectories are projected on the x-y plane. Blue curves represent observed trajectories obtained from experimental data. Red dashed curves show predicted results obtained when using all observations to recover the value of parameter c.

Chapter 5

Learning Quadrotor Dynamic Maneuvers

5.1 Introduction

Inverse optimal control is a tool that can be used to learn a cost function for the purpose of developing a convenient representation of behavior that can be generalized to new domains. There are two sources of computational complexity that have typically limited the application of inverse optimal control to low-dimensional problems. First, there is the complexity of the inverse optimal control algorithm itself. Standard methods have utilized iterative approaches that require the solution of an optimal control problem in an inner loop. Second, the particular instance of a single forward optimal control problem can often be difficult and computationally expensive to solve to high precision. In this chapter, we develop an approximate inverse optimal control algorithm that overcomes these challenges. In particular, from human flight demonstrations we learn cost functions for a highly dynamic and nonlinear quadrotor flight task – rapidly translating from a given initial hover condition to hover at a desired goal position. We will compare our learning from demonstration method with an existing numerical method that solves a minimum-time formulation of this problem which we will refer to as the *ETH Zürich method* [122,123].

5.1.1 Problem Statement

Consider the standard quadrotor system (e.g. derived in [124–128])

$$\begin{aligned}
\dot{q} &= v \\
\dot{v} &= \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} R_1^0 \begin{bmatrix} 0 \\ 0 \\ -u_4 \end{bmatrix} \\
\dot{\theta} &= S\omega \\
\dot{\omega} &= J^{-1} \left(\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} - \omega \times J\omega \right)
\end{aligned} \tag{5.1}$$

where the position is denoted by $q = [q_1, q_2, q_3]^T$, the velocity $v = [v_1, v_2, v_3]^T$, Euler angles $\theta = [\theta_1, \theta_2, \theta_3]^T$, body angular velocity $\omega = [\omega_1, \omega_2, \omega_3]$, mass m, moment of inertia $J = \text{diag}(J_1, J_2, J_3)$, and $R_1^0 \in SO(3)$ rotates vectors from the body frame 1 to the world frame 0. The input to the system consists of roll, pitch, and yaw torques and total thrust, and is denoted by $u = [u_1, u_2, u_3, u_4]^T \in \mathbb{R}^4$. We consider the full state of the quadrotor system to be

$$x = [q; v; \theta; \omega]$$

Task: The flight task we consider is rapid translation of the quadrotor from a specified initial condition x_0 to a desired goal position x_{goal}. Recent related work approaches this task as a minimum-time optimal control problem. However, features of this flight task make such a formulation difficult:

- High-acceleration maneuvers are difficult to engineer due to unmodeled dynamics and approximate knowledge of the system's physical characteristics.

- Exact solutions to constrained minimum-time optimal control problems for highly nonlinear systems in complex environments are computationally prohibitive.

- Recent solutions typically develop open loop controllers or similar solutions that are designed for one specific initial condition and goal, and do not generalize over direction of flight nor distance of motion.

Goal: Our goal is to *learn* a time-invariant policy that captures the desired flight task behavior and generalizes over direction and distance of motion. In particular, we will learn such a policy by observing *human demonstrations* of the flight task. The learning from demonstration approach to this problem poses the flight task to the human pilot as follows: fly from hover at initial condition x_0 to hover at the goal position x_{goal} as quickly as possible. We expect the result of these flights to very closely approximate solutions to the formal minimum time optimal control problem given as follows.

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & \dot{x}(t) = f(t, x(t), u(t)) \\ & x(0) = x_0 \\ & x(t_f) = x_{goal} \end{aligned} \tag{5.2}$$

We compare our learning approach to an existing solution from the Flying Machine Arena at ETH Zürich [122, 123]. As mentioned above, this recent state-of-the-art method has yielded interesting results in terms of automating very particular high-acceleration maneuvers such as multiple flips. The method runs into the issues raised above: it's implementation must be carefully tuned to handle model inaccuracies of the particular hardware system used, its solutions are specific to one precise maneuver in free-space and thus must be recomputed for every maneuver of interest. We show hardware results for both our method based on learning from demonstration and the ETH Zürich method.

5.2 Learning from Demonstration: Method

In this section we develop a method of learning from demonstration that begins with observations of human pilot flights and ultimately computes a time-invariant feedback policy capable of replicating the flight task while also generalizing over direction of flight and distance of motion. Our approach is composed of a combination of inverse optimal control and guided policy search, and is inspired by recent work in reinforcement learning [129, 130]. In reinforcement learning, direct policy search methods are often used to tackle high-dimensional problems in robotics [129, 131, 132]. However, it can often be beneficial to limit the class of policies so that convergence is achieved in fewer iterations without getting stuck in poor local optima. In our method, we aim to learn policies with general and flexible representations capable of representing a broad range of behaviors. An outline of our method is shown in Figure 5.1. In particular, our method begins with a small number of observations of human flights. We then take advantage of the differential flatness of the quadrotor dynamics to generate continuous-time full state trajectories given sampled-data observations of the position of the center of mass and quadrotor heading. Then, a central component of our method is to use inverse optimal control to learn a cost function for the full quadrotor model that efficiently represents the flight task. In particular, we use the learned cost function to generate simulated flights from novel initial conditions, conditions not seen in the human demonstrations. This provides us with a richer class of observed trajectories than that provided by human demonstrations alone. Finally, we define a desired class of feedback control policies and use direct policy search to find a time-invariant feedback policy capable of generating the desired flight task. In the following subsections, we will describe each of these components.

5.2.1 Quadrotor Differential Flatness

In our hardware experiments, we fly in an indoor laboratory fitted with an Optitrack motion capture system that tracks the position of the quadrotor center of mass and its orientation at 50Hz. In order to utilize our knowledge of the quadrotor's equations of motion, and our continuous-time method of inverse optimal control, we must generate continuous-time full-state trajectories from the sampled-data position and orientation measurements of the motion capture system. To do this, we take advantage of the differential flatness of the quadrotor dynamics, a feature that allows us to compute full state trajectories from observations of flat outputs. We follow the development in [133]. Consider flat outputs

$$y = [q_1, q_2, q_3, \theta_1]^T \tag{5.3}$$

The full state, x, of the system can be written as functions of $y, \dot{y}, \ddot{y}$, and $\dddot{y}$. The position and velocity are the first three components of y and $\dot{y}$. The

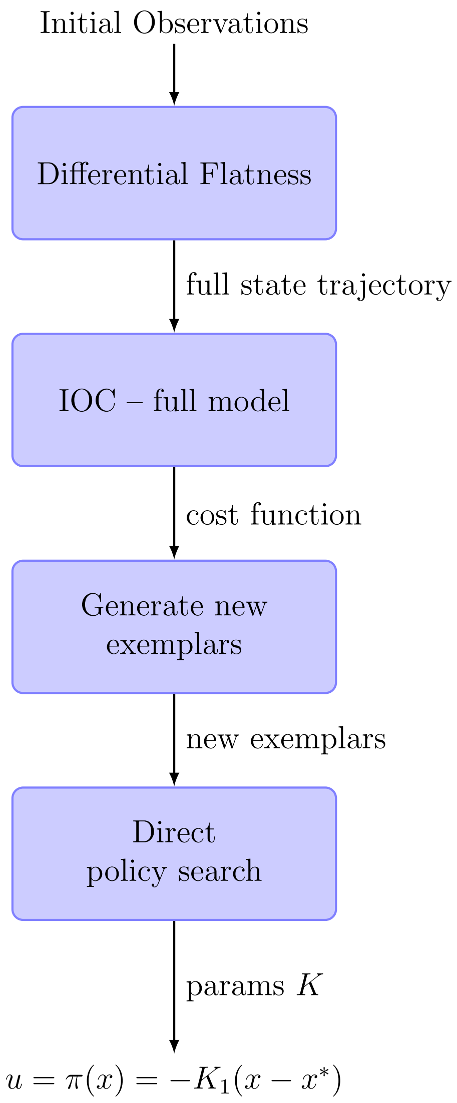


Figure 5.1: This figure shows an outline of our quadrotor learning from demonstration method. Inverse optimal control plays the critical role of efficiently representing the task in a way that can generalize.

rotation matrix R_1^0 is obtained as follows. First define the body frame z-axis z_1

$$z_1 = \frac{t}{\|t\|} \qquad t = [\ddot{y}_1, \ddot{y}_2, \ddot{y}_3 - g]^T \tag{5.4}$$

that points the body z-axis along the gravity corrected acceleration vector of the quadrotor center of mass. Next define

$$x_C = [\cos y_4, \sin y_4, 0]^T \tag{5.5}$$

Then

$$y_1 = \frac{z_1 \times x_C}{\|z_1 \times x_C\|} \qquad x_1 = y_1 \times z_1 \tag{5.6}$$

that yields

$$R_1^0 = [x_1 y_1 z_1] \tag{5.7}$$

The angular velocity ω is obtained as follows. Take the derivative of the velocity equations of motion (in the body frame) to get

$$m\dot{a} = -u_4 z_1 + \omega \times -u_4 z_1 \tag{5.8}$$

Project this vector along z_1 and use the fact that $\dot{u}_4 = z_1 \cdot -m\dot{a}$ to get

$$h = \omega \times z_1 = -\frac{m}{u_4} (\dot{a} - (z_1 \cdot \dot{a}) z_1) \tag{5.9}$$

where h is the project of $(m/u_4)\dot{a}$ onto the x_1–y_1 plane. Now

$$\begin{aligned} \omega_1 &= -h \cdot y_1 \\ \omega_2 &= -h \cdot x_1 \\ \omega_3 &= \omega \cdot z_1 = \dot{\theta}_1 z_0 \cdot z_1 \end{aligned} \tag{5.10}$$

The inputs u are computed as follows. First, $u_4 = m\|t\|$, the gravity-corrected acceleration. The other components of u can be solved using the Euler equations in the system equations (5.1).

5.2.2 Quadrotor Inverse Optimal Control

A central component of our method is to use inverse optimal control to learn a cost function for the full quadrotor model that efficiently represents the flight task. In particular, we use the learned cost function to generate simulated flights from novel initial conditions, conditions not seen in the human demonstrations. This provides us with a richer class of observed trajectories than that provided by human demonstrations alone.

We define the following basis functions for this flight task:

$$\phi(t, x(t), u(t)) = \begin{bmatrix} (q_1(t) - x_g)^2 \\ (q_2(t) - y_g)^2 \\ (q_3(t) - z_g)^2 \\ v_1(t)^2 \\ v_2(t)^2 \\ v_3(t)^2 \\ \theta_1(t)^2 \\ \theta_2(t)^2 \\ \theta_3(t)^2 \\ p(t)^2 \\ q(t)^2 \\ r(t)^2 \\ u_1(t)^2 \\ u_2(t)^2 \\ u_3(t)^2 \\ (u_4(t) - mg)^2 \end{bmatrix} \tag{5.11}$$

5.2.3 Generating Novel Exemplar Trajectories

We use the learned cost function to generate simulated flights from novel initial conditions, conditions not seen in the human demonstrations. This provides us with a richer class of observed trajectories than that provided by human demonstrations alone. To generate novel exemplar trajectories, we simply sample new initial and goal conditions around those seen in human demonstrations. For example, we will add zero-mean Gaussian noise to each component of the initial condition: position, velocity, orientation, and angular velocity, each with an independent standard deviation. We are then ready to solve the forward optimal control problem given the cost function learned via inverse optimal control in the previous section. We again use the generic numerical optimal control tool, GPOPS2, to solve the forward optimal control problem. The outcome of this step is a set of approximately optimal exemplar trajectories that we combine with the human demonstrations to compose the total set of trajectories that will be used in the next step of our method: direct policy search.

5.2.4 Direct Policy Search

We now define a desired class of feedback control policies and use direct policy search to find a time-invariant feedback policy capable of generating the desired flight task. In this particular flight task, we begin with a very straightforward class of policies, the class of time-invariant PD *outer loop* controllers. By outer loop controller we mean the following. Due to current quadrotor hardware designs, there is a natural decomposition of the quadrotor control in a two level hierarchical architecture. One level consists of an onboard fast *inner loop* controller that typically regulates attitude and angular velocity at up to 1000Hz. A slower, typically off-board *outer loop* controller regulates position errors. Thus, when we consider a class of feedback control policies for quadrotor position control tasks, we are generally talking about the outer loop control component. See Appendix A for a derivation of both quadrotor dynamics and the standard hierarchical control architecture. Ultimately, the outer loop controller can be thought of in the following context: consider a simplified model of the position dynamics of the quadrotor as a second order system

$$\ddot{x}(t) = u(t) \tag{5.12}$$

where $x(t) = (q_1, q_2, q_3)^T$ here is simply the 3D position in the world frame and $u(t) = (u_1, u_2, u_3, u_4)^T$ here represents desired 3D acceleration of the center of mass (u_1, u_2, u_3) and the yaw angular acceleration u_4. The class of time-invariant feedback controllers we will consider, PD controllers, can now be written as

$$\begin{aligned} u_1 &= -K_1(q_1 - q_{1,goal}) - K_4\dot{q}_1 \\ u_2 &= -K_2(q_2 - q_{2,goal}) - K_5\dot{q}_2 \\ u_3 &= -K_3(q_3 - q_{3,goal}) - K_6\dot{q}_3 \\ u_4 &= -K_8(\theta_1 - \theta_{1,goal}) - K_9\dot{\theta}_1 \end{aligned} \tag{5.13}$$

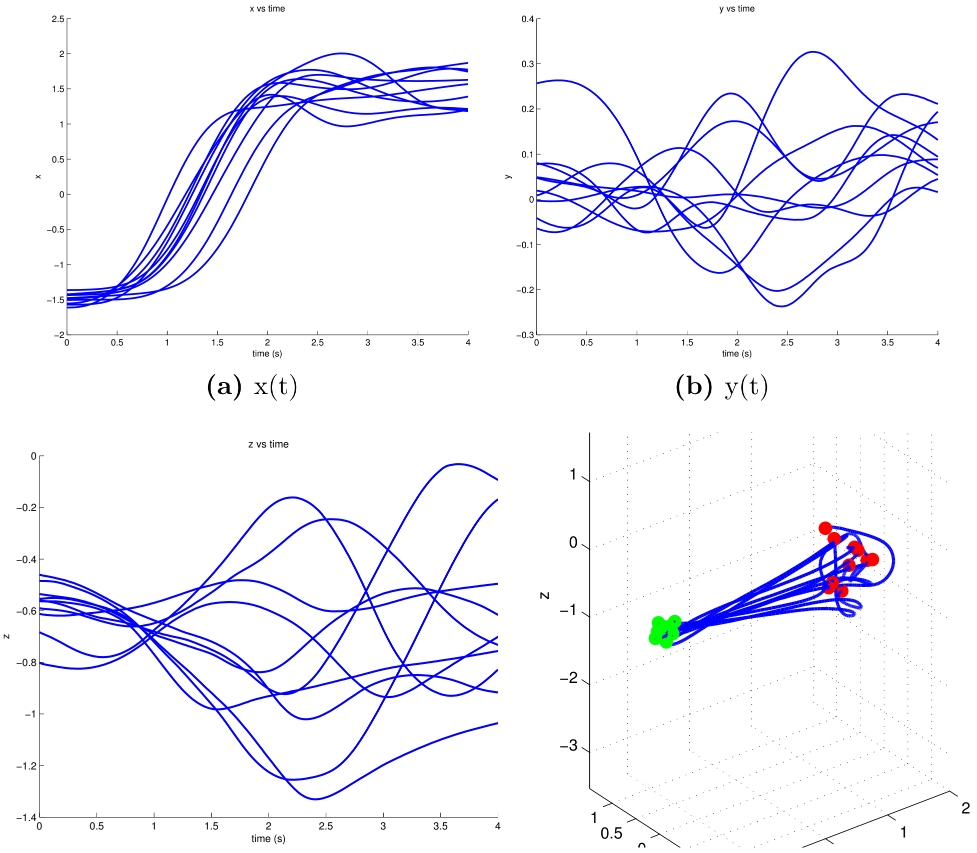
By definition, we can also write:

$$\begin{aligned}\dot{v}_1 &= u_1\\ \dot{v}_2 &= u_2\\ \dot{v}_3 &= u_3\\ \ddot{\theta}_1 &= u_4\end{aligned} \tag{5.14}$$

And the full equations of motion for the quadrotor give us expressions for these translational and angular accelerations. Therefore, because we earlier derived continuous-time, full state and control observed trajectories, we now have a system of equations that is linear in the gains $K_1, \ldots, K_9$. In other words, given full state and control observations and the class of PD feedback control policies described in this section, we can sample the observed trajectories to generate an overdetermined system of equations linear in the unknown gains K_i. We then use least squares to solve for the gains K_i that are most consistent with the observed trajectories. The result of this process is a set of gains that define a feedback control policy that represents the PD controller that is best able to reproduce observed flight behavior given boundary conditions similar to those seen in the original observed trajectories.

5.3 Learning from Demonstration: Experiments

We collected a set of human flight trajectories for the task of rapidly translating 2 meters in the x-axis direction while maintaining altitude. Figure 5.2 shows a subset of these human flight trajectories. From this figure, one can see that the primary translation motion took place in approximately 1 second, and stabilization in hover at the goal location is shown for approximately 2 seconds. During each flight, an Optitrack motion capture system recorded the position of the center of mass of the quadrotor and its orientation (as Euler angles and quaternions) at 50Hz. The pilot controlled the quadrotor via a standard hobby radio transmitter.



