Forward and Inverse Methods in Optimal Control and Dynamic Game Theory

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Chaitanya Awasthi

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Andrew Lamperski and Rajesh Rajamani

August, 2019

Acknowledgements

I am grateful to a lot of people at the University of Minnesota. I am particularly thankful to my adviser, Prof. Andrew Lamperski, for his patience and guidance over the years. I am also very thankful to my co-adviser, Prof. Rajesh Rajamani, and my committee member Prof. Timothy Kowalewski.

A big thank you also goes to John Gardner, who surely missed an opportunity to write a book from all the questions I have asked him over the years. Thank you to Prof. Chris Hogan as well for helping me whenever I was in need.

I would also like to thank the members of my present home, Medical Robotics and Devices Lab, for their companionship and support while I was writing my thesis.

Finally, I wouldn't be here if it weren't for my parents, Shri Tej Shankar Awasthi and Smt. Chetna Awasthi, and their unconditional love and support. To them, I thank from the bottom of my heart. Everything that I am or may become, I owe it to them.

Thank you to each and everyone of you!

Dedication

To Chanakya and Plato, two of my favorite teachers from antiquity.

Abstract

Optimal control theory is ubiquitous in mathematical sciences and engineering. However, in a classroom setting we barely move beyond linear quadratic regulator problems, if at all. In this work, we demystify the necessary conditions of optimality associated with nonlinear optimal control by deriving them from first principles. We also present two numerical schemes for solving these problems. Moving forward, we present an extension of inverse optimal control, which is the problem of computing a cost function with respect to which observed state and control trajectories are optimal. This extension helps us to handle systems which are subjected to state and/or control constraints. We then generalize the methodology of optimal control theory to solve constrained non-zero sum dynamic games. Dynamic games are optimization problems involving several players who are trying to optimize their respective cost functions subject to constraints. We present a novel method to compute Nash equilibrium associated with a game by combining aspects from direct and indirect methods of solving optimal control problems. Finally, we study constrained inverse dynamic games, which is a problem analogous to constrained inverse optimal control method. Here, we show that an inverse dynamic game problem can be decoupled and solved as an inverse optimal control problem for each of the players individually. Throughout the work, examples are provided to demonstrate efficacy of the methods developed.

Contents

List of Tables

List of Figures

List of Abbreviations

3D	3 Dimensional
BVP	Boundary Value Problem
DOF	Degree of Freedom
DRE	Differential Riccati Equation
FK	Forward Kinematics
IK	Inverse Kinematics
IOC	Inverse Optimal Control
IVP	Initial Value Problem
KKT	Karush-Kuhn-Tucker
LQR	Linear Quadratic Regulator
NLP	Nonlinear Program
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PS	Pseudospectral method
SQP	Sequential Quadratic Programming
TPBVP	Two Point Boundary Value Problem

Chapter 1

Introduction

In this introductory chapter, we present the motivation behind our work, the objectives of this work including original contributions of the author, and the outline of the work.

1.1 Motivation

Optimal control theory as developed in early part of the 20^{th} century has seen an enormous progress in areas as diverse as space exploration [1,2], chemical reactors [3,4], vehicle navigation [5], among others. Several numerical methods [6], such as direct methods [7,8], indirect methods [9, 10], dynamic programming [11, 12], etc., have emerged over the years to accurately solve optimal control problems. Although the algorithms in use are plenty, a look at the published literature can easily intimidate a beginner. There is therefore a clear need to completely spell out these algorithms in such a way that a novice can be able to read the work and apply to their own research.

Inverse optimal control (IOC), as the name suggests, is the inverse problem of the optimal control theory. Whereas a forward problem (or, just an optimal control problem) is that of generating optimal state and control trajectories by minimizing a certain cost function subjected to constraints, the inverse problem is that of trying to infer an underlying cost function given optimal (usually locally optimal) control and state trajectory data. The applications of IOC have enormous potential in the field of robot learning, autonomous navigation among others [13–15]. While some methods exist to solve IOC problems, it is either the case that they use

techniques from machine learning and do not take advantage of elegant mathematical framework provided by control theory [16], or they deal with only linear systems [17], or they involve only unconstrained systems [18], among other things. Therefore, we need a control theoretic method to address these shortcomings.

Dynamic games can be thought of as an extension to optimal control. While problems in optimal control are based in single objective optimization, dynamic games are typically multi-objective optimization problems with several players in cooperative or non-cooperative setting trying to optimize their respective cost functions [19, 20]. Dynamic games have found applications in defense [21, 22], biological systems [23], among others. Compared to optimal control, less work has been done in the area of dynamic games with a control-theoretic view in mind. While problems have been solved in the case of linear quadratic dynamic games [24, 25], as well as some nonlinear dynamic games [26], solution of constrained non-zero sum nonlinear dynamic games can hardly be found in the literature. This shows a clear need of solution strategies to solve such problems.

Inverse dynamic games are analogous to IOC problems, except multiple players are involved. Of all the three areas mentioned above (i.e., optimal control theory, inverse optimal control, and dynamic games), this area has the least presence in the literature. While some work in inverse differentials games exists such as [27–29], none of them deal with the problem of constrained nonlinear non-zero sum dynamic games. This necessitates a need to address this deficiency.

1.2 Thesis statement

The unifying theme of this work is based in solving constrained problems in forward and inverse optimal control using control-theoretic methods, and then generalizing these methods to solve problems in dynamic and inverse dynamic games.

1.3 Objectives

To accomplish the goal as put forth in the thesis statement, the present work will begin by deriving necessary conditions of optimality for constrained optimal control problems using calculus of variations. As such, these necessary conditions can only be used to solve limited problems in

control theory analytically. In the subsequent chapter, details will be provided for implementation of numerical methods to solve more general nonlinear problems. Use of optimal control methods is important because the solution to optimal control problems will be used to solve constrained inverse optimal control problems. A novel algorithm, based on necessary optimality conditions and residual minimization, will be presented to solve these inverse optimal control problems. Another novel algorithm, based again on necessary optimality conditions, will be presented to solve constrained non-zero sum dynamic games. Finally, it will be shown that an inverse dynamic game problem can be reduced to decoupled problems in residual minimization of individual players, which can then be solved using the method employed for solving inverse optimal control problems.

1.4 Contribution

1.4.1 Major Contribution

- Pedagogical introduction to solving nonlinear optimal control problems using classical collocation method (trapezoidal) and state-of-the-art pseudospectral method (PS method)
- A general nonlinear optimal control solver with capabilities such as:
  - choice between pseudospectral (Legendre) or local discretization (Trapezoidal)
  - nonlinear path constraints on states and/or control variables (equality/inequality)
  - isoperimetric constraints (or, integral constraints)
  - bounds on states and/or control variables
  - running (Lagrange) and terminal (Mayer) costs
  - shortest time problems
  - easy extension to multiphase problems
  - accuracy comparable to popular open source softwares such as PSOPT
- Solution of constrained inverse optimal control problems
- Solution of differential games using semi-direct method
- Solution of inverse differential games

1.4.2 Minor Contribution

- A unified framework for solving constrained initial value problems (IVP), boundary value problems (BVP) and two-point boundary value problems (TPBVP)
- Derivation of robot kinematics and dynamics of a non-planar 3 DOF robot and its control using classical and modern methods

1.5 Outline of Work

The chapters in this work are described below:

Chapter 2 presents relevant literature in forward and inverse optimal control as well as forward and inverse dynamic games. It also includes a description of a 3 DOF robot model, which is a full nonlinear robot model simulated by the author to be used in later chapters.

Chapter 3 presents nonlinear optimal control problem and derives necessary conditions of optimality using calculus of variations. These conditions are then used on an LQR problem to derive the closed-loop optimal controller.

Chapter 4 is presents two numerical methods for solving nonlinear optimal control problems. These methods are described in a pedagogical manner and are applied to three example problems.

Chapter 5 talks about inverse optimal control problem and develops a general framework to solve these problems in the light of corresponding forward problems being constrained. It also provides simulation results to show the efficacy of the method.

Chapter 6 presents constrained dynamic game problem and provides a novel way to compute Nash equilibrium of nonzero-sum games. Simulations are provided to show the method in action.

Chapter 7 presents inverse dynamic game problem and provides a method to solve it using an extension of inverse optimal control method. It also provides simulations to support the theoretical developments.

Chapter 8 provides concluding remarks on the methods developed in the work. It also discusses the current shortcomings of the various methods presented in the work and potential avenues of future work.

Appendix A provides a glossary of major technical terms used throughout the work.

Chapter 2

Literature Review and Background

In this chapter we review important literature in the field of optimal and inverse optimal control, as well as dynamic and inverse dynamic games.

2.1 Related Work

Optimal control problems can be seen as problems in trajectory optimization. There are several ways in which such problems can be formulated [6, 30, 31]. Classical methods in optimal control such as trapezoidal or Hermite-Simpson have been in use for some time and have extensive literature [32, 33]. While pseudospectral methods were first used in the solution of partial differential equations in the 1970s in the works of Orszag et al. [34, 35], pseudospectral methods, as used in optimal control theory, only made their appearance in the mid 1990s in the works of Elnagar et al. [36, 37], Ross et al. [38–40], among others.

Inverse optimal control (IOC), as the name suggests, is the inverse problem whereby we impute a cost function from given observations. Inverse optimal control, and the closely related field of inverse reinforcement learning, have found many applications in areas such as understanding human locomotion [13], control of quadcopters [16], and autonomous navigation [41].

IOC has its early beginnings in the work of Kalman [42]. The method has a very close

analogue in the machine learning community, and is called inverse reinforcement learning. Much work has been done in the field of inverse reinforcement learning since the early 2000s. A max-margin algorithm was presented in [16] to recover an unknown reward function from observations of a Markov decision process. The method was eventually used to autonomously navigate a helicopter using trajectories generated by human expert. Another widely studied technique is the maximum entropy reinforcement learning algorithm from [41]. The algorithm utilizes the principle of maximum entropy to systematically select a policy that is consistent with the observed trajectory data.

Inverse reinforcement learning methods based on likelihood optimization were applied to imitation learning problems in [43]. They applied the method to predict the turn decisions of taxi drivers. Inverse optimal control was used by [13] to learn human motion trajectories. They posed the problem as a bi-level optimization in which they solved the optimal control problem for a given set of parameters in an inner loop and minimized the deviations between computed and observed trajectories in an outer loop. Inverse optimal control based on local approximation of the reward function was examined in [15]. Their method involved optimizing a likelihood function which was derived by modelling expert's behavior using maximum entropy inverse reinforcement learning. This work utilized efficient local approximations of the reward functions, which enabled the method to scale to larger problems than is typically possible for inverse optimal control methods.

More closely related to the present work, [17] developed a method for estimating cost functions based on residual minimization of observations using Karush-Kuhn-Tucker (KKT) conditions. The resulting optimization problem is convex and they successfully used it to learn the underlying cost function. Even in cases in which the forward problem was not convex, the method was able to approximate the true cost function using a convex cost function which was capable of mimicking the observed behavior. The residual minimization approach of [17] was extended in [18]. This work utilized a control-theoretic framework to solve inverse optimal control problems for deterministic nonlinear systems. They demonstrated that control costs could be calculated from a quadratic optimization problem derived from Pontryagin's minimum principle. Inverse optimization for constrained discrete-time problems was studied in [44, 45], also based on minimizing a residual from the KKT conditions. The work in [44] also examined the effects of unmodeled dynamics and uncertain observations and proposed a bounded error approach to

inverse optimization.

Dynamic games arise when multiple agents with differing objectives interact over time [19,24]. Dynamic games have many applications including pursuit-evasion [46], active-defense [47, 48], economics [49], human interaction [50, 51], autonomous driving [52], and the smart grid [53]. Compared to centralized problems, however, less is known for game-theoretic settings. For cooperative problems, a method similar to the max-margin method from [16] was proposed in [54]. The maximum entropy method was extended to Markov games [55]. While these methods have a potential advantage of being model free, they also do not exploit model information when it is available.

Inverse dynamic games are important because they can help provide useful information about the intent of the players based on their actions. This particular area of investigation is still developing with some recent advances that include [27, 28].