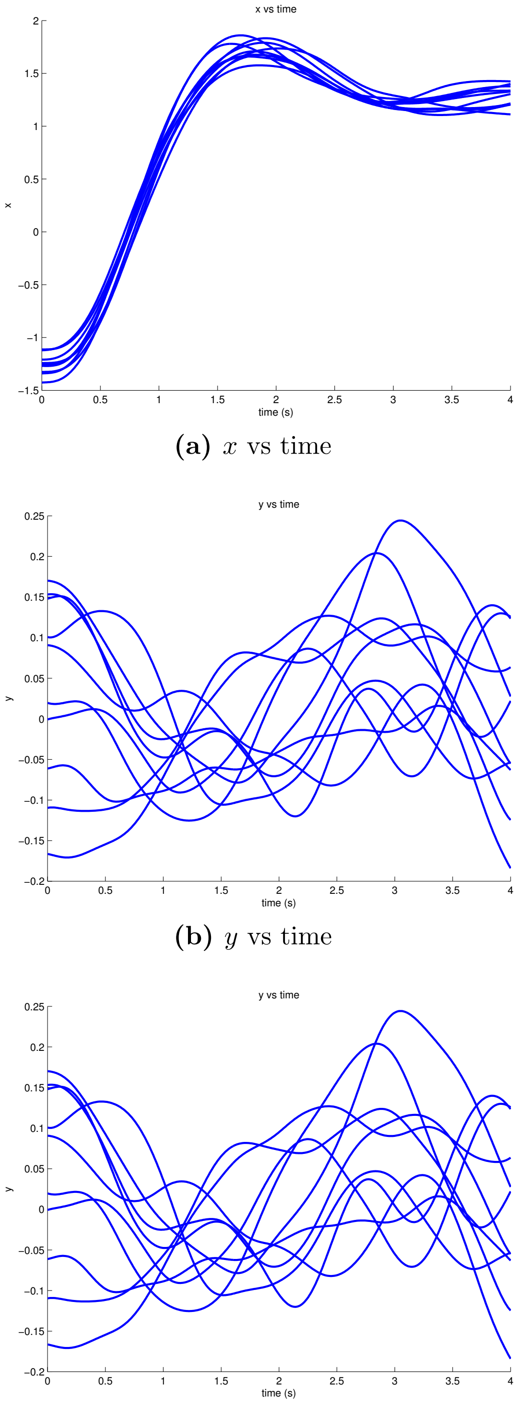
(a) x(t)

(b) y(t)

(c) z(t)

(d) 3D plot

Figure 5.2: Human demonstration flights used as input for our inverse optimal control method. This figure shows that the primary translation maneuver takes place in approximately 1 second, with a repeatable transition to hover at the goal location.



(a) x vs time

(b) y vs time

(c) z vs time

Figure 5.3: This figure shows the flight results of the feedback control policy resulting from our learning from demonstration method.

5.4 Time-Optimal Control: Method

This method uses a reduced two-dimensional model of the quadrotor with three degrees of freedom: the horizontal position x, vertical position z, and pitch angle θ. It is assumed that the angular velocity $\dot{\theta}$ can be controlled directly without dynamics and delay. This assumption is motivated by the fact that quadrotors can achieve high angular accelerations (several hundred rad/s^2) and high-bandwidth angular velocity sensors.

Model

In this section, we define the two-dimensional model of the quadrotor used to compute time-optimal translation maneuvers. We simplify the standard system equations given in equation 5.1 by considering only the following degrees of freedom: the horizontal position q_1, vertical position q_3, and pitch angle θ_2. These degrees of freedom are controlled by total thrust u_4 and pitch rate ω_2, both subject to saturation

$$u_{min} \leq u_4 \leq u_{max} \qquad |\omega_2| \leq \omega_{max} \tag{5.15}$$

For notational simplicity, we will consider the position of the simplified model $q = [q_1, q_3]^T$, the velocity $v = [v_1, v_3]^T$, the orientation $\theta = \theta_2$. We thus consider the reduced state vector $x = [q, v, \theta]$. We define the control vector $u = [\omega_2, u_4]$. The simplified two-dimensional equations of motion can now be written as

$$\begin{aligned} \dot{q} &= v \\ \dot{v} &= \begin{bmatrix} 0 \\ g \end{bmatrix} - \frac{1}{m} \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} u_2 \\ \dot{\theta} &= u_1 \end{aligned} \tag{5.16}$$

with control constraints

$$u \in U = \{u : |u_1| \leq u_{1,max}, u_{2,min} \leq u_2 \leq u_{2,max}\} \tag{5.17}$$

In vector form, we will refer to this system by

$$\dot{x} = f(x, u) \qquad u \in U. \tag{5.18}$$

Minimum Principle for Time-Optimal Maneuvers

We are considering the time-optimal maneuver that brings the quadrotor from a given initial state x_0 to a given final state x_T. Such a trajectory is the solution to the optimal control problem

$$\begin{aligned} \text{minimize} \quad & \int_0^T g(x,u)\,dt = \int_0^T 1 \; dt = T \\ \text{subject to} \quad & \dot{x} = f(x,u) \\ & x(0) = x_0 \\ & x(T) = x_T \\ & u \in U \quad \forall \quad t \in [0,T] \end{aligned} \tag{5.19}$$

This problem can be solved using Pontryagin's minimum principle to provide necessary conditions for optimal control. The Hamiltonian for this problem is

$$\begin{aligned} H(x,u,p) &= g(x,u) + p^T f(x,u) \\ &= 1 + p_1\dot{x} + p_2\dot{z} - p_3\frac{1}{m}\sin\theta u_2 + p_4\left(g - \frac{1}{m}\cos\theta u_2\right) + p_5 u_1 \end{aligned} \tag{5.20}$$

Because the terminal time is free, the Hamiltonian is zero along optimal trajectories $H = 0 \forall t \in [0,T]$, i.e. $H \equiv 0$. Optimal trajectories satisfy the adjoint equations

$$\dot{p} = -\nabla_x H(x^*, u^*, p). \tag{5.21}$$

For this problem, the adjoint equations yield

$$\begin{aligned} \dot{p}_1 &= 0 & p_1 &= c_1 \\ \dot{p}_2 &= 0 & p_2 &= c_2 \\ \dot{p}_3 &= -p_1 & p_3 &= c_3 - c_1 t \\ \dot{p}_4 &= -p_2 & p_4 &= c_4 - c_2 t \end{aligned} \tag{5.22}$$

where $c = (c_1, \ldots, c_4)$ is an unknown parameter vector. The differential equation for the last costate is

$$\dot{p}_5 = p_3\frac{1}{m}\sin\theta u_2 - p_4\frac{1}{m}\cos\theta u_2 \tag{5.23}$$

Now to compute the optimal angular velocity input, u_1^*, we minimize the Hamiltonian over possible inputs

$$u_1^* = \underset{|u_1| \leq u_{1,max}}{\arg\min} \{p_5 u_1\} \tag{5.24}$$

Along regular arcs, where p_5 is nonzero, the optimal control is

$$u_1^* = \pm u_{1,max} \tag{5.25}$$

depending on the sign of p_5. If $p_5 = 0$ over some interval of time, the optimal trajectory is a singular arc. Along this interval $\dot{p}_5$ is also zero and yields the condition

$$\dot{p}_5 = p_3 \frac{1}{m} \cos\theta u_2 - p_4 \frac{1}{m} \sin\theta u_2 = 0 \tag{5.26}$$

Since the thrust u_2 is always greater than zero by definition, we can solve for the pitch along the singular arc

$$\theta^*(t) = \arctan\left(\frac{p_3(t)}{p_4(t)}\right) = \arctan\left(\frac{c_3 - c_1 t}{c_4 - c_2 t}\right) \tag{5.27}$$

From the system dynamics, we have $u_1 = \dot{\theta}$, so we can differentiate the above equation to get

$$u_{1,singular} = \dot{\theta}^* = \frac{c_3 c_2 - c_1 c_4}{(c_1^2 + c_2^2)\, t^2 - 2(c_1 c_3 + c_2 c_4)t + c_3^2 + c_4^2} \tag{5.28}$$

that holds along the singular arc. To summarize, the optimal control u_1^* is given by

$$u_1^* = \begin{cases} u_{1,max} & \text{if } p_5 < 0 \\ u_{1,singular} & \text{if } p_5 = 0 \\ -u_{1,max} & \text{if } p_5 > 0 \end{cases} \tag{5.29}$$

We can similarly solve for the optimal thrust u_2^* by minimizing the Hamiltonian with respect to u_2

$$u_2^* = \underset{u_2 \in U}{\arg\min} \left(-p_3 \frac{1}{m} \sin\theta^* u_2 - p_4 \frac{1}{m} \cos\theta^* u_2\right) \tag{5.30}$$

Define a switching function

$$\Phi = p_3 \sin\theta^* + p_4 \cos\theta^* \tag{5.31}$$

For a singular arc to exist, the switching function must be zero for a finite interval interval of time. Using the solution of u_1^* that determines θ^*, it can be shown that the switching function is not zero for a finite interval of time. Therefore the optimal thrust control u_2^* is given by

$$u_2^* = \begin{cases} u_{2,max} & \text{if } \Phi \leq 0 \\ u_{2,min} & \text{if } \Phi > 0 \end{cases} \tag{5.32}$$

Computing Bang-Bang Optimal Controls

Consider maneuvers with no singular arcs and only bang-bang control behavior. The Zurich method for these maneuvers consists of three steps: (1) switching time optimization (STO) is used to find a bang-bang maneuver from the specified initial conditions to the desired goal position; (2) given the resulting trajectory from the previous step, the parameter vector $c = (c_1, c_2, c_3, c_4)$ is computed such that necessary conditions for optimal control are satisfied; (3) the resulting trajectory and costate trajectory are used as a very good initial guess in a numerical boundary value problem solver to refine the solution; (4) iterative learning control is used to adapt the switching times to handle model inaccuracies between the simplified 2D model and the real hardware system.

(1) Switching Time Optimization: The first step of the method makes an initial guess of (a) the initial control inputs $u(0)$, (b) switching times for the two control inputs $\{t_{1,i}\}$ for $i = 1, \ldots, n_1$ and $\{t_{2,j}\}$ for $j = 1, \ldots, n_2$ and (c) the terminal time t_f. Now, the switching time optimization method attempts to minimize the final state error defined by the following residual function

$$J_{res}\left(\{t_{1,i}\}, \{t_{2,j}\}, t_f\right) = (x(t_f) - x_{goal})^T W (x(t_f) - x_{goal}) \tag{5.33}$$

For a given set of switching times and terminal time, the final state $x(t_f)$ is computed using simulation of the simplified 2D dynamic model acting under open control that is fully specified by the initial control, switching times, and terminal time. Our implementation uses MATLAB's `fmincon` to perform numerical constrained optimization, where the objective is given above and the simulation is performed inside the function's iteration loop.

(2) Costate parameter estimation: After finding a bang-bang trajectory in the previous step, it must be shown that the trajectory satisfies the necessary conditions of optimal control. To do this, the costate parameter vector $c = (c_1, c_2, c_3, c_4)$ is computed and the costates are shown to satisfy the necessary conditions of optimal control. To compute the parameter vector c, a set of overdetermined linear constraint equations is formed and then solved using linear least squares. The first set of equations come from the condition that the costate $p_5(t)$ must be zero at the switching times $t = t_{1,i}$. The condition $H \equiv 0$ is used to form an equation for $p_5(t)$.

Another set of constraint equations comes from satisfying the integral of $\dot{p}_5(t)$ over arbitrary intervals $[a, b] \in [0, t_f]$:

$$p_5(b) - p_5(a) = \int_a^b \dot{p}_5 dt \tag{5.34}$$

where, again, the condition $H \equiv 0$ is used to compute the left hand side.

The final set of constraints come from the thrust control switching curve vanishing at switching times:

$$\Phi(t_{2,i}) = 0 \quad \text{for } i = 1, \ldots, n_2 \tag{5.35}$$

These constraints all yield expressions that are linear in the parameter vector c. Thus, the equations can be written in the form:

$$Ac = r \tag{5.36}$$

that is an overdetermined system of equations and has the least squares solution

$$c^* = (A^T A)^{-1} A^T r \tag{5.37}$$

If the resulting c^* yields a small residual, i.e. $Ac^* - r \approx 0$, then the necessary conditions for optimal control are approximately satisfied.

(3) Refinement via Numerical Solver: In our implementation we use the numerical solver GPOPS2 to refine the solution given the state and costate trajectories determined above as initialization. GPOPS2 is a general-purpose

tool for solving nonlinear optimal control problems using an *hp*-adaptive Radau pseudospectral Gaussian quadrature method where collocation is performed at the Legendre-Gauss-Radau quadrature points. GPOPS2 is able to use SNOPT and IPOPT compiled libraries for solving nonlinear programs.

(4) Adapting for Hardware via Iterative Learning Control: In this step, we develop the iterative learning control method outlined in [123]. This method numerically computes a Jacobian that describes small changes in the final state error due to small changes in the set of switching times and terminal time. Let P refer to the parameter vector composed of switching times singular arc durations and terminal time. Let the function $F(P)$ denote the final state error resulting from simulation under the control policy defined by parameter P. Consider the switching times and terminal time resulting from the previous switching time optimization and refinement in simulation denoted by P_0. The Taylor expansion of $F(P)$ around P_0 is given by

$$F(P_0 + P) = F(P_0) + \frac{\partial F}{\partial P} P = 0 + JP \tag{5.38}$$

where $F(P_0) = 0$ by definition. The Jacobian J is computed numerically by performing multiple simulations for changes in a single component of P at a time:

$$J_{ij} = \frac{F_i(P_0 + h_j e_j) - F(P_0)}{h_j} \tag{5.39}$$

where h_j is a small increment, e_j is a unit vector in the j-th coordinate direction, and again $F(P_0) = 0$. Once J is computed, we use it to update the parameter vector P in iterated hardware flights as follows:

$$P_{i+1} = P_i - \gamma J^{-1} E_i \tag{5.40}$$

where E_i is the terminal state error achieved in the i-th hardware flight iteration.

5.5 Time-optimal Control: Experiments

We implemented the method of [122, 123] in hardware using an Ascending Technologies hummingbird and an Optitrack motion capture system for position tracking. As outlined above, the method begins by off-line computing a nominal trajectory that satisfies the minimum principle for the time-optimal control problem. Then, in repeated flights, the iterative learning control algorithm adapts the switching times and total duration of the flight to reduce terminal state error. Figure 5.4 shows the resulting trajectory upon the first iteration of flight. The task is to perform a 2 meter translation along the x-axis while maintaining altitude and minimizing time of flight. Figure 5.4 exhibits approximately half meter final position error in the nominal flight time. Figure 5.5 shows the resulting trajectory after 20 iterations of the iterative learning control algorithm. Figure 5.6 shows the norm of the terminal state error over iterations of the algorithm. Iterative learning control is able to provide a benefit in this task because the simplified 2D model did not take into consideration the rotational inertia and dynamics of the real system, and instead directly commanded body angular velocity. Thus, directly applying the original nominal open loop control policy on the full system introduces lag that can be compensated for by adjusting the switching time of the bang-singular control policy.

In comparison with our learning from demonstration approach, one can see that the ETH Zürich method produces a slightly shorter duration flight. That is, they do a better job of solving the minimum-time optimal control problem given the constraints of the vehicle. This is, in fact, a major motivation for their work: to produce trajectories at the limits of the vehicles capabilities. However, the difference in total flight time between the ETH Zürich method and our learning approach is approximate 0.25 seconds, or 20% of the total flight time. Also, the result of our learning method is a very simple and general feedback policy that is robust to direction of motion and distance traveled, as well as robust to environmental disturbances such as slight gusts. The open loop policy of ETH Zürich works well in the nominal environment, but does not generalize to any other tasks than the one it was designed for. A simple demonstration of this behavior is to consider an experiment in which point to point trajectories were flown repeatedly to the same desired goal position starting from a distribution of initial conditions. Table 5.1 shows results comparing the outcome of flights executed under the control policy resulting from the minimum-time optimal control approach of ETH Zürich and the control policy resulting from our learning from demonstration and policy search method. Another point of comparison is computational complexity and manual parameter tuning. Qualitatively,

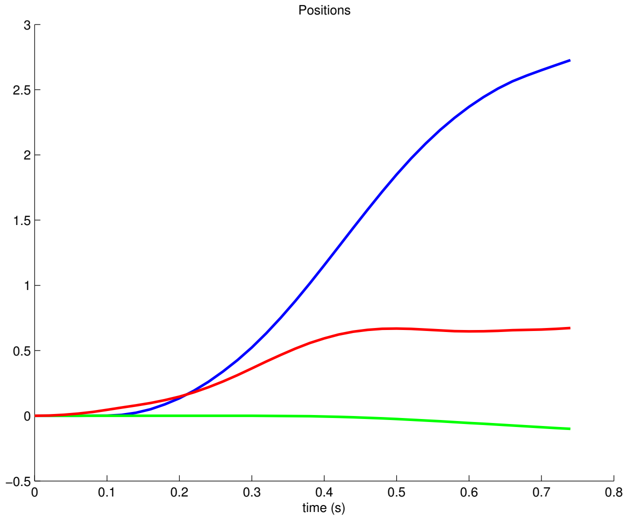


Figure 5.4: This figure shows the evolution of the position of the real quadrotor at the first iteration of Iterative Learning Control, i.e. applying the nominal control obtained from the switching time optimization. This trajectory exhibits large terminal error with respect to the goal state of the maneuver (2 meter translation along the x-axis).

Table 5.1: Comparison of time-optimal control and new method of inverse optimal control

System	Measurement	Time-optimal	New
Aggressive Flight	x_0 error	0.5	0.5
	x_f error	0.49	0.21

the ETH Zürich implementation requires a larger number of different optimizations and requires tuning parameters that affect the performance of the algorithm.

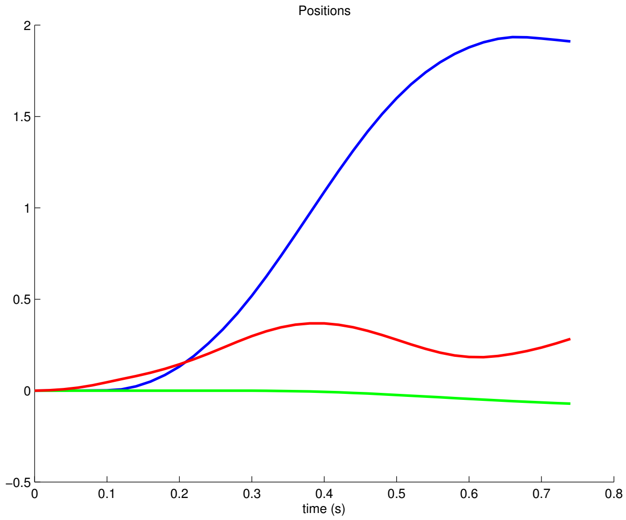


Figure 5.5: This figure shows the evolution of the position of the hardware quadrotor at iteration 20 of the Iterative Learning Control algorithm, i.e. applying the nominal control obtained from the open loop policy using modified switching times and terminal time. This trajectory exhibits much better performance in terms of matching the desired goal state of the maneuver (2 meter translation along the x-axis).