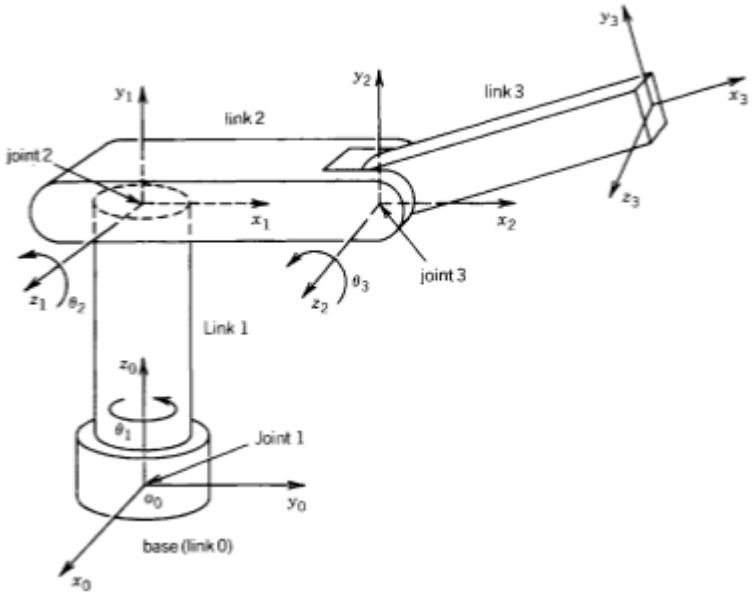


Figure 2.1: A robot manipulator with coordinate frames attached

2.2 Elbow Manipulator

For the purposes of present work, a 3 DOF non-planar robot (as shown in Figure 2.1) will be used as one of the several dynamic systems for simulation studies. This robot is sometimes also called as an elbow manipulator. While all other example problems to be presented in this work are either straightforward to model (using Newtonian or Lagrangian mechanics) or borrowed from literature, this elbow manipulator is completely modeled (in the joint space) by the author and includes modeling of often neglected Coriolis term. The kinematics of the robot is described using Denavit-Hartenberg parameters. This model is thoroughly tested using classical control schemes like PID control for point to point as well as trajectory tracking control. As such, it will serve as one of the most sophisticated dynamic system in this work. The idea for this robot (including Figure 2.1) is borrowed from [56].

Chapter 3

Optimal Control

Optimal control (OC) is a branch of modern control theory in which a sequence of control inputs are generated by solving some optimization problem. These control inputs are then guaranteed to be (locally) optimal with respect to some cost function. In this chapter, necessary conditions of optimality will be derived from first principles using calculus of variations. An application of these necessary conditions is used to derive closed-loop optimal controller for LQR problem.

3.1 Constrained Optimal Control

The problem of optimal control is to find a control law for a given dynamic system such that a certain performance criterion is met. In its mathematical form, the problem therefore is to find the control trajectory $u(t)$, the state trajectory $x(t)$, to minimize the cost function

$$\min_{u} \quad J[u] = \varphi[x(t_f), t_f] + \int_{t_0}^{t_f} L[t, (x(t), u(t)]dt$$

subject to the dynamic constraint,

$$\dot{x}(t) = f[t, x(t), u(t)], t \in [t_0, t_f]$$

path constraint,

$$h_L <= h[t, x(t), u(t)] <= h_U, t \in [t_0, t_f] \tag{3.1}$$

bound constraints,

$$u_L <= u(t) <= u_U, t \in [t_0, t_f]$$

$$x_L <= x(t) <= x_U, t \in [t_0, t_f]$$

and, boundary conditions,

$$\phi[t_0, x(t_0), t_f, x(t_f)] = 0$$

where, formally [57],

– $u \in U$, $x \in X$

– (U, X) are space of admissible functions (for ex., $u \in C^1(\Omega)$, $x \in C^1(\Omega)$, where $C^1(\Omega)$ is the space of continuous functions)

– $\Omega \subset \mathbb{R}^{\mathrm{n}}$, $n \geq 1$ is a bounded open set, a point in Ω is denoted by $t = (t_1, ..., t_n)$

– $u : \Omega \to \mathbb{R}^{\mathrm{n_u}}, n_u \geq 1, u = (u_1, ..., u_{n_u})$

– $x : \Lambda \to \mathbb{R}^{\mathrm{n_x}}, n_x \geq 1, x = (x_1, ..., x_{n_x})$, and hence $\nabla x = \left(\frac{\partial x_j}{\partial t_i}\right)_{1 \leq i \leq n}^{1 \leq j \leq n_x} \in \mathbb{R}^{\mathrm{n_x} \times \mathrm{n}} \quad \therefore \dot{x} = \left(\frac{\partial x_j}{\partial t_i}\right)_{i=1}^{1 \leq j \leq n_x} \in \mathbb{R}^{\mathrm{n_x}}$

– $\varphi : \mathbb{R}^{\mathrm{n_x}} \times \Omega \to \mathbb{R}$

– $L : \Omega \times \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}$

– $f : \Omega \times \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}^{\mathrm{n_x}}$

– $h : \Omega \times \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}^{\mathrm{n_h}}$

– $\phi : \Omega \times \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}^{\mathrm{n}_\phi}$

However, less formally, we use the following simpler definitions

– u : $[t_0, t_f] \rightarrow \mathbb{R}^{n_u}$

– x : $[t_0, t_f] \rightarrow \mathbb{R}^{n_x}$

– $\varphi : \mathbb{R}^{n_x} \times \mathbb{R} \rightarrow \mathbb{R}$

– L : $[t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$

– f : $[t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$

– h : $[t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_h}$

– $\phi : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \rightarrow \mathbb{R}^{n_\phi}$

Here, n_x, n_u, n_h, n_ϕ represent the dimensionality of the state vector, control vector, path constraint vector, and boundary conditions, respectively.

The cost function, J, expressed by (3.1) is said to be in Bolza form. The cost function without the integral term is said to be in Mayer form, and with the integral term alone is said to be in Lagrange form.

3.2 Necessary Conditions of Optimality

3.2.1 Problem Formulation

Let us simplify the very general problem stated in the previous section to only consider free endpoint and fixed final time problems. We also lump the inequality constraints in a single constraint vector $g(t)$. The problem can now be formally stated as:

$$\begin{aligned} \min_u \quad & J(u) = \phi(t_f, x_f) + \int_{t_0}^{t_f} L(t, x(t), u(t))\, dt \\ s.t. \quad & \dot{x}(t) = f(t, x(t), u(t)) \\ & g(t, x(t), u(t)) \leq 0 \\ & x(0) = x_{start} \\ & x(t_f) = x_{free} \end{aligned} \tag{3.2}$$

Notation:

$x(t) \in \mathbb{R}^{n_x}$ is the state vector

$u(t) \in \mathbb{R}^{n_u}$ is the control vector

$J(u)$ is the scalar cost function

$L(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the running cost

$f(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ is the system dynamics

$g(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^s$ is the mixed inequality constraint vector

Assumptions:

We make some assumptions on the optimal control problem for it to be well posed and amenable to the analysis of this paper.

1) $f[t, x(t), u(t)]$ is assumed to be continuous in time and differentiable in state and control variables.

2) $\nabla_x f[t, x(t), u(t)]$ is continuous in both time and control variables.

3) The control variable is at least piecewise continuous in time.

4) The running cost $L[t, x(t), u(t)]$ is differentiable in both state and control variables, and the terminal cost $\phi(t_f, x_f)$ is differentiable in states.

5) The mixed inequality constraint, $g[t, x(t), u(t)]$ is continuous in time, state and control variables and differentiable in state and control variables.