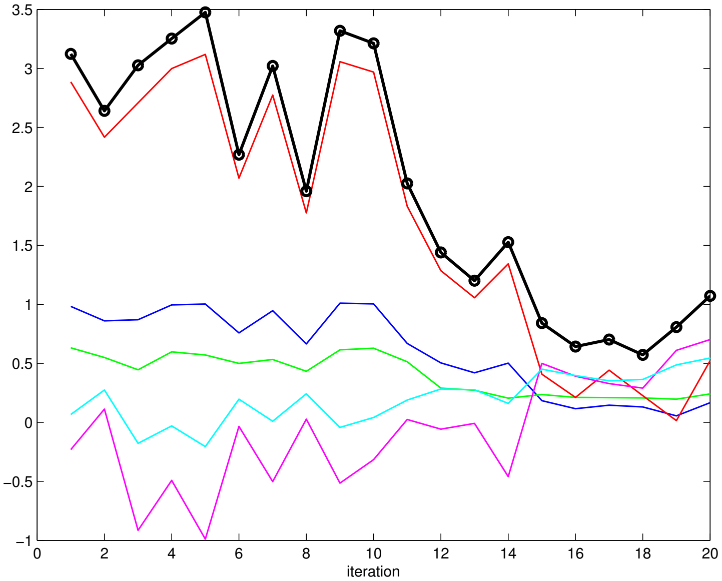


Figure 5.6: This figure shows the evolution of the terminal error at each iteration of the Iterative Learning Control algorithm. This behavior is sensitive to a variety of tuning parameters, including the step size η in the parameter update rule.

Chapter 6

Conclusion and Future Work

In this thesis, we presented a new method of inverse optimal control, and compared the method to three existing methods using a set of example systems designed to yield insight about the differences between the approaches. The classical problem of inverse optimal control is to infer the class of objective functions that make a given control policy optimal. In recent work, it is assumed that the underlying control policy is unknown. In this case the objective function is inferred from observations of trajectories of the system. The existing solution approaches to this problem search for values of the parameters that minimize the difference between predicted and observed trajectories. These approaches require solving a forward optimal control problem at each iteration. The approach presented in this chapter minimizes residual functions derived from first order necessary conditions for optimality. We compared our new approach with the following methods: inverse reinforcement learning by Abbeel and Ng [2], maximum margin planning by Ratliff, et al. [3], and inverse optimal control by Mombaur, et al. [1]. We demonstrate the performance of these methods by performing simulation experiments in which cost function parameters are inferred given optimal state-input trajectories of the system. We test the robustness of the methods by perturbing the true cost function – in other words, by considering the parameterized structure of the cost function as an approximation to the true cost function. Our results show that the new method we develop is better able to recover unknown parameters and is less computationally expensive than the existing methods. We then apply our method to three problems of interest in robotics. First, we apply inverse optimal control to learn the physical properties of an elastic rod. Second, we apply inverse optimal control to learn models of human walking paths given observations of people performing goal oriented walking. These models of human locomotion enable automation of mobile robots moving in a shared space with humans, and enable motion prediction of walking humans given partial trajectory observations. Finally, we apply inverse optimal control to develop a method of learning from demonstration for quadrotor dynamic maneuvering. We develop a new method and compare our learning-based method with a numerical method to an analogous minimum-time optimal control problem, developed by the Flying Machine Arena at ETH Zürich.

There are many exciting opportunities and avenues for future work. One problem of interest is to use inverse optimal control to transfer behavior that is possible given one system (sensors and actuators) to another potentially completely different system (set of sensors and actuators). For example, quadrotor flight tasks are quite simple given a motion capture system such as Optitrack or Vicon. However, if the quadrotor is fitted with an onboard monocular or RGB-D camera, it is not immediately clear what algorithm one should use to automate a similar flight task. For example, the literature on monocular visual servo control is expansive and diverse, with seemingly special tuning and algorithm modification presented in each publication. Can principles from inverse optimal control and reinforcement learning provide a convenient way to transfer control algorithms that are simple given motion capture sensors to a system that only uses monocular or RGB-D onboard cameras and control? A central idea is that one gets to observe trajectory features from both sets of sensors during learning. Another avenue of future work lies in taking advantage of systems with differential flatness to simplify or alter the performance of the inverse optimal control algorithm.

Appendix A

Quadrotor Dynamics and Control

We use the "modeling for control" approach that is common in recent literature [124–127]. We now derive nonlinear differential equations of motion for the quadrotor system

$$\dot{x}(t) = f\left(x(t), u(t), t, p\right)$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control vector, and $p \in \mathbb{R}^p$ is a parameter vector containing constants such as mass, moments of inertia, etc. Sometimes we will leave off the parameter vector and consider it an implicit component of the system. The state vector is defined as

$$x = (x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r)^T$$

where x, y, z represent the position of the center of mass in the world frame, v_x, v_y, v_z represent the velocity of the center of mass in the world frame, ϕ, θ, ψ represent ZYX Euler angles specifying the attitude of the vehicle, and p, q, r represent the angular velocity in the body frame. The control vector we will consider consists of the applied thrust and torques on the rigid body

$$u = (u_1, u_2, u_3, u_4)$$

where u_1, u_2, u_3 are roll, pitch, and yaw torques on the rigid body and u_4 is the overall thrust force applied by the rotors. We define two coordinate frames, the world frame that we will call frame 0, and the body-fixed frame that we will call frame 1. Unit vectors representing the world frame are denoted by $(\hat{I}, \hat{J}, \hat{K})$, and unit vectors representing the body frame are $(\hat{\imath}, \hat{\jmath}, \hat{k})$. We begin with the Newton-Euler equations to describe the translational and rotational dynamics of the rigid body quadrotor (in the body frame)

$$\begin{pmatrix} F \\ M \end{pmatrix} = \begin{pmatrix} mI & 0 \\ 0 & J \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \dot{\omega} \end{pmatrix} + \begin{pmatrix} 0 \\ \omega \times J\omega \end{pmatrix} \tag{A.1}$$

where m is the mass of the body, F is the total force acting on the center of mass (in the body frame), M is the total moment acting about the center of mass (in the body frame), J is the moment of inertia about the center of mass, I is a 3×3 identity matrix, $\ddot{q}$ is the acceleration of the center of mass (in the body frame), and ω is the angular velocity of the body (in the body frame). We assume the body frame is aligned with the principle axes, and that the moment of inertia J is diagonal. Now, we parameterize the rotational state of the quadrotor using roll, pitch, and yaw Euler angles, ϕ, θ, ψ (using the ZYX system). We denote the rotation matrix that rotates the body frame to the world frame as R_1^0.

We model the rotor induced forces and moments as functions of the rotor angular velocities, σ_i

$$F_i = -k_{F_i} \sigma_i^2 \hat{k} \tag{A.2}$$

$$M_1 = -k_{M_1} \sigma_1^2 \hat{k} \qquad M_2 = k_{M_2} \sigma_2^2 \hat{k} \tag{A.3}$$

$$M_3 = -k_{M_3} \sigma_3^2 \hat{k} \qquad M_4 = k_{M_4} \sigma_4^2 \hat{k} \tag{A.4}$$

where the moments act in the $\hat{k}$ direction, for $i = 1, 2, 3, 4$. In vector form, let $\omega_r = (\omega_1, \ldots, \omega_4)$. Note that rotors 1 and 3 rotate in the $-z_B$ direction, while rotors 2 and 4 rotate in the $+z_B$ direction. The rotors are built such that F_i act in the z_B direction, while M_1 and M_3 act in the z_B direction and M_2 and M_4 act in the $-z_B$ direction.

The input vector u representing total control torques and thrust force acting on the vehicle is related to the individual motor thrust forces F_i and

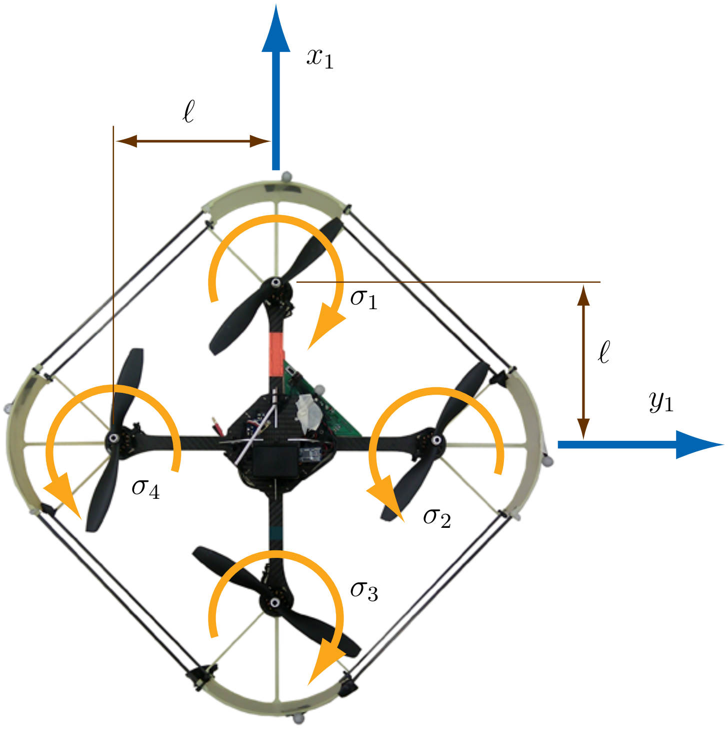


Figure A.1: An Ascending Technologies Hummingbird with two counter-clockwise and two clockwise rotors.

aerodynamic torques M_i

$$\begin{aligned}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} &= \begin{bmatrix} 0 & -lk_F & 0 & lk_F \\ lk_F & 0 & -lk_F & 0 \\ -k_M & k_M & -k_M & k_M \\ k_F & k_F & k_F & k_F \end{bmatrix} \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_3^2 \\ \sigma_4^2 \end{bmatrix} \\ &= W \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_3^2 \\ \sigma_4^2 \end{bmatrix}\end{aligned}$$

where l is the distance from the motor to the center of mass and σ_i is the angular velocity of the i-th motor.

The Euler angle rates are related to body angular velocities through the matrix $S(\phi, \theta, \psi)$ defined by

$$S = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\cos\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{A.5}$$

Now, we are ready to write down our final equations of motion

$$\begin{aligned}
\begin{matrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{matrix} &= \begin{matrix}v_x\\ v_y\\ v_z\end{matrix} \\
\begin{matrix}\dot{v}_x\\ \dot{v}_y\\ \dot{v}_z\end{matrix} &= \begin{bmatrix}0\\0\\g\end{bmatrix} + \frac{1}{m} R_1^0 \begin{bmatrix}0\\0\\-u_4\end{bmatrix} \\
\begin{matrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{matrix} &= S\omega \\
\dot{\omega} &= J^{-1}\left(\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix} - \omega \times J\omega\right)
\end{aligned} \tag{A.6}$$

We gather the parameters in this model, and call this the *parameter vector*

$$p = (m, J, l, k_{F_1}, \ldots, k_{F_4}, k_{M_1}, \ldots, k_{M_4}) \tag{A.7}$$

We can now write the equations of motion in vector format

$$\dot{x}(t) = f\left(x(t), u(t), p\right)$$

Nominal values of parameters can be extracted using first principles and experimental data. We use the following as nominal, or “true” values of the parameter vector:

$$\begin{aligned} m &= 0.7\text{kg} \\ J &= \operatorname{diag}\left(J_{xx}, J_{yy}, J_{zz}\right) = \operatorname{diag}\left(0.004, 0.004, 0.008\right) \\ L &= 0.17\text{m} \\ k_{F_1} &= \cdots = k_{F_4} = 6.7e-6\frac{N}{rad/sec^2} \\ k_{M_1} &= \cdots = k_{M_4} = 1.7e-7\frac{Nm}{rad/sec^2} \end{aligned} \tag{A.8}$$

A.1 Quadrotor Controller

In our analysis, the system state and input vectors are:

$$x = (x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r) \tag{A.9}$$

$$u = (u_1, u_2, u_3, u_4) \tag{A.10}$$

where u_1,u_2,u_3,u_4 are roll, pitch moments and thrust repectively. The control architecture uses an inner loop to control the roll, pitch and yaw of the vehicle and runs at approximately 1kHz. An outer loop computes desired angles based on world position and velocity tracking error.

A.1.1 Outer Loop: Position Control

The outer loop computes desired angles based on world frame position and velocity tracking error. The first step is to compute desired translational accelerations in the world frame. These desired accelerations are then mapped to a desired vehicle attitude (roll, pitch, and yaw) that is then sent to the inner control loop as a setpoint for regulation. Desired accelerations are computed from an outer loop LQR based on the position (and velocity) errors as shown:

$$a = K_{3\times 6} \begin{bmatrix} x - x^* \\ y - y^* \\ z - z^* \\ v_x - v_x^* \\ v_y - v_y^* \\ v_z - v_z^* \end{bmatrix} \tag{A.11}$$

where $a = (a_x, a_y, a_z)$ are the desired accelerations and $x^*, y^*, z^*, v_x^*, v_y^*, v_z^*$ are the desired vehicle positions and velocities. The LQR controller is derived based on the simplified system equations:

$$m\ddot{x} = u \tag{A.12}$$

where here x is simply 3D position in the world, and u is a 3D control vector.

Around hover conditions, ϕ^*, $\theta^* = 0$, $\psi^* = a \neq 0$, and $u_4 = mg$, we linearize the translation portion of the equations of motion $A.6$ to get a relationship between the desired accelerations a and roll and pitch angles ϕ and θ. The equations we will linearize are re-written here

$$\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} R_1^0 \begin{bmatrix} 0 \\ 0 \\ u_4 \end{bmatrix} \tag{A.13}$$

After expanding the rotation matrix, we obtain

$$m\mathbf{a} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \begin{bmatrix} \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\ cos\phi \sin\theta \sin\psi - \cos\psi \sin\phi \\ cos\theta \cos\phi \end{bmatrix} u_4 \tag{A.14}$$

We use the following trigonometry identities during linearization

$$\cos(\epsilon + \delta\epsilon) = \cos\epsilon - \delta\epsilon \sin\epsilon$$

$$\sin(\epsilon + \delta\epsilon) = \sin\epsilon + \delta\epsilon \cos\epsilon$$

The linearized result is

$$\begin{bmatrix} \delta\phi \\ \delta\theta \end{bmatrix} = \begin{bmatrix} -\cos\psi & -\sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \frac{1}{g} \tag{A.15}$$

and

$$\delta u_4 = m a_z \tag{A.16}$$

Thus, the information that is sent to the inner loop is given by

$$\begin{bmatrix} \phi^* \\ \theta^* \\ \psi^* \\ u_4 \end{bmatrix} = \begin{bmatrix} \delta\phi \\ \delta\theta \\ a \\ mg - \delta u_4 \end{bmatrix} \tag{A.17}$$

A.1.2 Inner Loop: Attitude Control

In the inner loop, three independent LQR loops are used for roll, pitch and yaw controls around the nominal hover state, $\dot{\phi} \approx p$, $\dot{\theta} \approx q$ and $\dot{\psi} \approx r$:

$$u_1 = -\,K \begin{bmatrix} \phi - \phi^* \\ p - p^* \end{bmatrix} \tag{A.18}$$

$$u_2 = -\,K \begin{bmatrix} \theta - \theta^* \\ q - q^* \end{bmatrix} \tag{A.19}$$

$$u_3 = -\,K \begin{bmatrix} \psi - \psi^* \\ r - r^* \end{bmatrix} \tag{A.20}$$

where inputs u_1, u_2 and u_3 are the roll, pitch and yaw torques respectively. The attitude controller is used to track trajectories in SO(3) that are close to the nominal hover state where the roll and pitch angles are small. Note that in hardware, we must then obtain the vector of desired rotor speeds is from the control vector u

$$\begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_3^2 \\ \sigma_4^2 \end{bmatrix} = W^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{A.21}$$

The motor speeds are then mapped to a hardware command vector using a mapping obtained from a motor-propeller calibration

$$\begin{bmatrix} cmd_1 \\ cmd_2 \\ cmd_3 \\ cmd_4 \end{bmatrix} = \mathbf{f} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \end{bmatrix} \tag{A.22}$$

Appendix B

Generalized LQR Solution

Consider the following optimal control problem

$$\begin{aligned} \min_{z(t),v(t)} \quad & \int_0^{t_f} \|F(z,v,t)z(t) + G(z,v,t)v(t) + h(z,v,t)\|^2 dt \\ \text{s.t.} \quad & \dot{z} = A(t)z(t) + B(t)v(t) \\ & z(0) = z_0 \end{aligned} \tag{B.1}$$

We will begin by considering a similar discrete time problem

$$\begin{aligned} \min_{z(0)\ldots,v(0)\ldots} \quad & \sum_{k=0}^{N-1} \|F(z(k),v(k),k)z(k) + G(z(k),v(k),k)v(k) + h(z(k),v(k),k)\|^2 \\ \text{s.t.} \quad & z(k+1) = A(k)z(k) + B(k)v(k) \\ & z(0) = z_0 \end{aligned} \tag{B.2}$$

B.1 Solution: discrete-time dynamic programming

This derivation of the LQR solution via dynamic programming is inspired by [134]. Write the recursive form of the value function as

$$V(z(k)) = \min_{v(k)} \left\{ \|Fz(k) + Gv(k) + h\|^2 + V(z(k+1)) \right\} \tag{B.3}$$

Now, assume the value function has the form

$$V(z(k) = z^T(k)P(k)z(k)) + 2d^T(k)z(k) + c(k) \tag{B.4}$$

where $P(k) = P^T(k) \geq 0$. Now, we can solve the minimization over $v(k)$ by taking the derivative of the value function with respect to $v(k)$ and setting it to zero

$$0 = v^T(G(k)^T G(k) + B(k)^T P(k+1) B(k)) + z(k)^T F(k)^T G(k) + z(k)^T A(k)^T P(k+1) B(k) \\ + h(k)^T G(k) + d(k+1)^T B(k) \quad \text{(B.5)}$$

Thus we have the optimal control

$$v^*(k) = -(G(k)^T G(k) + B(k)^T P(k+1) B(k))^{-1} \left\{ (G(k)^T F(k) + B(k)^T P(k+1) A(k)) z(k) \\ + G(k)^T h(k) + B(k)^T d(k+1) \right\} \quad \text{(B.6)}$$

We can rewrite this as

$$v^*(k) = K(k) z(k) + q(k) \quad \text{(B.7)}$$

where

$$\begin{aligned} K(k) &= -(G(k)^T G(k) + B(k)^T P(k+1) B(k))^{-1} \left(G(k)^T F(k) + B(k)^T P(k+1) A(k) \right) \\ q(k) &= -(G(k)^T G(k) + B(k)^T P(k+1) B(k))^{-1} \left(G(k)^T h(k) + B(k)^T d(k+1) \right) \end{aligned} \quad \text{(B.8)}$$

Plugging this expression for $v^*(k)$ back into the recursive value function, and equating like terms, we derive the recursive Riccati equations

$$P(k) = F(k)^T F(k) + K(k)^T G(k)^T G(k) K(k) + 2F(k)^T G(k) K(k) \\ + A(k)^T P(k+1) A(k) + K(k)^T B(k)^T P(k+1) B(k) K(k) + 2A(k)^T B(k) K(k) \quad \text{(B.9)}$$

$$d(k) = K(k)^T G(k)^T G(k) q(k) + F(k)^T G(k) q(k) + F(k)^T h(k) \\ + K(k)^T G(k)^T h(k) + K(k)^T B(k)^T P(k+1) B(k) q(k) + A(k)^T B(k) q(k) \quad \text{(B.10)}$$

$$c(k) = h(k)^T h(k) + q(k)^T G(k)^T G(k) q(k) + 2q(k)^T G(k)^T h(k) \\ + q(k)^T B(k)^T P(k+1) B(k) q(k) + c(k+1) \quad \text{(B.11)}$$

with terminal conditions $P(N) = 0$, $d(N) = 0$, $c(N) = 0$.

B.2 Solution: continuous time HJB

This derivation follows the general outline of [17,135]. Similar to the discrete time case, assume the value function has the following form

$$V(z(t)) = z(t)^T P(t)z(t) + 2d(t)^T z(t) + c(t) \tag{B.12}$$

The Hamilton-Jacobi-Bellman (HJB) equation is given by

$$-\frac{\partial V}{\partial t} = \min_v \left\{ L(z, v, t) + \frac{\partial V}{\partial z} f(z, v, t) \right\} \tag{B.13}$$

where

$$\frac{\partial V}{\partial t} = z(t)^T \dot{P}(t)z(t) + 2\dot{d}(t)^T z + \dot{c}(t) \tag{B.14}$$

and

$$\frac{\partial V}{\partial z} = 2z(t)^T P(t) + 2d(t)^T \tag{B.15}$$

$$L(z, v, t) = \|F(t)z(t) + G(t)v(t) + h(t)\|^2 \tag{B.16}$$

Taking the minimization over $v(t)$ on the right hand side, we find the optimal control

$$v^*(t) = -(G(t)^T G(t))^{-1} \left\{ (G(t)^T F(t) + B(t)^T P(t))z(t) + G(t)^T h(t) + B(t)^T d(t) \right\} \tag{B.17}$$

We can rewrite the optimal control as

$$v^*(t) = K(t)z(t) + q(t) \tag{B.18}$$

where

$$\begin{aligned} K(t) &= -(G(t)^T G(t))^{-1} \left(G(t)^T F(t) + B(t)^T P(t) \right) \\ q(t) &= -(G(t)^T G(t))^{-1} \left(G(t)^T h(t) + B(t)^T d(t) \right) \end{aligned} \tag{B.19}$$

Now, plugging this back into the HJB equation, and equating like terms, we obtain

$$\begin{aligned}-\dot{P}(t) &= F(t)^T F(t)+K(t)^T G(t)^T G(t)K(t)+F(t)^T G(t)K(t)+K(t)^T G(t)^T F(t)\\ &\quad + P(t)A(t) + A(t)^T P(t) + P(t)B(t)K(t) + K(t)^T B(t)^T P(t) \quad \text{(B.20)}\end{aligned}$$

$$\begin{aligned}-\dot{d}(t) &= K(t)^T G(t)^T G(t)q(t)+F(t)^T G(t)q(t)+F(t)^T h(t)+K(t)^T G^T(t)h(t)\\ &\quad + P(t)B(t)q(t) + A(t)^T d(t) + K(t)^T B(t)^T d(t) \quad \text{(B.21)}\end{aligned}$$

$$-\dot{c}(t) = h(t)^T h(t) + q(t)^T G(t)^T G(t)q(t) + 2q(t)^T G(t)^T h(t) + 2d(t)^T B(t)q(t) \quad \text{(B.22)}$$

with terminal conditions $P(t_f) = 0$, $d(t_f) = 0$, $c(t_f) = 0$.

B.3 Existence and uniqueness

In this section, we follow the development of necessary and sufficient conditions for optimality given in [63]. Define the Hamiltonian as

$$H(z(t), p(t), v(t)) = \|F(t)z(t)+G(t)v(t)+h(t)\|^2+p(t)^T (A(t)z(t)+B(t)v(t)) \quad \text{(B.23)}$$

where $p(t)$ is the costate of the system. We will make the following assumptions:

$$G^T(t)G > 0 \quad \text{(B.24)}$$

and

$$F(t)^T F(t) - F(t)^T G(t) \left(G(t)^T G(t)\right)^{-1} G(t)^T F(t) \geq 0 \quad \text{(B.25)}$$

This is similar to the regular LQR assumption that R is positive definite and Q is positive semidefinite.

Theorem 6. *Let $v^*(t)$ be an admissible control, and $z^*(t)$ be the trajectory corresponding to v^*, originating at $z(0) = z_0$. In order that v^* be optimal for the cost functional* (B.1)*, it is necessary that there exist a function $p^*(t)$ such that:*

1. *$z^*(t)$ and $p^*(t)$ are solutions to the canonical system*

$$\dot{z}^*(t) = D_p H(z^*, p^*, u^*) \tag{B.26}$$

$$\dot{p}^*(t) = -D_z H(z^*, p^*, u^*) \tag{B.27}$$

with boundary conditions

$$z^*(0) = z_0 \qquad z^*(t_f) \textit{ free} \tag{B.28}$$

.

2. *The Hamiltonian has a minimum as a function of v at $v = v^*(t)$ for $t \in [0, t_f]$*

$$\min_v H(z^*, p^*, v) = H(z^*, p^*, v^*) \tag{B.29}$$

3. *The costate has boundary condition*

$$p^*(t_f) = 0 \tag{B.30}$$

4. *The Hamiltonian is constant along the optimal trajectory*

$$H^*(t) = H^*(t_f) = \textit{ const} \tag{B.31}$$

Proof. This is the minimum principle for free endpoint, fixed final time, optimal control. The proof can be approached from the HJB equations, or from variational calculus.

Now, consider the following form of solution for the costate

$$p(t) = P(t)z(t) + d(t) \tag{B.32}$$

where $P(t)$ is the solution of a Riccati differential equation. Then, using the second necessary condition, and the fact that $H(z, p, v)$ is smooth, we take the gradient of H with respect to v and set it equal to zero. This leads us to:

$$v^*(t) = -(G(t)^T G(t))^{-1} \left\{(G(t)^T F(t) + B(t)^T P(t))z(t) + G(t)^T h(t) + B(t)^T d(t)\right\} \tag{B.33}$$

We can plug this back into the canonical equations to get the reduced canonical equations:

$$\begin{bmatrix} \dot{z}(t) \\ \dot{p}(t) \end{bmatrix} = M \begin{bmatrix} z(t) \\ p(t) \end{bmatrix} \tag{B.34}$$

Differentiating the supposed form for $p(t)$, and then using the reduced canonical equations, we come to the Riccati differential equations:

$$\begin{aligned} -\dot{P}(t) = F(t)^T F(t) + K(t)^T G(t)^T G(t) K(t) + F(t)^T G(t) K(t) + K(t)^T G(t)^T F(t) \\ + P(t)A(t) + A(t)^T P(t) + P(t)B(t)K(t) + K(t)^T B(t)^T P(t) \end{aligned} \tag{B.35}$$

$$\begin{aligned} -\dot{d}(t) = K(t)^T G(t)^T G(t) q(t) + F(t)^T G(t) q(t) + F(t)^T h(t) + K(t)^T G^T(t) h(t) \\ + P(t)B(t)q(t) + A(t)^T d(t) + K(t)^T B(t)^T d(t) \end{aligned} \tag{B.36}$$

$$-\dot{c}(t) = h(t)^T h(t) + q(t)^T G(t)^T G(t) q(t) + 2q(t)^T G(t)^T h(t) + 2d(t)^T B(t) q(t) \tag{B.37}$$

Now, using the terminal condition $p(t_f) = 0$, we can find the boundary conditions for the Riccati equations: $P(t_f) = 0$, $d(t_f) = 0$, $c(t_f) = 0$.

Existence and uniqueness of the costate solution follows from existence and uniqueness of the Riccati equation above. The Riccati equation is an initial value problem.

Sufficient conditions for optimality are as follows:

$$\frac{\partial^2 H}{\partial v(t)^2} > 0 \tag{B.38}$$

and

$$\begin{bmatrix} \frac{\partial^2 H}{\partial z^2} & \frac{\partial}{\partial v}\left(\frac{\partial H}{\partial z}\right) \\ \frac{\partial}{\partial z}\left(\frac{\partial H}{\partial v}\right) & \frac{\partial^2 H}{\partial v^2} \end{bmatrix} \geq 0 \tag{B.39}$$

In our problem, these conditions become:

$$\frac{\partial^2 H}{\partial v(t)^2} = G(t)^T G(t) > 0 \tag{B.40}$$

which holds from our assumptions. The second condition is

$$\begin{bmatrix} \frac{\partial^2 H}{\partial z^2} & \frac{\partial}{\partial v}\left(\frac{\partial H}{\partial z}\right) \\ \frac{\partial}{\partial z}\left(\frac{\partial H}{\partial v}\right) & \frac{\partial^2 H}{\partial v^2} \end{bmatrix} = \begin{bmatrix} F(t)^T F(t) & F(t)^T G(t) \\ G(t)^T F(t) & G(t)^T G(t) \end{bmatrix} \geq 0 \tag{B.41}$$

Because the upper left block is positive semidefinite and lower right block is positive definite, and the diagonal blocks are transposes of each other, this matrix is also positive semidefinite. Thus, the optimal control given above exists and is unique, and minimizes the objective function.

Lemma 7. *If $v(t) \neq 0$ then the cost J must be positive.*

Proof. This follows from the assumptions that $G^T G > 0$.

Lemma 8. *The Riccati solution $P(t)$ along normal extremal trajectories is positive semidefinite for all t.*

Proof. The proof is given in Wonham [136] and Dieci and Eirola [137]. We reproduce the outline here. The standard form of the Riccati equation is

$$\dot{X}(t) = -\bar{A}(t)X(t) - X(t)\bar{A}^T(t) + X(t)\bar{B}(t)X(t) - \bar{C}(t) \tag{B.42}$$

with symmetric positive semidefinite boundary condition $X(t_f) = X_f$. It is assumed that $\bar{B}(t)$ and $\bar{C}(t)$ are symmetric and positive semidefinite. The cited papers prove that the solution of (B.42) is symmetric and positive semidefinite for all $t \geq 0$. Further, if $X(s)$ or $\bar{C}(s)$ is positive for some $s \geq 0$, then $X(t)$ is positive for all $t > s$.

We can rearrange our Riccati equation for $P(t)$ such that it is equivalent to this standard problem by gathering matrices. After rearranging, we have:

$$\bar{A}(t) \leftarrow A(t) - B(t)\left(G(t)^T G(t)\right)^{-1} G^T(t)F(t) \tag{B.43}$$

$$\bar{B}(t) \leftarrow B(t)(G^T(t)G(t))^{-1}B^T(t) \tag{B.44}$$

$$\bar{C}(t) \leftarrow F^T(t)F(t) - F^T(t)G(t)(G^T(t)G(t))^{-1}G^T(t)F(t) \tag{B.45}$$

Note that this is the same form as the standard LQR with cross terms derivation, where

$$Q(t) = F^T(t)F(t) \quad R(t) = G^T(t)G(t) \quad S(t) = F^T(t)G(t) \tag{B.46}$$

These equations motivate the assumptions we made in the LQR development above.

Appendix C

Differential Dynamic Programming

C.1 Introduction

Differential dynamic programming is an iterative numerical algorithm that finds a locally-optimal trajectory given an initial nominal trajectory and control policy. Differential dynamic programming is a well-used method, sometimes appearing with variation under the title of iterative-LQR or Gauss-Newton LQR, and has much in common with a variety of first and second order gradient algorithms [64, 138]. Jacobson and Mayne (1970) [139] is the dominant reference in recent related literature, and we will explore the algorithm given there. We are motivated to understand this algorithm because it solves interesting problems in biological control and reinforcement learning [52, 140–143]. Specifically, we will use DDP in a reinforcement learning scheme that will teach a fixed-wing UAV to perform path-tracking as well as aerobatic maneuvers.

C.2 Algorithm

In this paper, we will consider systems described by nonlinear difference equations. While continuous-time systems have more extensive global minima results, we focus on discrete-time systems that approximate the continuous problem. Such systems are also natural in the context of stochastic control problems that we are interested in exploring in future studies.

The system we will consider is defined by

$$x_{i+1} = f_i(x_i, u_i) \qquad i = 0, 1, \ldots, N-1 \tag{C.1}$$

$$x_0 = \bar{x}_0 \tag{C.2}$$

The sequence $\{u_i\}$ for $i = 0, \ldots, N-1$ is often referred to as a control sequence or control schedule. The sequence $\{x_i\}$ is referred to as the state trajectory. A control policy π is defined as a sequence of state feedback control laws $u_i = h_i(x_i)$. Let π° denote the optimal policy. The cost function that we wish to minimize is defined by

$$V_0\left(x_0, \{u_i\}\right) = \sum_{i=0}^{N-1} L_i(x_i, u_i) + F(x_N) \tag{C.3}$$

where L_i and F are nonnegative nonlinear functions representing the one-step cost at time i and terminal cost, respectively.

Dynamic Programming: The dynamic programming solution to this problem is as follows. Let $V_i^\circ(x_i)$ denote the optimal cost – the cost starting from x_i if the optimal policy π° is followed. Then, from the principle of optimality, we have the recursive Bellman equation:

$$V_i^\circ(x_i) = \min_{u_i} \left[L_i(x_i, u_i) + V_{i+1}^\circ(f_i(x_i, u_i))\right] \tag{C.4}$$

Performing the minimization in C.4 at time i yields the optimal control action and the optimal cost for state x_i. Repeating this minimization for all $x_i \in E^n$ yields the optimal cost-to-go from time i, $V_i(\cdot)$, and the optimal control law at time i, $h_i(\cdot)$. Iteration backwards in time, using the boundary condition

$$V_N^\circ(x_N) = F(x_N) \tag{C.5}$$

yields the optimal cost function $V_0^\circ(\cdot)$ [and all intermediate optimal cost-to-go functions $V_i^\circ(\cdot)$] as well as the optimal policy $\pi^\circ = \left\{h_0^\circ(\cdot), \ldots, h_{N-1}^\circ(\cdot)\right\}$. Such direct dynamic programming implementations suffer from the curse of dimensionality: both computational time and storage requirements are difficult to meet for systems with more than two or three states. Iterative techniques get around this by comparing the nominal trajectory with neighboring trajectories and selecting the neighboring trajectory that yields the most significant decrease in cost. The new trajectory, and the associated control sequence become the nominal sequences for the next iteration.

Differential Dynamic Programming: In this section we give an overview of the algorithm. As mentioned above, the central idea of DDP and other iterative gradient algorithms is to search neighboring trajectories in the hopes of slightly lowering the cost in each iteration and converging to a local minima. To accomplish this efficiently, DDP computes an approximation of the value function along the current nominal trajectory by a Taylor series expansion (second order terms are retained). Then, a new linear state feedback control policy is hypothesized. This policy is substituted into the approximated value function. Finally, the principle of optimality is used to choose the control that will minimize the approximated value function. The steps up to this point are referred to as the backward-sweep. This new control policy is used to generate a new trajectory, and the new cost is compared with the previous cost (forward-sweep). If the cost has improved, then we begin another iteration. If the cost has not improved, the new control policy is reduced in magnitude and the forward-sweep is computed again. A critical requirement for the algorithm to succeed is that variations in the state from the nominal state due to the new control sequence should remain sufficiently small so that the Taylor series expansion is a good approximation. Next, we will give the full definition of the algorithm and its derivation.

C.2.1 Algorithm overview

An overview of the differential dynamic programming algorithm is shown in Figure C.1.

C.2.2 Derivation

The algorithm begins with a nominal control schedule $\{\bar{u}_i\}$ and corresponding state trajectory $\{\bar{x}_i\}$. The value function (non-optimal) obtained from these nominal sequences is $\bar{V}_i(x_i, \{\bar{u}_i\})$. If this function is a sufficiently smooth function of x_i, it can be approximated in a small neighborhood of $\bar{x}$ by the following Taylor series expansion (truncated after second order terms):

$$\bar{V}_i(x_i) = \bar{V}_i(\bar{x}_i) + \left[\bar{V}_x^i(\bar{x}_i)\right]^T \delta x_i + \frac{1}{2}\delta x_i^T \left[\bar{V}_{xx}^i(\bar{x}_i)\right] \delta x_i \tag{C.6}$$

where $\bar{V}_x^i$ is a column vector whose j-th component is given by $\partial/\partial(x_i)_j \bar{V}_i(\bar{x}_i)$. Now, the terms $\bar{V}_i(\bar{x}_i), \bar{V}_x^i(\bar{x}_i), \bar{V}_{xx}^i(\bar{x}_i)$ are the terms we need to calculate. We will find recursive equations for these terms. First, lets recall the recursive

DDP algorithm overview:

1. Given nominal control schedule $\{u_i\}$ and initial condition $\bar{x}_0$, compute nominal trajectory $\{x_i\}$.
2. *Backward-sweep:* Calculate second-order approximation of value function. Compute δu_i that maximally reduces cost. Store α_i, β_i, and estimated cost a_i (use Equations (C.27), (C.28)).
3. *Forward-sweep:* Set $\varepsilon = 1$. Calculate the new control and state sequences using:

   $$\begin{aligned} x_{i+1} &= f_i(x_i, u_i), \qquad x_0 = \bar{x}_0 \\ u_i &= u_i + \varepsilon\alpha_i + \beta_i(x_i - x_i) \end{aligned}$$

   Calculate the actual change in cost:

   $$\begin{aligned} \Delta V_0 = F(x_N) &- F(\bar{x}_N) \\ &+ \sum_{i=0}^{N-1}[L_i(x_i, u_i) - L_i(x_i, u_i)] \end{aligned}$$

   If ΔV_0 is positive (new policy increased cost) or if it is negative but,

   $$\frac{|\Delta V_0|}{|\varepsilon(1 - \varepsilon/2)a_0|} < c$$

   set $\varepsilon = \varepsilon/2$ and go back to beginning of Step 2 (forward sweep). Otherwise, $\Delta V_0 < 0$ and is not too much smaller than the predicted change in cost, and the new sequences $\{x_i\}$ and $\{u_i\}$ are stored and used as the nominal sequences in a new iteration starting with the Step 1 (backward-sweep).
4. The algorithm is terminated when

   $$|a_0(\bar{x}_0)| \leq \eta$$

   where $\eta > 0$ is a small quantity determined from numerical stability considerations.