3.2.2 Derivation of Necessary Conditions of Optimality

To discuss the problem of optimal control under mixed inequality constraints, we first define the Hamiltonian, $\mathscr{H}$ and the Lagrangian, $\mathscr{L}$ as follows (for brevity, we will, at times, omit the time dependence of variables in the equations to follow):

$$\mathscr{H}(t, x, u, p) = L(t, x, u) + p^T f(t, x, u) \tag{3.3}$$

$$\mathscr{L}(t, x, u, p, \mu) = \mathscr{H}(t, x, u, p) + \mu^T g(t, x, u) \tag{3.4}$$

where $p \in \mathbb{R}^{n_x}$ is an adjoint variable and $\mu \in \mathbb{R}^k$ is a multiplier. We also define the state-dependent control region as

$$\Lambda(t, x) = \{u \in \mathbb{R}^{n_u} | g(t, x, u) \leq 0\} \subset \mathbb{R}^{n_u} \tag{3.5}$$

We now present some mathematical background which we will make use of in deriving the necessary optimality conditions.

Mathematical Preliminaries

Definition 1 (***Variation***): A variation $\delta x(t)$, in the context of calculus of variations, is defined as difference between perturbed trajectory, $x(t)$ and nominal solution trajectory, $x^*(t)$, i.e.,

$$\delta x(t) = x(t) - x^*(t)$$

Definition 2 (***Increment of a functional***): Given a functional $F(x)$, the increment of a functional, ΔF, is defined as:

$$\begin{aligned}\Delta F &= F(x(t) + \delta x(t)) - F(x(t)) \\ &= \left(\frac{\partial F}{\partial x}\right)\delta x + \frac{1}{2!}\left(\frac{\partial^2 F}{\partial x^2}\right)(\delta x^2) + \dots \\ &= \delta F + \delta^2 F + \dots\end{aligned}$$

Definition 3 (***First variation of a functional***): Given a functional $F(u, v)$, the first variation of the functional, $\delta F(u, v)$, in the neighborhood of (u,v), is the first order term in the increment of a variation $\Delta F(u, v)$:

$$\delta F(u, v) = \frac{\partial F(u, v)}{\partial u}\delta u + \frac{\partial F(u, v)}{\partial v}\delta v$$

We now present a few properties of a variation, a lemma, and a theorem, all without proof, which we would make use of later to derive the necessary optimality conditions.

Properties of a variation

1. $\delta(x(t) + y(t)) = \delta x(t) + \delta y(t)$
2. $\frac{d}{dt}(\delta x(t)) = \delta(\dot{x}(t))$
3. $\int_{t_0}^{t_f} \delta x(t)\, dt = \delta(\int_{t_0}^{t_f} x(t)\, dt)$

Lemma: (**Fundamental Lemma of Calculus of Variations**) If for every continuous function $h(t)$, it is the case that

$$\int_{t_0}^{t_f} h(t)\delta x(t)\, dt = 0$$

where the variation $\delta x(t)$ is continuous in $t \in [t_0, t_f]$, then

$$h(t) = 0 \quad \forall t \in [t_0, t_f]$$

Theorem: (**Lagrange Multiplier Theorem**)

Let x^* be a local minimum of

$$\min_x \ f(x)$$

$$s.t. \quad h(x) = 0$$

Also, assume that the constraint gradients $\nabla h_1(x^*), ..., \nabla h_m(x^*)$ are linearly independent (constraint qualification). Then, there exists a unique vector $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$, called a Lagrange multiplier vector, such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) = 0$$

Although this is a necessary condition of optimality, it is not the only condition. We also need x^* to satisfy the condition $h(x^*) = 0$. The two conditions can be repackaged elegantly in terms of a Lagrangian function defined as

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

The necessary conditions can now be stated compactly as

$$\nabla_x L(x^*, \lambda^*) = 0 \qquad \nabla_\lambda L(x^*, \lambda^*) = 0$$

3.2.3 Necessary Conditions of Optimality

We are now in a position to state a formulation of Pontryagin's minimum principle that is used when dealing with optimal control problems with mixed inequality constraints. The formal statement of the theorem is presented in [58] and we paraphrase and prove its weak form here.

Theorem: Let (x^*, u^*) be an optimal pair for (3.2) over a fixed interval [0,T], such that u^* is right-continuous with left-hand limits and the linear independence constraint qualification (LICQ) holds for every triple {t, x^*(t), u(t)}, t ∈ [0,T] with u ∈ $\Omega(t, x^*(t))$, then there exists a costate trajectory $p(t)$ mapping [0,T] into $\mathbb{R}^{n_x}$, piecewise continuous multiplier function $\mu(t)$ mapping [0,T] into $\mathbb{R}^{n_s}$ and the following conditions hold almost everywhere:

$$u^*(t) = \underset{u \in \Lambda(t, x^*)}{\operatorname{argmin}} \mathscr{H}(t, x^*(t), u, p(t)) \tag{3.6}$$

$$\frac{\partial \mathscr{L}^*}{\partial u}(t, x, u, p(t), \mu(t)) = 0 \tag{3.7}$$

$$\dot{p}(t) = -\frac{\partial \mathscr{L}^*}{\partial x}(t, x, u, p, \mu)$$
$$\left(\frac{\partial \phi}{\partial x} - p\right)\Big|_{t_0}^{t_f} = 0 \tag{3.8}$$
$$\mu(t) \geq 0 \tag{3.9}$$
$$\mu(t)^T g^*(t, x, u) = 0 \tag{3.10}$$

Note: A symbol with an asterisk as a superscript either indicates an optimal variable or indicates evaluation of the symbol with respect to the optimal pair (x^*, u^*).

Proof: The terminal cost $\phi(t_f, x_f)$ in $J(u)$, can be written using Fundamental Theorem of Calculus, as

$$\phi(t_f, x_f) - \cancelto{0}{\phi(t_0, x_0)} = \int_{t_0}^{t_f} \frac{d\phi(t, x)}{dt}\, dt \tag{3.11}$$

where we set $\phi(t_0, x_0)$ to zero, by assumption.