Figure C.1: DDP algorithm overview.

Bellman equation (principle of dynamic programming):

$$\bar{V}_i(x_i) = L_i(x_i, u_i) + \bar{V}_{i+1}(x_{i+1}) \qquad \text{(C.7)}$$

with

$$x_{i+1} = f_i(x_i, u_i) \tag{C.8}$$

and boundary condition

$$\bar{V}_N(x_N) = F(x_N) \tag{C.9}$$

Also, lets introduce a pseudo-Hamiltonian function:

$$H_i(x_i, u_i, \lambda) = L_i(x_i, u_i) + \lambda^T f_i(x_i, u_i) \tag{C.10}$$

for $i = 0, \dots, N-1$. Now, to obtain recursive equations for the Jacobian and Hessian of $\bar{V}_i(\cdot)$, let us differentiate both sides of Equation (C.7):

$$\begin{aligned} \bar{V}_x^i(x_i) &= L_x^i(x_i, u_i) + [f_x^i(x_i, u_i)]^T \bar{V}_x^{i+1}(x_{i+1}) \\ &= H_x^i(x_i, u_i, \bar{V}_x^{i+1}(x_{i+1}) \end{aligned} \tag{C.11}$$

Now, let us differentiate Equation (C.11) with respect to x_i:

$$\begin{aligned} \bar{V}_{xx}^i(x_i) &= H_{xx}^i(x_i, u_i, \bar{V}_x^{i+1}(x_{i+1})) \\ &\quad + [f_x^i(x_i, u_i)]^T [\bar{V}_{xx}^{i+1}(x_{i+1})][f_x^i(x_i, u_i)] \end{aligned} \tag{C.12}$$

Letting $x_i = x_i$, we now have difference equations for $\bar{V}_i(\bar{x}_i), \bar{V}_x^i(\bar{x}_i), \bar{V}_{xx}^i(\bar{x}_i)$:

$$\begin{aligned} \bar{V}_i(x_i) &= L_i(x_i, u_i) + \bar{V}_i(\bar{x}_{i+1}) \\ \bar{V}_x^i(x_i) &= H_x^i(x_i, u_i, \bar{V}_x^{i+1}(\bar{x}_{i+1}) \\ \bar{V}_{xx}^i(x_i) &= H_{xx}^i(x_i, u_i, \bar{V}_x^{i+1}(\bar{x}_{i+1})) \\ &\quad + [f_x^i(x_i, u_i)]^T [\bar{V}_{xx}^{i+1}(\bar{x}_{i+1})][f_x^i(x_i, u_i)] \end{aligned} \tag{C.13}$$

with boundary conditions

$$\begin{aligned} \bar{V}_N(\bar{x}_N) &= F(\bar{x}_N) \\ \bar{V}_x^N(\bar{x}_N) &= F_x(\bar{x}_N) \\ \bar{V}_{xx}^N(\bar{x}_N) &= F_{xx}(\bar{x}_N) \end{aligned} \tag{C.14}$$

These equations are calculated backward in time and produce the time history of the Taylor series approximation to the value function in a small neighborhood of x_i. Also, by replacing u_i with $h_i(x_i)$, and $\bar{V}_i$ with $V_i(x_i, \pi)$ we obtain the Taylor series approximation of $V_i(x_i, \pi)$ about x_i. Recall that

$V_i(x_i, \pi)$ is the cost due to initial condition x_i and policy $\pi = \{h_0(\cdot), \ldots, h_{N-1}(\cdot)\}$.

We will now apply the above differentiation of the Bellman equation, with the following additional assumptions:

$$u_i = u_i + \delta u_i(x_i) \tag{C.15}$$

and

$$\delta x_i = x_i - x_i \tag{C.16}$$

where δu_i is a function of x_i. The goal now is to find $\delta u_i(x_i)$ that minimizes the approximate value function in the neighborhood of x_i.

Now, using Equations (C.15) and (C.16), our (non-optimal) value function becomes:

$$\begin{aligned} V_i(x_i + \delta x_i) = {} & L_i(x_i + \delta x_i, u_i + \delta u_i) \\ & + V_{i+1}(\bar{x}_{i+1} + \delta x_{i+1}) \end{aligned} \tag{C.17}$$

where

$$\delta x_{i+1} = f_i(x_i + \delta x_i, u_i + \delta u_i) - \bar{x}_{i+1} \tag{C.18}$$

Now, expand both sides of Equation (C.17) about the x_i and u_i:

$$\begin{aligned} & V_i(x_i) + [V_x^i(x_i)]^T \delta x_i + \frac{1}{2}\delta x_i^T [V_{xx}^i(x_i)] \delta x_i = \\ & \quad L_i(x_i, u_i) + V_{i+1}(\bar{x}_{i+1}) \\ & \quad + [H_x^i]^T \delta x_i + [H_u^i] \delta u_i + \frac{1}{2}\delta x_i^T [H_{xx}^i] \delta x_i \\ & \quad + \delta u_i^T [H_{uu}^i] \delta x_i + \frac{1}{2}\delta u_i^T [H_{uu}^i] \delta u_i \\ & \quad + \frac{1}{2}\delta x_i^T [f_x^i]^T [V_{xx}^i(\bar{x}_{i+1})][f_x^i] \delta x_i \\ & \quad + \delta u_i^T [f_u^i]^T [V_{xx}^{i+1}(\bar{x}_{i+1})][f_x^i] \delta x_i \\ & \quad + \frac{1}{2}\delta u_i^T [f_u^i]^T [V_{xx}^{i+1}(\bar{x}_{i+1})][f_u^i] \delta u_i + h.o.t. \end{aligned} \tag{C.19}$$

We now apply the principle of optimality and find δu_i to minimize the left-hand side of Equation (C.19). Dropping the higher order terms, and considering unconstrained δu_i, we can find the minimizing value by differentiating the right-hand side of Equation (C.19) with respect to δu_i. The resulting expression is

$$\begin{aligned}0 &= H_u^i \\ &\quad + \left(H_{uu}^i + [f_u^i]^T[V_{xx}^{i+1}(\bar{x}_{i+1})][f_u^i]\right)\delta u_i \\ &\quad + \left(H_{ux}^i + [f_u^i]^T[V_{xx}^{i+1}(\bar{x}_{i+1})][f_x^i]\right)\delta x_i \\ &= H_u^i + C_i\delta u_i + B_i\delta x_i\end{aligned} \tag{C.20}$$

where

$$\begin{aligned}B_i &= H_{ux}^i + [f_u^i]^T[V_{xx}^{i+1}(\bar{x}_{i+1})][f_x^i] \\ C_i &= H_{uu}^i + [f_u^i]^T[V_{xx}^{i+1}(\bar{x}_{i+1})][f_u^i]\end{aligned} \tag{C.21}$$

Thus, if C_i is positive-definite , the unique minimum of the right-hand side of Equation (C.19) is

$$\begin{aligned}\delta u_i &= -[C_i]^{-1}H_u^i - [C_i]^{-1}B_i\delta x_i \\ &= \alpha_i + \beta_i\delta x_i\end{aligned} \tag{C.22}$$

where

$$\begin{aligned}\alpha_i &= -[C_i]^{-1}H_u^i \\ \beta_i &= -[C_i]^{-1}B_i\end{aligned} \tag{C.23}$$

We can now substitute this expression for δu_i back into Equation (C.19), and equate coefficients of like-powers of δx_i. This will give us recursive difference equations for $V_i(x_i), V_x^i(x_i), V_{xx}^i(x_i)$. First, let us define the difference in cost obtained when starting from state x_i and using the nominal control schedule $\{\bar{u}_i, \ldots, \bar{u}_{N-1}\}$ and the new schedule $\{u_i, \ldots, u_{N-1}\}$:

$$a_i = V_i(x_i) - \bar{V}_i(x_i) \tag{C.24}$$

where $\bar{V}_i(x_i)$ is the cost starting from x_i and following policy $\{\bar{u}_i, \ldots, \bar{u}_{N-1}\}$, and $V_i(x_i)$ is the cost starting from x_i and following policy $\{u_i, \ldots, u_{N-1}\}$. Now, plugging in $\delta u_i = \alpha_i + \beta_i\delta x_i$ into Equation (C.19) and equating like powers of δx_i yields

$$\begin{aligned} a_i &= a_{i+1} + [H_u^i]^T \alpha_i + \frac{1}{2}\alpha_i^T C_i \alpha_i \\ V_x^i(x_i) &= H_x^i + \beta_i^T H_u^i + [C_i\beta_i + B_i]^T \alpha_i \\ V_{xx}^i(x_i) &= A_i + \beta_i^T C_i \beta_i + \beta_i^T B_i + B_i^T \beta_i \end{aligned} \tag{C.25}$$

where

$$A_i = H_{xx}^i + [f_x^i]^T [V_{xx}^{i+1}(\bar{x}_{i+1})][f_x^i] \tag{C.26}$$

Finally, plugging in the values for α_i and β_i from Equations (C.23) into Equation (C.25), we have

$$\begin{aligned} a_i &= a_{i+1} - \frac{1}{2}[H_u^i]^T [C_i]^{-1} [H_u^i] \\ V_x^i(x_i) &= H_x^i + \beta_i^T H_u^i \\ V_{xx}^i(x_i) &= A_i - \beta_i^T C_i \beta_i \end{aligned} \tag{C.27}$$

and the boundary conditions are:

$$\begin{aligned} a_N &= 0 \\ V_x^N(\bar{x}_N) &= F_x^N(\bar{x}_N) \\ V_{xx}^N(\bar{x}_N) &= F_{xx}^N(\bar{x}_N) \end{aligned} \tag{C.28}$$

At this point, we have an estimate for a linear controller that will maximally improve the cost (approximated to second-order). However, this only holds when δx_i is sufficiently small. Also note that the parameter that directly affects the magnitude of δx_i is α_i. If the derived contol improvement does not actually lead to a lower cost, we can replace Equation (C.23) with

$$\alpha_i = -\varepsilon [C_i]^{-1} H_u^i \tag{C.29}$$

where $\varepsilon > 0$. If C_i is positive-definite, then this ensures that α_i is of order ε. Once we have the new control policy for a particular ε, we can compute the estimated change in total cost:

$$a_0 = -\varepsilon(1 - \varepsilon/2) \sum_{i=0}^{N-1} [H_u^i]^T [C_i]^{-1} [H_u^i] \tag{C.30}$$

and thus a_0 is of order ε. If C_i is positive-definite, then a_0 is negative. Let ΔV_0 denote the true change in cost. Then,

$$|\Delta V_0 - a_0| = O(\varepsilon^3) \tag{C.31}$$

Thus, there exists an ε sufficiently small such that $a_0 < 0$ guarantees $\Delta V_0 < 0$, i.e. gaurantees that the new policy produces a reduction in cost. Finally, the algorithm is terminated when the reduction in cost passes a fixed threshold that is chosen by considering numerical stability.

C.3 Some properties of DDP

C.3.1 Reduction to LQR

An important property of the algorithm is that it reduces to the standard LQR solution in one iteration when the system is linear and the cost is quadratic in the state and control. This is easily shown – consider the system and cost function:

$$x_{i+1} = Ax_i + Bu_i \tag{C.32}$$

$$V_0(x_0) = x_N^T Q_f x + \frac{1}{2}\sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \tag{C.33}$$

We then have:

$$H_i(x_i, u_i, \lambda) = x_i Q^T x_i + u_i^T R u_i + \lambda^T f_i(x_i, u_i) \tag{C.34}$$

and

$$\begin{aligned} H_x^i &= x_i^T Q + \lambda^T A \\ H_u^i &= u_i^T R + \lambda^T B \\ H_{uu}^i &= R \\ H_{ux}^i &= 0 \end{aligned} \tag{C.35}$$

Now the difference equation for V_{xx}^i in Equations (C.27),(C.28) becomes the matrix Riccati difference equation.

$$\begin{aligned} V_{xx}^i(x_i) &= H_{xx} + [f_x^i]^T [V_{xx}^{i+1}][f_x^i] - \beta^T C \beta \\ &= Q + A^T V_{xx}^{i+1} A - \beta^T C_i \beta \\ &= Q + A^T V_{xx}^{i+1} A - [-C_i^{-1} B_i]^T [C_i][-C_i^{-1} B_i] \\ &= Q + A^T V_{xx}^{i+1} A - B_i^T C_i^{-1} B_i \end{aligned} \tag{C.36}$$

where B_i reduces to

$$\begin{aligned} B_i &= H_{ux} + [f_u^i]^T [V_{xx}^{i+1}][f_x^i] \\ &= B^T V_{xx}^{i+1} A \end{aligned} \tag{C.37}$$

and C_i reduces to

$$\begin{aligned} C_i &= H_{uu} + [f_u^i]^T [V_{xx}^{i+1}][f_u^i] \\ &= R + B^T V_{xx}^{i+1} B \end{aligned} \tag{C.38}$$

Thus, letting $v_{xx}^i = P_i$, we obtain the matrix Riccati difference equation

$$\begin{aligned} P_{t-1} = Q &+ A^T P_t A \\ &- A^T P_t B \left(R + B^T P_t B \right)^{-1} B^T P_t A \end{aligned} \tag{C.39}$$

with boundary condition

$$P_N = Q_f \tag{C.40}$$

Similarly, we can see that the new control reduces to the LQR solution

$$\begin{aligned} \delta u_i &= \alpha_i + \beta_i \delta x_i \\ &= -C_i^{-1} H_u - C_i^{-1} B_i \delta x_i \\ &= -C_i^{-1} [H_u - B_i \delta x_i] \\ &= -(R + B^T V_{xx}^{i+1} B)^{-1} B^T V_{xx}^{i+1} A \delta x_i \end{aligned} \tag{C.41}$$

noting that $H_u = 0$ is satisfied through construction of the co-state equation in the LQR solution.

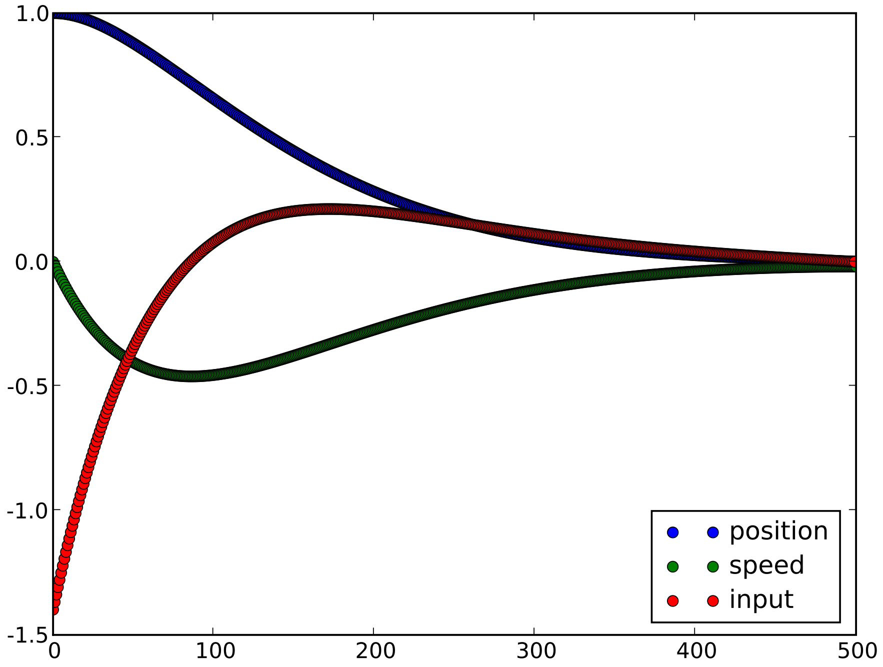


Figure C.2: DDP results from linear system with quadratic cost. Algorithm converged in one iteration to the standard LQR solution.

C.4 Simulations

In this section, we will show two simple simulations: one that demonstrates the single iteration solution to an LQR problem, and one that demonstrates the solution to an optimal control problem where the system is nonlinear in control. The second-order DDP algorithm is implemented in python (the file can be found under `trunk/documents/miles/mjj-0001/ddp.py`).

C.4.1 Simple LQR problem

Consider a point mass being accelerated by a scalar input u. This system is given by

$$x_{i+1} = Ax_i + Bu_i \qquad x_0 = [1,\, 0]^T \tag{C.42}$$

and cost function

$$V_0(x_0) = x_N^T Q_f x + \frac{1}{2}\sum_{i=0}^{N-1} x_i^T Q x_i + Ru_i^2. \tag{C.43}$$

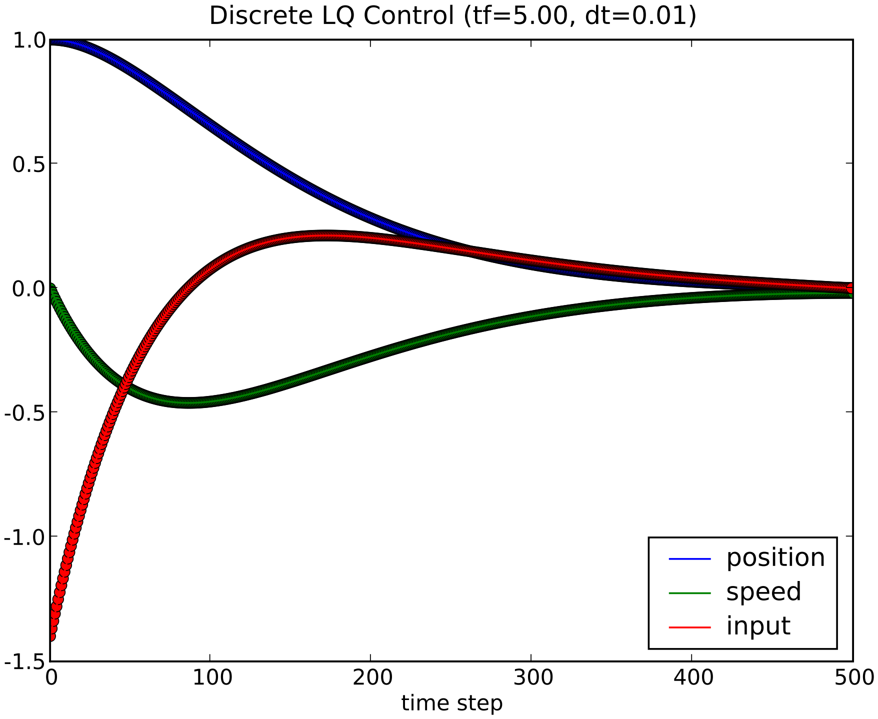


Figure C.3: LQR results from linear system with quadratic cost.