Also, let us add slack variable λ to the inequality constraint and set

$$\overline{g}(t, x, u, \lambda) = g(t, x, u) + \lambda^2 = 0 \tag{3.12}$$

As a result, we can modify our original cost function $J(u)$ to $\overline{J}(u)$ and reformulate (3.2) as:

$$\begin{aligned} \min_u \ \overline{J}(u) &= \int_{t_0}^{t_f} \Big[\frac{d\phi(t, x)}{dt} + L(t, x, u)\Big]\, dt \\ s.t. \qquad & \dot{x}(t) = f(t, x(t), u(t)) \\ & \overline{g}(t, x(t), u(t), \lambda(t)) = 0 \\ & x(0) = x_{start} \\ & x(t_f) = free \end{aligned} \tag{3.13}$$

Using Lagrange Multiplier Theorem, the augmented cost function can be written as

$$J_a(u, p, \mu, \lambda) = \int_{t_0}^{t_f} \Big[\frac{d\phi(t, x)}{dt} + L(t, x, u) + p^T(f(t, x, u) - \dot{x}) + \mu^T \overline{g}(t, x, u, \lambda)\Big]\, dt \tag{3.14}$$

If $(u^*, p^*, \mu^*, \lambda^*)$ is a locally optimal solution to the minimization of (3.14), then it must be the case that in the vicinity of the solution, any small perturbations in (u, p, μ, λ) should satisfy the following condition

$$\begin{aligned} J_a(u^*, p^*, \mu^*, \lambda^*) &\leq J_a(u^* + \Delta u, p^* + \Delta p, \mu^* + \Delta\mu, \lambda^* + \Delta\lambda) \\ &= J_a(u^*, p^*, \mu^*, \lambda^*) + \frac{\partial J_a}{\partial u}\delta u + \frac{\partial J_a}{\partial p}\delta p \\ &\quad + \frac{\partial J_a}{\partial \mu}\delta\mu + \frac{\delta J_a}{\partial \lambda}\delta\lambda + \text{higher order terms} \end{aligned} \tag{3.15}$$

Inequality (3.15) holds if

$$\begin{aligned} \frac{\partial J_a}{\partial u} &= 0 \\ \frac{\partial J_a}{\partial p} &= 0 \\ \frac{\partial J_a}{\partial \mu} &= 0 \\ \frac{\partial J_a}{\partial \lambda} &= 0 \end{aligned} \tag{3.16}$$

(3.16) can be written as first variation in cost function $J_a(u, p, \mu, \lambda)$ as,

$$\delta J_a(u, p, \mu, \lambda) = 0 \tag{3.17}$$

(3.17) is therefore a necessary condition for optimality and will be used in proof of the theorem.

Using the definition of total derivative, (3.14) can be expanded to

$$J_a(u, p, \mu, \lambda) = \int_{t_0}^{t_f} \left\{ \left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x} \right] + L(t, x, u) + p^T(f(t, x, u) - \dot{x}) + \mu^T \overline{g}(t, x, u, \lambda) \right\} dt \tag{3.18}$$

Letting

$$\overline{L}(t, x, \dot{x}, u, p, \mu, \lambda) = \left\{ \left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x} \right] + L(t, x, u) + p^T(f(t, x, u) - \dot{x}) + \mu^T \bar{g}(t, x, u, \lambda) \right\}$$

results in

$$J_a(u, p, \mu, \lambda) = \int_{t_0}^{t_f} \overline{L}(t, x, \dot{x}, u, p, \mu, \lambda)\, dt \tag{3.19}$$

To minimize (3.19), we set its first variation to zero (see (3.17)) to get

$$\delta J_a(u, p, \mu, \lambda) = \delta \int_{t_0}^{t_f} \overline{L}(t, x, \dot{x}, u, p, \mu, \lambda)\, dt = 0 \tag{3.20}$$

Using *Property 3* of a variation,

$$\delta \int_{t_0}^{t_f} \overline{L}(t, x, \dot{x}, u, p, \mu, \lambda)\, dt = \int_{t_0}^{t_f} \delta\overline{L}(t, x, \dot{x}, u, p, \mu, \lambda)\, dt \tag{3.21}$$

Using *Definition 3*, we can rewrite (3.21) as (we subsequently drop the arguments of $\overline{L}$ and other functions for brevity)

$$\int_{t_0}^{t_f} \delta\overline{L}\, dt = \int_{t_0}^{t_f} \left\{ \frac{\partial \overline{L}}{\partial x}\delta x + \frac{\partial \overline{L}}{\partial \dot{x}}\delta\dot{x} + \frac{\partial \overline{L}}{\partial u}\delta u + \frac{\partial \overline{L}}{\partial p}\delta p + \frac{\partial \overline{L}}{\partial \mu}\delta\mu + \frac{\partial \overline{L}}{\partial \lambda}\delta\lambda \right\} dt \tag{3.22}$$

Rewriting the second term in the above eqn. using integration by parts, we get

$$\int_{t_0}^{t_f} \frac{\partial \overline{L}}{\partial \dot{x}}\delta\dot{x}\, dt = \frac{\partial \overline{L}}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f} - \int_{t_0}^{t_f} \frac{d}{dt}\left(\frac{\partial \overline{L}}{\delta \dot{x}}\right)\delta x\, dt \tag{3.23}$$

Substituting the above expression in (3.22), we have

$$\int_{t_0}^{t_f} \delta\overline{L}\, dt = \int_{t_0}^{t_f} \left\{ \left(\frac{\partial \overline{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \overline{L}}{\partial \dot{x}}\right)\right)\delta x + \frac{\partial \overline{L}}{\partial u}\delta u + \frac{\partial \overline{L}}{\partial p}\delta p + \frac{\partial \overline{L}}{\partial \mu}\delta\mu + \frac{\partial \overline{L}}{\partial \lambda}\delta\lambda \right\} dt + \frac{\partial \overline{L}}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f} \tag{3.24}$$

Using the definition of $\overline{L}$ and $\overline{g}$, we expand the above eqn. to get,

$$\begin{aligned}
\int_{t_0}^{t_f} \delta\overline{L}\, dt = \int_{t_0}^{t_f} \Bigg[& \Bigg\{ \Bigg(\cancel{\frac{\partial}{\partial x}\left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x}\right]} + \frac{\partial L}{\partial x} + p^T \frac{\partial f}{\partial x} + \mu^T \frac{\partial g}{\partial x} \Bigg) \\
& - \frac{d}{dt}\Bigg(\cancel{\frac{\partial}{\partial \dot{x}}\left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x}\right]} - p \Bigg) \Bigg\}\delta x + \left\{ \frac{\partial L}{\partial u} + p^T \frac{\partial f}{\partial u} + \mu^T \frac{\partial g}{\partial u} \right\}\delta u \\
& + \Big\{ f - \dot{x} \Big\}\delta p + \Big\{ g + \lambda^2 \Big\}\delta\mu + \Big\{ 2\mu^T \lambda \Big\}\delta\lambda \Bigg]\, dt + \frac{\partial \overline{L}}{\partial \dot{x}}\delta x\Big|_{t_0}^{t_f}
\end{aligned} \tag{3.25}$$

The two terms above canceled because simplification of the first term gives

$$\frac{\partial}{\partial x}\left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x}\right] = \frac{\partial^2 \phi}{\partial x \partial t} + \frac{\partial^2 \phi}{\partial x^2}$$

while simplification of the second term gives

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial}{\partial \dot{x}}\left[\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\dot{x}\right]\right) &= \frac{d}{dt}\left(\frac{\partial \phi}{\partial x}\right) \left[\because \frac{\partial}{\partial \dot{x}}\left(\frac{\partial \phi}{\partial t}\right) = 0\right] \\
&= \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\dot{x}\right) + \frac{\partial}{\partial t}\left(\frac{\partial \phi}{\partial x}\right) \\
&\quad \text{[using definition of total derivative]} \\
&= \frac{\partial^2 \phi}{\partial x^2}\dot{x} + \frac{\partial^2 \phi}{\partial t \partial x}
\end{aligned}$$

which are the same (using Clairaut's theorem).

Simplifying (3.25), we have

$$\begin{aligned}\int_{t_0}^{t_f} \delta\overline{L}\, dt = \int_{t_0}^{t_f} \Bigg[&\left\{ \left(\frac{\partial L}{\partial x} + p^T \frac{\partial f}{\partial x} + \mu^T \frac{\partial g}{\partial x} \right) + \dot{p} \right\} \delta x \\ &+ \left\{ \frac{\partial L}{\partial u} + p^T \frac{\partial f}{\partial u} + \mu^T \frac{\partial g}{\partial u} \right\} \delta u + \left\{ f - \dot{x} \right\} \delta p \\ &+ \left\{ g + \lambda^2 \right\} \delta\mu + \left\{ 2\mu^T \lambda \right\} \delta\lambda \Bigg]\, dt + \frac{\partial \overline{L}}{\partial \dot{x}} \delta x \Big|_{t_0}^{t_f}\end{aligned} \tag{3.26}$$

Since (3.26) is the first variation in cost function $J_a(u, p, \mu, \lambda)$, the entire expression must evaluate to zero.

While we will make use of the fact that the expression under the integral and the non-integral expression are independently equal to zero, we do not prove it here (see [59] for some insight). Thus,

$$\begin{aligned}\int_{t_0}^{t_f} \Bigg[&\left\{ \left(\frac{\partial L}{\partial x} + p^T \frac{\partial f}{\partial x} + \mu^T \frac{\partial g}{\partial x} \right) + \dot{p} \right\} \delta x \\ &+ \left\{ \frac{\partial L}{\partial u} + p^T \frac{\partial f}{\partial u} + \mu^T \frac{\partial g}{\partial u} \right\} \delta u + \left\{ f - \dot{x} \right\} \delta p \\ &+ \left\{ g + \lambda^2 \right\} \delta\mu + \left\{ 2\mu^T \lambda \right\} \delta\lambda \Bigg]\, dt = 0\end{aligned} \tag{3.27}$$

and,

$$\begin{aligned}&\frac{\partial \overline{L}}{\partial \dot{x}} \delta x \Big|_{t_0}^{t_f} = \left(\frac{\partial \phi}{\partial x} - p \right) \Bigg|_{t_f} \delta x(t_f) = 0 \\ &[\because x(t_f) \text{ is free but } x(t_0) \text{ is specified}] \\ &\text{or,} \quad \left(\frac{\partial \phi}{\partial x} - p \right) \Bigg|_{t_f} = 0\end{aligned} \tag{3.28}$$

(3.28) is often referred to as transversality condition.

With reference to (3.27), the constraint

$$\dot{x} = f(t, x, u) \tag{3.29}$$

must be satisfied by the optimal solution so that coefficient of δp is zero. The Lagrange multipliers, p, are arbitrary, so let us select them in such a way to make the coefficient of δx equal to zero, i.e.,

$$\dot{p} = -\frac{\partial L}{\partial x} - p^T \frac{\partial f}{\partial x} - \mu^T \frac{\partial g}{\partial x} \tag{3.30}$$

The remaining variations $\delta u, \delta\mu,$ and $\delta\lambda$ are independent, so, using fundamental lemma of calculus of variations, their coefficients must be independently zero, thus

$$\frac{\partial L}{\partial u} + p^T \frac{\partial f}{\partial u} + \mu^T \frac{\partial g}{\partial u} = 0 \tag{3.31}$$

$$g + \lambda^2 = 0 \tag{3.32}$$

$$2\mu^T \lambda = 0 \tag{3.33}$$

Using (3.32), (3.33) it can be shown that

$$\mu^T g = 0 \tag{3.34}$$

Furthermore, while we are skipping the argument, it can be shown, using (3.30), that

$$\mu(t) \geq 0 \tag{3.35}$$

Using (3.3) and (3.4), (3.30) can be rewritten as

$$\dot{p} = -\frac{\partial \mathscr{L}}{\partial x} \tag{3.36}$$

Similarly, using (3.3) and (3.4), (3.31) can be rewritten as

$$\frac{\partial \mathscr{L}}{\partial u} = 0 \tag{3.37}$$

This completes the proof of the theorem. ■

3.2.4 Complete Set of Necessary Conditions for Optimality

In addition to the conditions derived above, few additional conditions are required for the pair (x,u) to be optimal. We now present the complete set of necessary optimality conditions and classify them as such:

Stationarity

$$\frac{\partial \mathscr{L}^*}{\partial u}(t, x, u, p, \mu) = 0 \tag{N1}$$

$$\dot{p}(t) = -\frac{\partial \mathscr{L}^*}{\partial x}(t, x, u, p, \mu)$$
$$\left(\frac{\partial \phi}{\partial x} - p\right)\Big|_{t_f} = 0 \tag{N2}$$

Primal feasibility

$$\begin{aligned} \dot{x}(t) &= \frac{\partial L^*}{\partial p}(t, x, u, p, \mu) \\ x(0) &= x_0 \end{aligned} \tag{N3}$$

$$g(t, x(t), u(t)) \leq 0 \tag{N4}$$

Dual feasibility

$$\mu(t) \geq 0 \tag{N5}$$

Complementary slackness

$$\mu(t)^T g^*(t, x, u) = 0 \tag{N6}$$

3.3 Linear Quadratic Regulator

Before we begin the study of nonlinear optimal control problems, we will derive important and relevant results for a class of problems known as linear quadratic regulator (LQR). In this special class of optimal control problems, the dynamics of the system are linear and the cost function considered is quadratic in state and control variables. The (unconstrained) problem is thus to find optimal control and state trajectories, $u(t)$ and $x(t)$, respectively, that minimize the cost function

We now apply the necessary optimality conditions derived above to the case where the cost function is quadratic, the dynamics are linear, and there are no additional constraints on the system (like the inequality constraints). The goal of the problem is to regulate the dynamic system.

We show that in the case of LQR, the necessary conditions are reduced to a differential Riccati equation (DRE) and the resulting controller is a unique gain multiplied by the solution of DRE. The problem is formally stated as:

$$\min_u \ \mathrm{J(u)} = \frac{1}{2} \int_{t_0}^{t_f} \left((x(t)^T Q x(t) + u(t)^T R u(t) \right) dt \tag{3.38}$$

subject to the dynamic constraint,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad t \in [t_0, t_f] \\ x(0) &= x_0 \end{aligned} \tag{3.39}$$

where,

Q is a positive semi-definite state weighting matrix,

R is a positive definite control weighting matrix,

N is a cross-coupling matrix between state and control,

A is a system matrix, and

B is a control input matrix.