The system and cost parameters are given by

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \tag{C.44}$$

and

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R = 0.5 \quad Q_f = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{C.45}$$

Now use DDP to find the control sequence $\{u_i\}$ that minimizes the cost in Equation (C.43). To initialize the algorithm, let $u_i = 0$ for $i = 0, \ldots, N-1$. Also, let $\Delta t = 0.01$, and $N = 500$ (equivalent to $t_f = 5$ seconds).

Results: After one iteration, the DDP algorithm produces the results shown in Figure C.2 (this can be compared to the LQR solution shown in Figure C.3). The optimal costs for the DDP solution and LQR solution are shown in Table C.1.

	LQR	DDP
Optimal cost	156.367	156.366

Table C.1: Optimal costs for LQR and DDP solutions.

C.4.2 Nonlinear in control problem

Consider the following continuous-time problem:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \text{sat}(u) \end{aligned} \tag{C.46}$$

where

$$\text{sat}(u) = \begin{cases} u, & |u| \leq D \\ 1, & u > D \\ -1, & u < -D \end{cases} \tag{C.47}$$

where $D > 0$. Let the cost function of this continuous-time problem be given by

$$V_0 = \int_0^{t_f} \frac{1}{2} x_1^2 dt. \tag{C.48}$$

The discrete-time version of this problem is given by:

$$\begin{aligned} x_{i+1}^1 &= x_i^1 + \Delta t x_i^2 + \frac{\Delta t^2}{2} s(u_i) \\ x_{i+1}^2 &= x_i^2 + \Delta t s(u_i) \end{aligned} \tag{C.49}$$

where

$$s(u_i) = \begin{cases} u_i & |u_i| \leq D \\ 1 - (1-D) exp\left\{\frac{-u+D}{1-D}\right\} & u > D \\ -1 + (1-D) exp\left\{\frac{u+D}{1-D}\right\} & u < -D \end{cases} \tag{C.50}$$

And the cost function for the discrete-time version becomes:

$$V_0 = \frac{1}{2}[x_N^1]^2 + \frac{1}{2}\sum_{i=0}^{N-1}\left([x_i^1]^2 + Ru_i^2\right) \tag{C.51}$$

The system and cost can be written in a more compact way. The system is

$$x_{i+1} = Ax_i + g(u) \tag{C.52}$$

where

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{C.53}$$

and

$$g(u) = \begin{bmatrix} \frac{\Delta t^2}{2} s(u) \\ \Delta t s(u) \end{bmatrix} \tag{C.54}$$

The cost is

$$V_0 = x_N^T Q_f x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + R u_i^2 \tag{C.55}$$

where

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad Q_f = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{C.56}$$

Now, some of the intermediate variables we have to provide the DDP algorithm are:

$$f_x^i = A \tag{C.57}$$

$$f_u^i = \begin{bmatrix} \Delta t^2/2 s_u(u) \\ \Delta t s_u(u) \end{bmatrix} \tag{C.58}$$

where

$$s_u(u) = \begin{cases} 1 & |u| \leq D \\ exp\left\{\frac{-u+D}{1-D}\right\} & u > D \\ exp\left\{\frac{u+D}{1-D}\right\} & u < -D \end{cases} \tag{C.59}$$

$$f_{uu}^i = \begin{bmatrix} \Delta t^2/2 s_{uu}(u) \\ \Delta t s_{uu}(u) \end{bmatrix} \tag{C.60}$$

where

$$s_{uu}(u) = \begin{cases} 0 & |u| \leq D \\ \frac{-1}{1-D} exp\left\{\frac{-u+D}{1-D}\right\} & u > D \\ \frac{1}{1-D} exp\left\{\frac{u+D}{1-D}\right\} & u < -D \end{cases} \tag{C.61}$$

and derivatives of the Hamiltonian

$$
\begin{aligned}
H_u^i &= Ru_i + [V_x^{i+1}]^T[f_u^i] \\
H_x^i &= x_i^T Q + [V_x^{i+1}]^T[f_x^i] \\
H_{ux}^i &= 0 \\
H_{uu}^i &= R + [V_x^{i+1}]^T[f_{uu}^i] \\
H_{xx}^i &= Q
\end{aligned}
\tag{C.62}
$$

In the simulation Δt to 0.01 seconds, and N was set to 1000, corresponding to $t_f = 10$ seconds.

Results: Results for two values of R are shown in Figure C.4. This figure shows the phase portrait x_1 vs x_2 for two different values of R. The red line in the figure shows the switching curve defined by $x_1 = -0.4446x_2|x_2|$ that is obtained in the optimal solution of the continuous-time problem. An interesting note here is that the number of iterations required by the DDP algorithm increases as R decreases. As R decreases, the control is able make larger swings, and this effects the change in cost. A difficulty with this system is that even though the algorithm attempts to change the control, the saturation function reduces the affect of changing δu_i on the total cost.

C.5 First order method

Finally, it is interesting to note that a simple first-order algorithm can be obtained by neglecting second-order and above terms in the previous derivation. Thus, the local control law becomes

$$
\delta u_i = -\varepsilon H_u^i \tag{C.63}
$$

And recursive equations for V_i and V_x^i become:

$$
\begin{aligned}
a_i &= a_{i+1} - \varepsilon[H_u^i]^T[H_u^i] \\
V_x^i(x_i) &= H_x^i
\end{aligned}
\tag{C.64}
$$

with boundary conditions as given before. It is clear here that there will be a reduction in cost if $H_u^i \neq 0$. Now, the simple first-order algorithm is shown in Figure C.5.

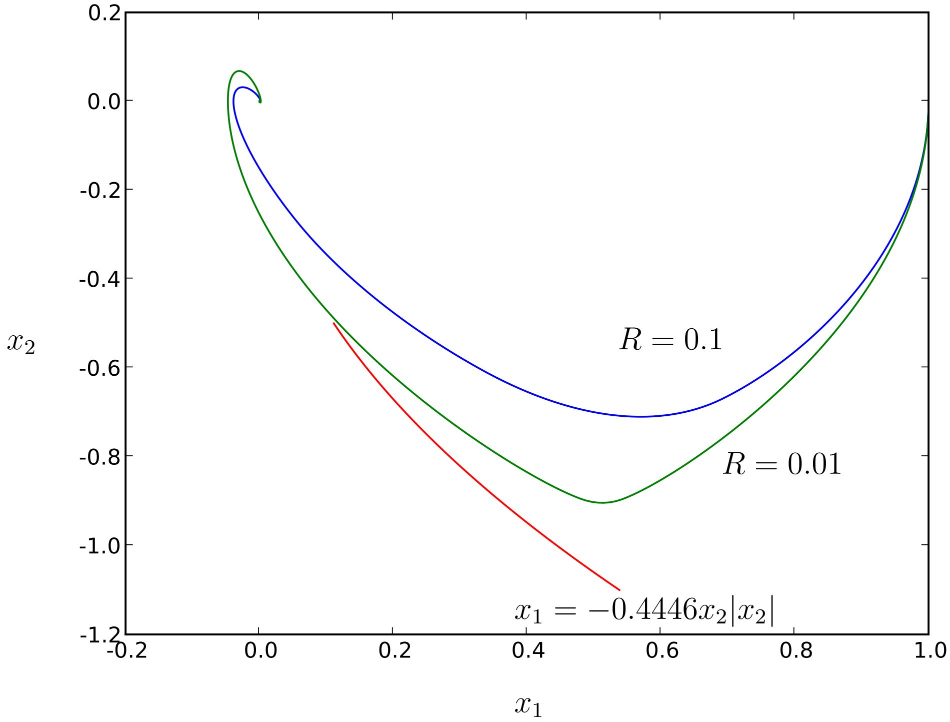


Figure C.4: DDP results for system with nonlinearity in control. This figure shows a phase portrait with x_1 (position) on the x-axis and x_2 (velocity) on the y-axis. The red line shows the switching curve $x_2 = -0.4446x_1|x_1|$ that the discrete-time DDP solution approaches as R is reduced.

First-order DDP algorithm overview:

1. Given nominal control schedule $\{u_i\}$ and initial condition x_0, calculate trajectory $\{x_i\}$.

2. *Backward-sweep:* Starting from the final time, calculate $\{a_i\}$ and $\{V_x^i(x_i)\}$ using Equations (C.64) and boundary conditions given in Equation (C.28).

3. *Forward-sweep:* Set $\varepsilon = 1$. Calculate new control and state sequences using

$$\begin{aligned} x_{i+1} &= f_i(x_i, u_i), \qquad x_0 = \bar{x}_0 \\ u_i &= u_i - \varepsilon H_u^i \end{aligned}$$

   Calculate the actual change in cost ΔV_0 and compare with the estimated reduction $|a_0|$. If ΔV_0 is sufficiently negative, accept the new control and trajectory, otherwise reduce ε and go back to beginning of Step 3 (forward-sweep).

4. The algorithm terminates when $|a_0| < \eta$ for small $\eta > 0$.

Figure C.5: First-order DDP algorithm overview.

Appendix D

Iterative LQR

D.1 Introduction

Iterative linear quadratic regulation (ILQR) is an iterative, gradient based, local dynamic programming method. This method also falls under the category of model-based policy search algorithms in reinforcement learning, in which control policies are found using a model or simulator of a dynamic system. Models range from general Markov decision processes (MDPs) to deterministic ordinary differential equations. However, it is often the case that ones model is known to be inaccurate, or only an approximation of a real system, and thus control policies are found that work well in simulation, but poorly in real-life. In this paper, we are concerned with learning control policies that allow steady horizontal flight of a real aircraft, using reduced order kinematic model as an initialization only. We describe the techniques in [138, 143] that can be used for vehicle trajectory tracking problems. We briefly discuss the related work of [**?**, 144] that describe algorithms to learn to fly an aircraft using behavioral cloning and relational abstraction of the state space.

D.2 Problem Statement

Our objective is to design a control policy that causes a UAV to track a smooth planar trajectory in the presence of environmental disturbances (steady and turbulent wind). Further, we assume the trajectory is composed of piecewise circular arc segments of constant arc length. We also assume that we have prior knowledge of this target trajectory up to some finite horizon. Many performance measures can be considered, but for this paper we will consider cost functions that are quadratic in the deviation from the target trajectory and integrated over the duration of the flight.

Consider a dynamic system

$$x_{k+1} = f_k(x_k, u_k) \tag{D.1}$$

that can be linearized about a nominal trajectory $\{x_k, u_k\}$ to

$$x_{k+1} = A_k x_k + B_k u_k \tag{D.2}$$

The accuracy of linearization is an important issue that we will return to in more detail below. Our objective is to track a target trajectory $\{x_k^*, u_k^*\}$ over a finite horizon N, and we will do this by minimizing a quadratic cost function

$$\begin{aligned} J(x_0) = (x_N - x_N^*)^T Q_N (x_N - x_N^*) \\ + \sum_{k=0}^{N-1} \big((x_k - x_k^*)^T Q_k (x_k - x_k^*) \\ + (u_k - u_k^*)^T R_k (u_k - u_k^*) \big) \end{aligned} \tag{D.3}$$

where Q, Q_N are positive semi-definite, and R is positive definite. Thus far, we have described the general linear quadratic tracking (LQR) problem. However, we will further consider *system dynamics models that are inaccurate, or only an approximation of the real system.* We have access to the real system – we can run a control policy and observe the resulting system behavior. Thus our problem is to use *trials* of the real system to learn a control policy that causes the real system to track the target trajectory.

In the next section, we will briefly review standard LQR. Following that, we will review iterative LQR (iLQR) in the reinforcement learning context.

D.3 Tracking LQR

LQR is an optimal control algorithm that computes a feedback control policy that drives a system such that a quadratic cost function is minimized. There are many minor variations that cover continuous time system, discrete time systems, infinite horizon, finite horizon, set point regulation, servo problems, tracking problems, model-following problems [**?**,17]. We will consider discrete time, finite horizon, set point regulation and tracking formulations.

D.3.1 Set point Regulation

The LQR algorithm assumes a linear (or linearized) dynamical system

$$x_{k+1} = A_k x_k + B_k u_k$$

and a quadratic cost function

$$J(x_0) = \sum_{0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$

Minimizing this cost function will generally drive the state to the origin. There are generally two ways to derive the LQR solution: using the *variational methods* and using *dynamic programming*. We will briefly highlight a dynamic programming derivation in this section, and show a more involved minimum principle derivation in Section D.4.2. A starting point for the derivation is *value iteration*. The value of a state x is the expected future cost of the system starting from x and executing policy π. A recursive equation can be written for the optimal value function

$$J_{i+1}(x) = \min_u [x^T Q x + u^T R u + \sum_{x' = Ax+Bu} J_i(x')] \tag{D.4}$$

For deterministic systems as implied above, this becomes

$$J_{i+1}(x) = \min_u [x^T Q x + u^T R u + J_i(Ax + Bu)] \tag{D.5}$$

Now, assume that J_0 has a quadratic form:

$$J_0(x) = x^T P_0 x \tag{D.6}$$

Then J_1 becomes

$$\begin{aligned} J_1(x) &= \min_u [x^T Q x + u^T R u + J_0(Ax + Bu)] \\ &= \min_u [x^T Q x + u^T R u + (Ax + Bu)^T P_0 (Ax + Bu)] \end{aligned} \tag{D.7}$$

Now, we can try to find the optimal u by taking the derivative of the bracketed expression with respect to u and setting it equal to zero to obtain

$$Ru + B^T P_0(Ax + Bu) = 0 \tag{D.8}$$

Solving (C.38) for u yields

$$u = -(R + B^T P_0 B)^{-1} B^T P_0 Ax \tag{D.9}$$

Substituting this expression for u back into the value iteration equation (C.37) yields

$$J_1(x) = x^T P_1 x \tag{D.10}$$

where

$$\begin{aligned} P_1 &= Q + K_1^T R K_1 + (A + BK_1)^T P_0 (A + BK_1) \\ K_1 &= -(R + B^T P_0 B)^{-1} B^T P_0 A \end{aligned} \tag{D.11}$$

This process can be continued, and it can be shown that the form of the value iteration update is the same at each iteration, and can be reduced to a closed form computation. In summary, set $P_0 = 0$. Then for $i = 1, 2, 3, \ldots$

$$\begin{aligned} K_i &= -(R + B^T P_{i-1} B)^{-1} B^T P_{i-1} A \\ P_i &= Q + K_i^T R K_i + (A + BK_i)^T P_{i-1} (A + BK_i) \end{aligned} \tag{D.12}$$

and the optimal policy is given by

$$\pi(xv) = K_i x \tag{D.13}$$

and the optimal cost-to-go function is given by

$$J_i(x) = x^T P_i x \tag{D.14}$$

D.3.2 Trajectory Tracking

Let us now consider LQR for trajectory tracking under nonlinear system dynamics. Let x_k^*, u_k^* denote the desired target trajectory. We now want to minimize the quadratic cost function

$$J(x_0) = \sum_{k=0}^{N-1} (x_k - x_k^*)^T Q (x_k - x_k^*) + (u_k - u_k^*)^T R (u_k - u_k^*) \tag{D.15}$$

subject to the system dynamics

$$x_{k+1} = f_k(x_k, u_k) \tag{D.16}$$

We now linearize the system dynamics around the target trajectory

$$x_{k+1} \approx f_k(x_k^*, u_k^*) + A_k(x_k - x_k^*) + B_k(u_k - u_k^*) \tag{D.17}$$

where $A_k = D_x f_k(x_k, u_k)$, $B_k = D_u f_k(x_k, u_k)$, and D_x, D_u denote the Jacobian of $f_k(\cdot)$ with respect to x and u respectively. Now we have

$$x_{k+1} - x_{k+1}^* \approx A_k(x_k - x_k^*) + B_k(u_k - u_k^*) \tag{D.18}$$

and we can run the standard LQR backup iterations. These result in the control sequence

$$u_k - u_k^* = K_k(x_k - x_k^*) \tag{D.19}$$

that can be rearranged

$$u_k = u_k^* + K_k(x_k - x_k^*) \tag{D.20}$$

D.4 Iterative-LQR Algorithm

D.4.1 Algorithm Overview

Iterative-LQR is a local policy improvement algorithm – it iteratively makes local improvements upon the current policy. Each iteration starts with a nominal control state-action sequence $\bar{x}_k, \bar{u}_k$, where the states $\bar{x}_k$ are obtained by running the system $f_k(\cdot, \cdot)$, and the inputs $\bar{u}_k$ are obtained from a control policy π

$$\pi\colon\ \ \bar{u}_k = h_k(\bar{x}_k) \quad \forall k \in 0, \ldots, N-1$$

iLQR algorithm overview:

1. Set $i = 0$. Obtain initial policy $\pi^{(0)}$.
2. Execute policy $\pi^{(i)}$ and record state-action history $\{x_0^{(i)}, u_0^{(i)}, \dots\}$
3. Linearize $f_k(x, u)$ about $\{x_0^{(i)}, u_0^{(i)}, \dots\}$ to obtain A_k, B_k
4. Run LQR that computes policy

$$\pi^{(i+1)}\colon \;\; h_k^{(i+1)}(x_k^{(i)}) = u_k^* + b_k + K_k(x_k^{(i+1)} - x_k^*)$$

5. Set $i = i + 1$.

Figure D.1: ILQR algorithm overview.

D.4.2 Derivation

Let us first derive the ILQR algorithm in the context of a standard set point regulation problem [138]. Consider the system

$$x_{k+1} = f_k(x_k, u_k) \tag{D.21}$$

and cost function

$$\begin{aligned} J_0 = \frac{1}{2}(x_N - x_N^*)^T Q_N (x_N - x_N^*) \\ + \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) \end{aligned} \tag{D.22}$$

The ILQR algorithm is initialized with a nominal control sequence u_k and corresponding nominal trajectory x_k obtained by applying u_k to the system. A typical initialization is $u_k = 0$. Each iteration produces an improved control sequence by linearizing the system dynamics around u_k, x_k and solving an LQR problem. This process repeats until convergence. Denote deviations from the nominal u_k, x_k by $\delta u_k, \delta x_k$. The linearized system is then given by

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{D.23}$$

where $A_k = D_x f_k(x_k, u_k)$, $B_k = D_u f_k(x_k, u_k)$, and D_x, D_u denote the Jacobian of $f_k(\cdot)$ with respect to x and u respectively. Using the linearized model, we can write the cost function

$$\begin{aligned} J = &\frac{1}{2}(x_N + \delta x_N - x_N^*)^T Q_N (x_N + \delta x_N - x_N^*) \\ &+ \frac{1}{2}\sum_{k=0}^{N-1} \big((x_k + \delta x_k)^T Q (x_k + \delta x_k) \\ &+ (u_k + \delta u_k)^T R (u_k + \delta u_k)\big) \end{aligned} \tag{D.24}$$

Now, define the Hamiltonian function

$$\begin{aligned} H_k = &\frac{1}{2}(x_k + \delta x_k)^T Q (x_k + \delta x_k) \\ &+ \frac{1}{2}(u_k + \delta u_k) R (u_k + \delta u_k) \\ &+ \lambda_{k+1}^T (A_k \delta x_k + B_k \delta u_k) \end{aligned} \tag{D.25}$$

where λ_{k+1} is a Lagrange multiplier. The optimal control improvement δu_k is now computed by setting the derivative of the Hamiltonian with respect to δu_k to zero, and solving the state equation (D.23) and co-state equation

$$\lambda_k = A_k^T \lambda_{k+1} + Q(x_k + \delta x_k) \tag{D.26}$$

with boundary condition

$$\lambda_N = Q_N (x_N + \delta x_N - x_N^*) \tag{D.27}$$

Setting the derivative of the Hamiltonian equal to zero

$$H_{u_k} = R(u_k + \delta u_k) + B_k^T \lambda_{k+1} = 0 \tag{D.28}$$

and solving for δu_k yields

$$\delta u_k = -R^{-1} B_k^T \lambda_{k+1} - u_k \tag{D.29}$$

Now, substituting (D.29) into (D.23) and combining with (D.26), we have the resulting Hamiltonian system

$$\begin{aligned}\begin{pmatrix}\delta x_{k+1}\\ \lambda_k\end{pmatrix} &= \begin{pmatrix}A_k & -B_k R^{-1} B_k^T\\ Q & A_k^T\end{pmatrix}\begin{pmatrix}\delta x_k\\ \lambda_{k+1}\end{pmatrix}\\ &+ \begin{pmatrix}-B_k u_k\\ Q x_k\end{pmatrix}\end{aligned} \tag{D.30}$$

This system is not homogeneous, but is driven by a forcing term that depends on the current trajectory u_k, x_k. Thus, the form of solution for δu_k will not be strictly linear, but will have additional terms that are functions of the current trajectory. Based on the boundary condition (D.27), let us try a solution for the co-state equation of the form

$$\lambda_k = P_k \delta x_k + v_k \tag{D.31}$$

for unknown sequences P_k and v_k. We will now substitute this assumption into the state and co-state equations, apply the matrix inversion lemma, and find the resulting solution for δu_k.

Substituting the assumed form of the costate solution into the state equation (D.23) yields

$$\begin{aligned}\delta x_{k+1} = \left(I + B_k R^{-1} B_k^T P_{k+1}\right)^{-1} (A_k \delta x_k\\ -B_k R^{-1} B_k^T v_{k+1} - B_k u_k\big)\end{aligned} \tag{D.32}$$

Next, substituting the assumption (D.31) into the costate equation (D.26) gives

$$\begin{aligned}P_k x_k + v_k = Q\delta x_k &+ A_k^T P_{k+1} (I + B_k R^{-1} B_k^T P_{k+1})^{-1}\\ &(A_k \delta x_k - B_k R^{-1} B_k^T v_{k+1} - B_k u_k)\\ &+ A_k^T v_{k+1} + Q x_k\end{aligned} \tag{D.33}$$

We now apply the matrix inversion lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1} B \left(D A^{-1} B + C^{-1}\right)^{-1} D A^{-1} \tag{D.34}$$

and group all terms that multiply δx_k and all terms that do not. After this, we obtain

$$\begin{aligned} P_k = A_k^T P_{k+1} \left[I - B_k \left(B_k^T P_{k+1} B_k + R \right)^{-1} B_k^T P_{k+1} \right] A_k \\ + Q \end{aligned} \tag{D.35}$$

and

$$\begin{aligned} v_k = A_k^T v_{k+1} - A_k^T P_{k+1} \left[I - B_k (B_k^T P_{k+1} B_k + R)^{-1} \right. \\ \left. B_k^T P_{k+1} \right] B_k R^{-1} B_k^T v_{k+1} \\ - A_k^T P_{k+1} [I - B_k (B_k^T P_{k+1} B_k + R)^{-1} B_k^T P_{k+1}] B_k u_k \\ + Q x_k \end{aligned} \tag{D.36}$$

Using the matrix inversion lemma again on the matrix inverse in (D.36) we have

$$\begin{aligned} (R + B_k^T P_{k+1} B_k)^{-1} = R^{-1} \\ - (R + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} B_k R^{-1} \end{aligned}$$

The second term in v_k is now

$$-A_k^T P_{k+1} B_k (R + B_k^T P_{k+1} B_k)^{-1} B_k^T v_{k+1}$$

and the third term in v_k becomes

$$-A_k^T P_{k+1} B_k (R + B_k^T P_{k+1} B_k)^{-1} R u_k.$$

To simplify the form of these equations, let us define

$$K = (B_k^T P_{k+1} B_k + R)^{-1} B_k^T P_{k+1} A_k \tag{D.37}$$

Now we can write P_k and v_k as

$$P_k = A_k^T P_{k+1} (A_k - B_k K) + Q \tag{D.38}$$

and

$$v_k = (A_k - B_k K)^T v_{k+1} - K^T R u_k + Q x_k \tag{D.39}$$

Finally, substituting the assumed costate solution (D.31) and (D.32) into

(D.29) gives

$$\begin{aligned}\delta u_k &= -(R + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k \delta x_k \\ &\quad - (R + B_k^T P_{k+1} B_k)^{-1} B_k^T v_{k+1} \\ &\quad - (R + B_k^T P_{k+1} B_k)^{-1} R u_k\end{aligned} \tag{D.40}$$

We now summarize the results in the following

$$\begin{aligned}K &= (B_k^T P_{k+1} B_k + R)^{-1} B_k^T P_{k+1} A_k \\ K_v &= (B_k^T P_{k+1} B_k + R)^{-1} B_k^T \\ K_u &= (B_k^T P_{k+1} B_k + R)^{-1} R \\ P_k &= A_k^T P_{k+1}(A_k - B_k K) + Q \\ v_k &= (A_k - B_k K)^T v_{k+1} - K^T R u_k + Q x_k \\ \delta u_k &= -K_u u_k - K_v v_{k+1} - K \delta x_k\end{aligned} \tag{D.41}$$

with boundary conditions

$$P_N = Q_N \qquad v_N = Q_N(x_N - x_N^*) \tag{D.42}$$

From these boundary conditions, we see that we can solve for the entire sequence of P_k using the backward recursion in (D.41). Once this LQR problem is solved, our improved nominal control sequence becomes

$$u_k^* = u_k + \delta u_k \tag{D.43}$$

D.5 ILQR Given Inaccurate Model

Now that we have derived the ILQR in it's traditional form, we can extend it to the reinforcement learning context. Let us now consider that our system dynamics model is inaccurate, or simply an approximation of a real system. Also, assume that we have access to the real system, enabling us to execute a feedback control policy and record the resulting state-action history. With this information, we can improve our control policy by iterating over real executions.

The major modification here is that when we execute a policy on the real system, for any particular time step k and state x_k, the result of our control input $u_k^{(i)} = h_k^{(i)}(x_k^{(i)})$ will differ from our prediction, i.e.

$$x_{k+1}^{(i)} \neq f_k(x_k^{(i)}, u_k^{(i)}) \tag{D.44}$$

D.5.1 Linear Time Varying Formulation

We will now state ILQR for inaccurate models in the linear time varying (LTV) format. Here again, the LQR objective is to regulate to a set point (drive the state to origin). Again we have the state equation and quadratic cost function

$$x_{k+1} = A_k x_k + B_k u_k$$
$$J = \sum_{0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$

Linearizing the state equation around $x_k^{(i)}, u_k^{(i)}$ gives us

$$\begin{aligned} x_{k+1} = f_k(x_k^{(i)}, u_k^{(i)}) &+ D_x f(x_k^{(i)}, u_k^{(i)})(x_k - x_k^{(i)}) \\ &+ D_u f(x_k^{(i)}, u_k^{(i)})(u_k - u_k^{(i)}) \end{aligned} \tag{D.45}$$

or

$$x_{k+1} = f_k(x_k^{(i)}, u_k^{(i)}) + A_k \delta x_k + B_k \delta u_k \tag{D.46}$$

where $\delta x_k = x_k - x_k^{(i)}$ and $\delta u_k = u_k - u_k^{(i)}$. Subtracting $x_{k+1}^{(i)}$ from both sides yields the following error dynamics

$$\delta x_{k+1} = f_k(x_k^{(i)}, u_k^{(i)}) - x_{k+1}^{(i)} + A_k \delta x_k + B_k \delta u_k \tag{D.47}$$

Now, we can slightly change variables to obtain the standard LQR structure – let $z_k = [\delta x_k^T \quad 1]^T$. Then we can redefine the state equation as

$$\begin{aligned} z_k &= [\delta x_k^T \quad 1]^T \\ w_k &= \delta u_k \\ A_k &= \begin{bmatrix} D_x f_k(x_k^{(i)}, u_k^{(i)}) & f_k(x_k^{(i)}, u_k^{(i)}) - x_{k+1}^{(i)} \\ 0 & 1 \end{bmatrix} \\ B_k &= \begin{bmatrix} D_u f_k(x_k^{(i)}, u_k^{(i)}) \\ 0 \end{bmatrix} \end{aligned} \tag{D.48}$$

Similarly, Q and R can be derived using the new variables.

A very important issue here is that this formulation will work in practice *only when the learned policy keeps the system close to the current nominal trajectory*! This is a very serious assumption, and often in practice some further work must be done to ensure that the as-run state remains close to the nominal trajectory. One way to do this is to modify the cost function in each iteration such that each one step cost function becomes

$$(1-\alpha)g(x_k^{(i)}, u_k^{(i)}) + \alpha(\|\delta x_k\|^2 + \|\delta u_k\|^2) \tag{D.49}$$

As α approaches one the second term will dominate and the LQR solution will work harder to keep the state-action history close to the nominal trajectory, ensuring that the linearization above is a good approximation of the system dynamics.

D.5.2 Trajectory Tracking Formulation

Above we showed ILQR with reinforcement learning for set point regulation. A similar method is outlined for trajectory tracking. Let x_k^*, u_k^* denote a desired target trajectory. Note that this is not necessarily a *feasible* state action history for the real system. Again, let $x_k^{(i)}, u_k^{(i)}$ denote the current nominal trajectory obtained from an execution of the real dynamics. We will linearize the system dynamics around the nominal trajectory, not the target trajectory!

Let a deviation from the nominal trajectory be given by $\delta x_k = x_k - x_k^{(i)}$ and $\delta u_k = u_k - u_k^{(i)}$. Then we have

$$\begin{aligned} x_{k+1} = f_k(x_k^{(i)}, u_k^{(i)}) &+ D_x f_k(x_k^{(i)}, u_k^{(i)})\delta x_k \\ &+ D_u f_k(x_k^{(i)}, u_k^{(i)})\delta u_k \end{aligned} \tag{D.50}$$

Now, we subtract the *target* state from both sides of the above equation

$$\begin{aligned} x_{k+1} - x_{k+1}^* = f_k(x_k^{(i)}, u_k^{(i)}) &- x_{k+1}^* \\ &+ D_x f_k(x_k^{(i)}, u_k^{(i)})\delta x_k \\ &+ D_u f_k(x_k^{(i)}, u_k^{(i)})\delta u_k \end{aligned} \tag{D.51}$$

To massage this into a linear equation format, we can re-write the above equation as

$$\begin{aligned}x_{k+1} - x^*_{k+1} = f_k(x^{(i)}_k, u^{(i)}_k) &- x^*_{k+1} \\ &+ D_x f_k(x^{(i)}_k, u^{(i)}_k)(x_k - x^*_k) \\ &+ D_x f_k(x^{(i)}_k, u^{(i)}_k)(x^*_k - x^{(i)}_k) \\ &+ D_u f_k(x^{(i)}_k, u^{(i)}_k)(u_k - u^*_k) \\ &+ D_u f_k(x^{(i)}_k, u^{(i)}_k)(u^*_k - u^{(i)}_k)\end{aligned} \tag{D.52}$$

Now, a change in variables will bring us to the standard LQR form, let $z_k = [(x_k - x^*_k) \quad 1]^T$ and $w_k = u_k - u^*_k$. Then we have

$$z_{k+1} = A_k z_k + B_k w_k \tag{D.53}$$

where the upper left block of A_k is given by

$$D_x f_k(x^{(i)}_k, u^{(i)}_k)$$

the upper right block of A_k is

$$f_k(\cdot) - x^*_{k+1} + D_x f_k(\cdot)(x^*_k - x^{(i)}_k) + D_u f_k(\cdot)(u^*_k - u^{(i)}_k)$$

the lower left block of A_k is 0 and the lower right is 1.

References

[1] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Autonomous Robots*, vol. 28, pp. 369–383, 2010, 10.1007/s10514-009-9170-7. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9170-7

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning.* New York, NY, USA: ACM, 2004, p. 1.

[3] N. Ratliff, J. A. D. Bagnell, and M. Zinkevich, "Maximum margin planning," in *International Conference on Machine Learning*, July 2006.

[4] T. Bretl and Z. McCarthy, "Equilibrium configurations of a kirchhoff elastic rod under quasi-static manipulation," in *WAFR*, 2012.

[5] A. Keshavarz, Y. Wang, , and S. Boyd, "Imputing a convex objective function," To appear in the Proceedings of the IEEE Multi-Conference on Systems and Control, 2011.

[6] H. K. Khalil, *Nonlinear Systems.* Prentice-Hall, Inc., 2002.

[7] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction.* Princeton University Press, 2011.

[8] R. E. Kalman, "When is a linear control system optimal?" *Transactions of the ASME, Journal of Basic Engineering*, vol. 86, pp. 51–60, 1964.

[9] A. Jameson and E. Kreindler, "Inverse problem of linear optimal control," *SIAM Journal of Control*, vol. 11, no. 1, pp. 1–19, 1973. [Online]. Available: http://dx.doi.org/doi/10.1137/0311001

[10] B. Molinari, "The stable regulator problem and its inverse," *IEEE Transactions on Automatic Control*, vol. 18, no. 5, pp. 454 – 459, oct 1973.

[11] J. L. Willems and H. van de Voorde, "The return difference for discrete-time optimal feedback systems," *Automatica*, vol. 14, no. 5, pp. 511 – 513, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109878900110

[12] T. Fujii and M. Narazaki, "A complete optimality condition in the inverse problem of optimal control," *SIAM Journal on Control and Optimization*, vol. 22, no. 2, pp. 327–341, 1984. [Online]. Available: http://dx.doi.org/doi/10.1137/0322022

[13] F. Thau, "On the inverse optimum control problem for a class of nonlinear autonomous systems," *IEEE Transactions on Automatic Control*, vol. 12, no. 6, pp. 674 – 681, dec 1967.

[14] E. Kreindler and A. Jameson, "Optimality of linear control systems," *Automatic Control, IEEE Transactions on*, vol. 17, no. 3, pp. 349 – 351, jun 1972.

[15] Y. Niho and J. H. Makin, "A solution to the inverse problem of optimal control: Note," *Journal of Money, Credit and Banking*, vol. 10, no. 3, pp. pp. 371–377, 1978. [Online]. Available: http://www.jstor.org/stable/1991515

[16] J. Casti, "On the general inverse problem of optimal control theory," *Journal of Optimization Theory and Applications*, vol. 32, no. 4, pp. 491–497, 12 1980. [Online]. Available: http://dx.doi.org/10.1007/BF00934036

[17] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods.* Prentice-Hall, Inc., 1990.

[18] T. Shimomura and T. Fujii, "Strictly positive real h2 controller synthesis from the viewpoint of the inverse problem," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, dec 1997, pp. 1014 –1019 vol.2.

[19] W. Rugh, "On an inverse optimal control problem," *IEEE Transactions on Automatic Control*, vol. 16, no. 1, pp. 87 – 88, feb 1971.

[20] P. Moylan and B. Anderson, "Nonlinear regulator theory and an inverse optimal control problem," *Automatic Control, IEEE Transactions on*, vol. 18, no. 5, pp. 460 – 465, oct 1973.

[21] M. Krstic and P. Tsiotras, "Inverse optimal stabilization of a rigid spacecraft," *IEEE Transactions on Automatic Control*, vol. 44, no. 5, pp. 1042 –1049, may 1999.

[22] W. Li, E. Todorov, and D. Liu, "Inverse optimality design for biological movement systems," in *IFAC*, 2011.

[23] H. Deng and M. Krstic, "Stochastic nonlinear stabilization — ii: Inverse optimality," *Systems and Control Letters*, vol. 32, no. 3, pp. 151 – 159, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167691197000674

[24] C. Tang and T. Basar, "Inverse optimal controller design for strict-feedback stochastic systems with exponential-of-integral cost," in *Proceedings of the 15th IFAC World Congress*, 2002.

[25] K. Lenz, P. Khargonekar, and J. Doyle, "When is a controller H_∞-optimal?" *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 1, pp. 107–122, 1988, 10.1007/BF02551404. [Online]. Available: http://dx.doi.org/10.1007/BF02551404

[26] T. Fujii and P. Khargonekar, "Inverse problems in h-infinity control theory and linear-quadratic differential games," in *Proceedings of the 27th IEEE Conference on Decision and Control*, dec 1988, pp. 26 –31 vol.1.

[27] M. Kogan, "Solution to the inverse problem of minimax control and worst case disturbance for linear continuous-time systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 5, pp. 670 –674, may 1998.

[28] D. Alazard, O. Voinot, and P. Apkarian, "A new approach to multi-objective control design from the viewpoint of the inverse optimal control problem," in *IFAC Symposium on System Structure and Control*, 2004.

[29] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 11–73, 1997.

[30] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proceedings of the International Conference on Machine Learning*, 1997.

[31] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems*, 1997.

[32] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, pp. 49–60, 2002, 10.1023/A:1015727715131. [Online]. Available: http://dx.doi.org/10.1023/A:1015727715131

[33] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003. [Online]. Available: http://rstb.royalsocietypublishing.org/content/358/1431/537.abstract

[34] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin / Heidelberg, 2005, vol. 15, pp. 561–572. [Online]. Available: http://dx.doi.org/10.1007/11008941_60

[35] A. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2001, pp. 752 –757 vol.2.