In this case, the Hamiltonian is given by

$$\mathscr{H} = \frac{1}{2}(x^T Q x + u^T R u + 2x^T N u) + p^T(Ax + Bu) \tag{3.40}$$

Because there are no inequality constraints, $\mathscr{L}$ and $\mathscr{H}$ are the same here (see (3.3), (3.4)). Applying the necessary conditions of optimality, we have

$$\dot{x} = \nabla_p \mathscr{H} = Ax + Bu \tag{3.41}$$

$$\dot{p} = -\nabla_x \mathscr{H} = -(Qx + Nu + A^T p) \tag{3.42}$$

$$0 = \nabla_u \mathscr{H} = Ru + B^T x + B^T p \tag{3.43}$$

Applying the transversality conditions to our problem with fixed final time (i.e. $\delta t_f = 0$) and free end state ($\delta x_f \neq 0$), we get

$$p(t_f) = 0 \tag{3.44}$$

(since $\varphi = 0$ in our case as there is no end point cost).

As we can see, (3.41) returns the original state dynamics and is therefore known as state dynamics equation. Similarly, (3.42) is known as co-state equation and (3.43) is known as optimal control equation. It is important to note that while (3.41) and (3.42) are differential equations, (3.43) is an algebraic equation. Also important to note is that while the state dynamics equation marches forward in time (due to boundary condition, $x(0) = x_0$), the co-state equation marches backward in time (due to boundary condition, $p(t_f) = 0$). As such, the two differential equations constitute a two-point boundary value problem (TPBVP) as opposed an initial value problem (IVP) which is simpler to solve.

Solving for the optimal control input, u(t), using (3.43), we get

$$u = -R^{-1}(N^T x + B^T p) \tag{3.45}$$

However, as it is, u(t) is a function of both the state vector, x and co-state vector, p and we would like to find an expression for u(t) in terms of x alone. Therefore, we guess the following

relationship between x and p. Let

$$p(t) = P(t)x(t) \tag{3.46}$$

where P is some square matrix of size nxn. Since $p(t_f) = 0$, we set $P(t_f) = 0$ at the end time so the equality is satisfied. Differentiating (3.46) with respect to time, we get

$$\begin{aligned}
\dot{p} &= \dot{P}x + P\dot{x} \\
&= \dot{P}x + P(Ax + Bu) && \text{(using (3.41))} \\
&= \dot{P}x + P(Ax - BR^{-1}(N^T x + B^T p)) && \text{(using (3.43))} \\
&= (\dot{P} + PA - PBR^{-1}N^T - PBR^{-1}B^T P)x && \text{(using (3.46))}
\end{aligned} \tag{3.47}$$

Now, substituting (3.45) in (3.43) and setting it equal to LHS of (3.47), we get

$$\begin{aligned}
&-(Qx - NR^{-1}(N^T x + B^T p) + A^T p) = (\dot{P} + PA - PBR^{-1}N^T - PBR^{-1}B^T P)x \\
&\implies -(Q - NR^{-1}N^T - NR^{-1}B^T P + A^T P)x = (\dot{P} + PA - PBR^{-1}N^T - PBR^{-1}B^T P)x \\
&\implies (\dot{P} + PA - PBR^{-1}N^T - PBR^{-1}B^T P + Q - NR^{-1}N^T - NR^{-1}B^T P + A^T P)x = 0
\end{aligned} \tag{3.48}$$

Since x is arbitrary in (3.48), it must be the case that its coefficient is zero. We therefore have the following equality:

$$\dot{P} + PA - PBR^{-1}N^T - PBR^{-1}B^T P + Q - NR^{-1}N^T - NR^{-1}B^T P + A^T P = 0 \tag{3.49}$$

which is known as the differential riccati equation (DRE).

We know have an optimal controller, u(t), that is a solution to out problem of optimal control. Substituting (3.46) in (3.45), we get a linear fullstate feedback controller

$$u(t) = -R^{-1}(N^T + B^T P(t))x(t) \tag{3.50}$$

or,

$$u(t) = -K(t)x(t) \tag{3.51}$$

where $K(t) = R^{-1}(N^T + B^T P(t))$ and P(t) is the solution to the DRE given by (3.48) subject to $P(t_f) = 0$.

While the necessary conditions of optimality are very useful in finding optimal solutions, most of the nonlinear optimal control problems do not admit such analytical solutions. As a result, in the next chapter, we present two very useful and important numerical schemes to compute the solution to such problems.

Chapter 4

Numerical Methods in Optimal Control

In this chapter, we present two numerical methods for solving constrained nonlinear optimal control problems. These methods are then implemented on three example cases.

4.1 Introduction

Let us recall the nonlinear optimal control problem that we are trying to solve:

$$\begin{aligned} \min_{u} \quad & J(u) = \phi(t_f, x_f) + \int_{t_0}^{t_f} L(t, x(t), u(t))\, dt \\ s.t. \qquad & \dot{x}(t) = f(t, x(t), u(t)) \\ & g(t, x(t), u(t)) \leq 0 \\ & x(0) = x_{start} \\ & x(t_f) = x_{free} \end{aligned} \tag{4.1}$$

Before we present the two numerical schemes, we take a brief look at some of the methods that are available to us, as well their shortcomings.

4.2 Methods of Optimal Control

In the previous chapter, we derived an analytical solution to the OCP when the cost function is quadratic and the state dynamics are linear. However, generally speaking, these problems are seldom linear and no analytical expressions exist for optimal controller. Such (nonlinear) problems can be solved using three distinct methods which are:

1. Dynamic programming: This method requires solving the Hamilton-Jacobi-Bellman equation in the entire state space. Typically, the problems are solved within the mathematical framework of Markov decision processes. This method works well for low dimensional problems and generally provides globally optimal solution. However, it does not scale well for higher dimensional systems because it involves discretization of the state space.

2. Indirect method: This method involves solving problems within the mathematical framework of calculus of variations. The problem is solved using Euler-Lagrange equation along with transversality condition to account for boundary conditions. The optimal control itself is found using Pontryagin's minimum principle. The resulting set of equations lead to TPBVP and is then solved numerically. This method is only suitable for simple problems where the control trajectories are smooth and continuous. The solutions resulting from this method usually tend to be numerically unstable. Also, good initial co-state (or, adjoint variable) estimates are required to initialize the problem.