[36] ——, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1398–1403.

[37] F. Guenter and A. Billard, "Using reinforcement learning to adapt an imitation task," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-nov. 2 2007, pp. 1022 –1027.

[38] J. L. Yepes, I. Hwang, and M. Rotea, "New algorithms for aircraft intent inference and trajectory prediction," *Journal of Guidance Control and Dynamics*, vol. 30, pp. 370–382, 2007.

[39] D. Grollman and O. Jenkins, "Sparse incremental learning for interactive robot control policy estimation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, may 2008, pp. 3315 –3320.

[40] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 763 –768.

[41] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 797 –803.

[42] D. H. Grollman and O. C. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," in *International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010.

[43] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen, "Constructing skill trees for reinforcement learning agents from demonstration trajectories," in *Advances In Neural Information Processing Systems*, 2010.

[44] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.

[45] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *IEEE International Symposium on Intelligent Control (ISIC)*, sept. 2011, pp. 613 –619.

[46] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl, "A convex approach to inverse optimal control and its application to modeling human locomotion," in *IEEE International Conference on Robotics and Automation*, 2012.

[47] A. Terekhov, Y. Pesin, X. Niu, M. Latash, and V. Zatsiorsky, "An analytical approach to the problem of inverse optimization with additive objective functions: an application to human prehension," *Journal of Mathematical Biology*, vol. 61, pp. 423–453, 2010, 10.1007/s00285-009-0306-3. [Online]. Available: http://dx.doi.org/10.1007/s00285-009-0306-3

[48] A. Terekhov and V. Zatsiorsky, "Analytical and numerical analysis of inverse optimization problems: conditions of uniqueness and computational methods," *Biological Cybernetics*, vol. 104, pp. 75–93, 2011, 10.1007/s00422-011-0421-2. [Online]. Available: http://dx.doi.org/10.1007/s00422-011-0421-2

[49] J. Park, V. M. Zatsiorsky, and M. L. Latash, "Finger coordination under artificial changes in finger strength feedback: A study using analytical inverse optimization," *Journal of Motor Behavior*, vol. 43, no. 3, pp. 229–235, 2011. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00222895.2011.568990

[50] T. D. Nielsen and F. V. Jensen, "Learning a decision maker's utility function from (possibly) inconsistent behavior," *Artificial Intelligence*, vol. 160, pp. 53–78, 2004.

[51] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. [Online]. Available: http://portal.acm.org/citation.cfm?id=645529.657801

[52] P. Abbeel, “Apprenticeship learning and reinforcement learning with application to robotic control,” Ph.D. dissertation, Stanford University, 2008.

[53] P. Abbeel, A. Coates, and A. Y. Ng, “Autonomous Helicopter Aerobatics through Apprenticeship Learning,” *The International Journal of Robotics Research*, 2010. [Online]. Available: http://ijr.sagepub.com/content/early/2010/06/16/0278364910371999.abstract

[54] J. Tang, A. Singh, N. Goehausen, and P. Abbeel, “Parameterized maneuver learning for autonomous helicopter flight,” in *IEEE International Conference on Robotics and Automation*, may. 2010, pp. 1142 –1148.

[55] U. Syed, M. Bowling, and R. E. Schapire, “Apprenticeship learning using linear programming,” in *Proceedings of the 25th international conference on Machine learning*, ser. ICML ’08. New York, NY, USA: ACM, 2008, pp. 1032–1039. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390286

[56] D. Ramachandran and E. Amir, “Bayesian inverse reinforcement learning,” in *Proceedings of the 20th international joint conference on Artifical intelligence*, ser. IJCAI’07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591. [Online]. Available: http://dl.acm.org/citation.cfm?id=1625275.1625692

[57] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Proc. AAAI*, 2008, pp. 1433–1438.

[58] ——, “Human behavior modeling with maximum entropy inverse optimal control,” *AAAI Spring Symposium on Human Behavior Modeling*, 2009.

[59] K. Dvijotham and E. Todorov, “Inverse optimal control with linearly-solvable mdps,” in *International Conference on Machine Learning*, 2010.

[60] N. Aghasadeghi and T. Bretl, “Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, sept. 2011, pp. 1561 –1566.

[61] S. Javdani, S. Tandon, J. Tang, J. O’Brien, and P. Abbeel, “Modeling and perception of deformable one-dimensional objects,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 1607 –1614.

[62] A. V. Rao, D. A. Benson, C. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington, "Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Trans. Math. Softw.*, vol. 37, no. 2, pp. 22:1–22:39, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1731022.1731032

[63] M. Athans and P. L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications.* McGraw-Hill, 1966.

[64] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control.* Hemisphere Publishing Co., 1975.

[65] R. Levien, "The elastica: A mathematical history," *Technical Report UCB/EECS-2008-103*, vol. EECS Department, University of California, Berkeley, 2008.

[66] I. Todhunter and K. Pearson, *A history of the theory of elasticity and the strength of materials from Galilei to Lord Kelvin.* Cambridge University Press, 1893.

[67] M. Born, "Untersuchungen über die stabilität der elastischen linie in ebene und raum, under verschiedenen grenzbedingungen," Ph.D. dissertation, University of Göttingen, 1906.

[68] S. S. Antman, *Nonlinear Problems of Elasticity*, ser. Applied Mathematical Sciences. New York: Springer, 2005, vol. 107.

[69] A. E. H. Love, *A Treatise on the Mathematical Theory of Elasticity.* Dover Publications, 1944.

[70] R. E. Caflisch and J. H. Maddocks, "Nonlinear dynamical theory of the elastica," *Procrceedings of the Royal Society of Edinburgh: Section A Mathematics*, vol. 99, no. 1-2, pp. 1–23, 1984.

[71] F. Bertails, B. Audoly, B. Querleux, F. Leroy, J. L. L. evêque, and M. P. Cani, "Predicting natural hair shapes by solving the statics of flexible rods," *Eurographics*, 2005.

[72] A. Goriely and S. Neukirch, "Mechanics of climbing and attachment in twining plants," *Physical Review Letters*, vol. 97, no. 18, p. 184302, 2006.

[73] B. Audoly and A. Boudaoud, "'ruban à godets': An elastic model for ripples in plant leaves," *C. R. Mecanique*, vol. 330, pp. 831–836, 2002.

[74] J. E. Hearst and Y. Shi, "New solutions for the stationary states of the elastic rod model are useful in the representation of dna configurations in the living cell," *Nonlinear Science Today*, vol. 5, no. 1, 1995.

[75] Y. Shi and J. Hearst., “The kirchhoff elastic rod, the nonlinear schrd̈ingier equation, and dna supercoiling,” *Journal of Chemical Physics*, vol. 101, no. 6, pp. 5186–5200, 1994.

[76] D. Swigon, B. Coleman, and I. Tobias, “The elastic rod model for dna and its applications to the tertiary structure of dna minicircles in mononucleosomes,” *Biophysics Journal*, vol. 78, pp. 2515–2530, 1998.

[77] S. Goyal, N. C. Perkins, , and C. L. Lee, “Nonlinear dynamics and loop formation in kirchhoff rods with implications to the mechanics of dna and cables,” *Journal of Computational Physics*, vol. 209, pp. 371–389, 2005.

[78] S. T. Santillan and L. N. Virgin, “Numerical and experimental analysis of the static behavior of highly deformed risers,” *Ocean Engineering*, vol. 38, pp. 1397–1402, 2011.

[79] P. K. Agnihotri and S. Basu, “Single-walled nanotubes as kirchhoff elasticas,” *International Journal of Applied Mechanics*, vol. 2, no. 4, pp. 719–743, 2005.

[80] E. Mockensturm and A. Mahdavi, “Van der waal’s elastica,” *ASME International Mechanical Engineering Congress and Exposition*, 2005.

[81] R. H. Plaut, A. D. Borum, and D. A. Dillard, “Analysis of carbon nanotubes and graphene nanoribbons with folded racket shapes,” *ASME Journal of Engineering Materials and Technology*, vol. 134, no. 2, p. 021009, 2012.

[82] S. Ryan, D. Golovaty, and J. P. Wilber, “An elastica model of the buckling of a nanoscale sheet perpendicular to a rigid substrate,” *International Journal of Solids and Structures*, vol. 49, pp. 3681–3692, 2012.

[83] F. Lamiraux and L. E. Kavraki, “Planning paths for elastic objects under manipulation constraints,” *International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[84] M. Moll and L. E. Kavraki, “Path planning for deformable linear objects,” *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 625–636, 2006.

[85] T. W. Bretl and Z. McCarthy, “Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations,” *The International Journal of Robotics Research*, 2013. [Online]. Available: http://ijr.sagepub.com/content/early/2013/06/13/0278364912473169.abstract

[86] J. Biggs, W. Holderbaum, and V. Jurdjevic, "Singularities of optimal control problems on some 6-d lie groups," *IEEE Trans. Autom. Control*, vol. 52, no. 6, pp. 1027–1038, June 2007.

[87] G. Walsh, R. Montgomery, and S. Sastry, "Optimal path planning on matrix lie groups," in *IEEE Conference on Decision and Control*, vol. 2, Dec. 1994, pp. 1258 –1263 vol.2.

[88] D. Matthews and T. Bretl, "Experiments in quasi-static manipulation of a planar elastic rod," *IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, 2012.

[89] K. Gopalakrishnan, K. Goldberg, G. Bone, M. Zaluzec, R. Koganti, R. Pearson, and P. Deneszczuk, "Unilateral fixtures for sheet-metal parts with holes," *IEEE Transactions on Automation Science and Engineering*, vol. 1, no. 2, pp. 110–120, 2004.

[90] K. Kosuge, H. Yoshida, T. Fukuda, M. Sakai, and K. Kanitani, "Manipulation of a flexible object by dual manipulators," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 318–323, 1995.

[91] W. Nguyen and J. Mills, "Multi-robot control for flexible fixtureless assembly of flexible sheet metal auto body parts," *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2340–2345, 1996.

[92] A. Angerer, C. Ehinger, A. Hoffman, W. Reif, G. Reinhart, and G. Strasser, "Automated cutting and handling of carbon fiber fabrics in aerospace industries," *IEEE International Conference on Robotics and Automation*, pp. 2713–2718, 2011.

[93] A. Angerer, C. Ehinger, A. Hoffman, W. Reif, and G. Reinhart, "Design of an automation system for performing processes in aerospace industries," *IEEE Conference on Automation Science and Engineering*, pp. 557–562, 2011.

[94] X. Jiang, K. M. Koo, K. Kikuchi, A. Konno, and M. Uchiyama, "Robotized assembly of a wire harness in a car production line," *Advanced Robotics*, vol. 25, no. 3-4, pp. 473–489, 2011.

[95] Y. Asano, H. Wakamatsu, E. Morinaga, E. Arai, and S. Hirai, "Deformation path planning for manipulation of flexible circuit boards," in *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, Oct. 2010, pp. 5386 –5391.

[96] H. Wakamatsu, T. Yamasaki, E. Arai, and S. Hirai, "Modeling of flexible belt objects toward their manipulation," *IEEE International Symposium on Assembly and Manufacturing*, pp. 1–6, 2007.

[97] H. Wakamatsu, E. Morinaga, E. Arai, and S. Hirai, “Deformation modeling of belt object with angles,” *International Conference on Robotics and Automation*, pp. 606–611, 2009.

[98] ——, “Path planning for belt object manipulation,” *IEEE International Conference on Robotics and Automation*, pp. 4334–4339, 2012.

[99] M. Bell and D. Balkcom, “Knot tying with single piece fixtures,” in *Int. Conf. Rob. Aut.*, May 2008, pp. 379–384.

[100] J. E. Hopcroft, J. K. Kearney, and D. B. Krafft, “A case study of flexible object manipulation,” *The International Journal of Robotics Research*, vol. 10, no. 1, pp. 41–50, 1991.

[101] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi, “Representation for knot-tying tasks,” *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 65 – 78, Feb. 2006.

[102] M. Saha and P. Isto, “Manipulation planning for deformable linear objects,” *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1141–1150, Dec. 2007.

[103] H. Wakamatsu, E. Arai, and S. Hirai, “Knotting/unknotting manipulation of deformable linear objects,” *International Journal of Robotics Research*, vol. 24, no. 4, pp. 371–395, 2006.

[104] R. Jansen, K. Hauser, N. Chentanez, F. van der Stappen, and K. Goldberg, “Surgical retraction of non-uniform deformable layers of tissue: 2d robot grasping and path planning,” in *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, Oct. 2009, pp. 4092 –4097.

[105] N. Chentanez, R. Alterovitz, D. Ritchie, J. Cho, K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O’Brian, “Interactive simulation of surgical needle insertion and steering,” *ACM Transactions on Graphics*, vol. 28, no. 88, pp. 1–10, 2009.

[106] H. Inoue and H. Inaba, “Hand-eye coordination in rope handling,” in *Robotics Research: The First International Symposium*, 1985, pp. 163–174.

[107] J. van den Berg, S. Patil, R. Alterovitz, P. Abbeel, and K. Goldberg, “Lqg-based planning, sensing, and control of steerable needles,” in *Algorithmic Foundations of Robotics IX*, ser. Springer Tracts in Advanced Robotics, D. Hsu, V. Isler, J.-C. Latombe, and M. Lin, Eds. Springer Berlin / Heidelberg, 2011, vol. 68, pp. 373–389.

[108] Y. Yamakawa, A. Namiki, and M. Ishikawa, “Motion planning for dynamic folding of a cloth with two high-speed robot hands and two high-speed sliders,” in *Int. Conf. Rob. Aut.*, May 2011, pp. 5486–5491.

[109] N. M. Amato and G. Song, “Using motion planning to study protein folding pathways,” *Journal of Computational Biology*, vol. 9, no. 2, pp. 149–168, 2002.

[110] G. S. Chirikjian and J. W. Burdick, “The kinematics of hyper-redundant robot locomotion,” *IEEE Trans. Robot. Autom.*, vol. 11, no. 6, pp. 781–793, Dec. 1995.

[111] D. C. Rucker, R. J. Webster, G. S. Chirikjian, and N. J. Cowan, “Equilibrium conformations of concentric-tube continuum robots,” *The International Journal of Robotics Research*, vol. 29, no. 10, pp. 1263–1280, 09 2010.

[112] R. J. Webster and B. A. Jones, “Design and kinematic modeling of constant curvature continuum robots: A review,” *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 11 2010.

[113] H. Tanner, “Mobile manipulation of flexible objects under deformation constraints,” *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 179–184, Feb. 2006.

[114] M. Bergou, M. Wardetzky, S. Robin, B. Audoly, and E. Grinspun, “Discrete elastic rods,” *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–12, 2008.

[115] D. J. Balkcom and M. T. Mason, “Robotic origami folding,” *International Journal of Robotics Research*, vol. 27, no. 5, pp. 613–627, 2008.

[116] C. Tomlin, “Splining on lie groups,” 1995.

[117] F. C. Park and B. Ravani, “Smooth invariant interpolation of rotations,” *ACM Trans. Graph.*, vol. 16, no. 3, pp. 277–295, Jul. 1997. [Online]. Available: http://doi.acm.org/10.1145/256157.256160

[118] C. Altafini, “The de casteljau algorithm on se(3),” in *Nonlinear control in the Year 2000*, ser. Lecture Notes in Control and Information Sciences, A. Isidori, F. Lamnabhi-Lagarrigue, and W. Respondek, Eds. Springer Berlin / Heidelberg, 2000, vol. 258, pp. 23–34, 10.1007/BFb0110205. [Online]. Available: http://dx.doi.org/10.1007/BFb0110205

[119] C. Belta and V. Kumar, “An svd-based projection method for interpolation on se(3),” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 3, pp. 334 –345, jun 2002.

[120] N. E. Dowling, *Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue.* Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

[121] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "An optimality principle governing human walking," *Robotics, IEEE Transactions on*, vol. 24, no. 1, pp. 5 –14, feb. 2008.

[122] R. Ritz, M. Hehn, S. Lupashin, and R. D'Andrea, "Quadrocopter performance benchmarking using optimal control," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 5179–5186.

[123] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 1642 –1648.

[124] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole Polytechnique Federale De Lausanne, 2007.

[125] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *In the Proceedings of the 2007 AIAA Conference on Guidance, Control, and Navigation*, August 2007.

[126] I. Cowling, "Towards autonomy of a quadrotor uav," Ph.D. dissertation, Cranfield University, 2008.

[127] D. Mellinger, M. Shomin, and V. Kumar, "Control of quadrotors for robust perching and landing," in *Proceedings of the International Powered Lift Conference*, Oct 2010.

[128] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011.

[129] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[130] T. Park and S. Levine, "Inverse optimal control for humanoid locomotion," in *Robotics Science and Systems Workshop on Inverse Optimal Control and Robotic Learning from Demonstration*, 2013.

[131] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682 – 697, 2008, robotics and Neuroscience. [Online]. Available: http://www.sciencedirect.com/science/article/B6T08-4SCDB0T-2/2/cbce139eb572cc3825906881cf766c9b

[132] J. Z. Kolter and A. Y. Ng, "Policy search via the signed derivative," in *Robotics: Science and Systems*, 2009.

[133] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2520–2525.

[134] S. Boyd. (2009) EE363: Linear Dynamical Systems. [Online]. Available: http://www.stanford.edu/class/ee363/index.html

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

[136] W. M. Wonham, "On a matrix riccati equation of stochastic control," *SIAM Journal on Control*, vol. 6, no. 4, pp. 681–697, 1968.

[137] L. Dieci and T. Eirola, "Positive definiteness in the numerical solution of riccati differential equations," *Numerische Mathematik*, vol. 67, pp. 303–313, 1994, 10.1007/s002110050030. [Online]. Available: http://dx.doi.org/10.1007/s002110050030

[138] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *International Conference on Informatics in Control, Automation and Robotics*, 2004, pp. 222–229.

[139] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., 1970.

[140] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference*, vol. 1, 2005, pp. 300–306.

[141] Y. Tassa, T. Erez, and B. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems (NIPS 2007)*, 2007, pp. 1465–1472.

[142] E. Todorov and Y. Tassa, "Iterative local dynamic programming," in *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

[143] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning*, vol. 148, 2006, pp. 1–8.

[144] E. F. Morales and C. Sammut, "Learning to fly by combining reinforcement learning with behavioural cloning," in *International Conference on Machine Learning*, 2004.