3. Direct method: This method discretizes the original continuous time optimal control problem into a finite dimensional nonlinear programming problem. It then solves it using a nonlinear programming solver such as sequential quadratic programming (SQP), interior-point method, etc. This method is also known as the direct transcription method because it directly transcribes the OCP into an NLP. Due to its better applicability in both high and low dimensional systems, as well as low overhead for implementation, this method is going to be the focus of our work.

4.3 Direct Method

As mentioned earlier, the direct method discretizes an infinite dimensional function optimization problem and converts it into a finite dimensional parameter optimization problem. This step is also known as transcription. The resulting optimization problem is an NLP and is typically solved using optimization solvers such as SQP. Transcription methods generally fall into two categories, namely, shooting methods and collocation methods. The major difference is based on how each method enforces the dynamic constraint. While shooting methods use simulation to explicitly enforce the dynamic constraint, collocation methods enforce the dynamics on a series of points (known as collocation points) along the trajectory. While shooting methods may be easier to implement, they quickly become inadequate when dealing with complicated problems, such as highly nonlinear constraints, etc. Therefore, in this work, we will focus exclusively on collocation methods.

Collocation methods

Collocation methods directly represent state and control trajectory using decision variables, and enforce the constraints (dynamic or static) only at certain specific points along the trajectory. These specific points are what are known as collocation points. Five main steps are involved in order to convert a continuous time optimal control problem into a discrete NLP (transcription), and another two steps are involved once an NLP has been formulated (nonlinear programming). These seven steps are listed as under:

4.3.1 Transcription

1. Choose time grid size and type

The grid type is either chosen to be uniform or non-uniform. Each of the two types have their advantages and disadvantages. For example, while uniform grid is easy and straightforward to implement, it leads to Runge's phenomenon, which is a problem of oscillation at the edges of the (time) interval when using polynomial interpolation. On the other hand, while non-uniform grid can easily overcome Runge's phenomenon, there are many ways to implement it and care must be taken to ensure that function approximation respects the non-uniform nature of grid.

2. Discretize state and control vectors

Once a grid has been chosen, the state and the control trajectories can be discretized on the

same grid. The specific discretization procedure depends on the type of method chosen.

3. Discretize cost function

Discretizing cost function involves approximating the integral cost function (such as Riemann sum approximation). However, this step too depends on the particular transcription method chosen.

4. Discretize and enforce dynamic constraint

Dynamic constraint typically refers to the differential equation that the system needs to satisfy at every node of the grid. Again, this step depends on the chosen transcription method.

5. Discretize and enforce static constraints

Discretizing static constraint is straightforward and one only needs to ensure that the constraint is enforced at every node of the chosen grid.

4.3.2 Nonlinear Programming

6. Solve the NLP

Steps 1-5 listed above were part of the transcription process which converts an infinite dimensional functional optimization problem to a finite dimensional parameter optimization problem. Following transcription, we now have an NLP which can be solved using a number of solvers such as SQP, interior-point method, etc.

7. Interpolate the solution

Solving the NLP results in optimal state and control vectors at the chosen grid points. Depending on the choice of collocation method, this NLP solution is interpolated to generate smooth trajectories between the nodes.

Collocation methods are a family of methods that can use either local or global function approximation, uniform or non-uniform grids, or can have several other distinguishing features, depending upon the choices made in the steps 1-7 mentioned above. In the context of present work, two methods will be implemented, namely trapezoidal collocation method and pseudospectral collocation method. While the former is an easy to implement and easy to understand method, it is less accurate and requires far more (typically uniform) grid points than the latter method, which while not intuitive, is far more accurate and requires much less (non-uniform) grid points.

While there is no sharp distinction among collocation methods, one way in which they

can be divided is orthogonal and non-orthogonal methods. In orthogonal collocation method, orthogonal polynomials are used to approximate state and control trajectories. Orthogonal polynomials have very appealing numerical properties which makes it is easy to use them for performing integration, differentiation and interpolation procedures, procedures which as we have seen in steps 1-7 above, are essential for solution of optimal control problem.

On the other hand, classical methods such as trapezoidal and Hermite-Simpson, are non-orthogonal collocation methods as they use non-orthogonal polynomials for state and control trajectory approximation. All Runge-Kutta schemes typically belong to non-orthogonal collocation methods. For example, constant approximation between grid points leads to Euler's method, linear approximation between grid points leads to trapezoidal method, etc. More advanced methods such as those based on higher order

The two methods that we will present in this work, namely trapezoidal collocation and pseudospectral collocation, are non-orthogonal and orthogonal methods, respectively. It is important to note that while non-uniform grid can be chosen for the trapezoidal method, it is relatively more tedious than its uniform grid based cousin, which is something we want to avoid given trapezoidal rule is only second order accurate anyway. If not for the slow convergence of trapezoidal method, compared to the much faster pseudospectral method, trapezoidal method does quite well on problems which are generally considered difficult.

We now describe the two methods in detail and present a framework for solving optimal control problems using them.

4.4 Trapezoidal Collocation Method

This is a local method and so the discretization of optimal control problem described above involves local approximation between the grid points. In this method, the control trajectory is approximated as a linear spline between grid points and the state trajectory is approximated as a quadratic spline. We now describe in detail the steps involved in implementing the Trapezoidal method for solving optimal control problems [60].

1. Grid type

We will use a uniform grid type for this method which means

$$t \to t_0, ..., t_k, ..., t_N$$

where N is the number of segments and $N + 1$ is the number of nodes in the grid (or, grid points). These grid points become what are known as collocation points.

2. Discretized state and control vectors

Depending on the chosen collocation points, the state and control trajectories are discretized at those points. The discretized trajectories now become decision variables in the optimal control problem.

$$u \to u_0, ..., u_k, ..., u_N$$

$$x \to x_0, ..., x_k, ..., x_N$$

where u_k is a discrete representation of the control trajectory at the k^{th} collocation point. Likewise, x_k is a discrete representation of the state trajectory at the k^{th} collocation point. In other words, $x_k = x(t_k)$ and $u_k = u(t_k)$. u_k and x_k are now finite dimensional vectors of size n_x and n_u, respectively.

3. Integral approximation for cost function

The cost function is approximated using trapezoidal rule to give:

$$J = \phi(t_f, x_f) + \int_{t_0}^{t_f} L(t, x(t), u(t))\ dt \approx \phi(t_k, x_k) + \sum_{k=0}^{N-1} \frac{1}{2} h_k (L_k + L_{k+1}) \tag{4.2}$$

where $h_k = t_{k+1} - t_k = h = \text{constant}$ (because we chose a uniform grid), and $L_k = L(t_k, x_k, u_k)$.

4. Enforcing system dynamics as dynamic constraint

In trapezoidal method, the system dynamics are enforced at each collocation point by integrating the dynamics and approximating the resulting integral using trapezoidal rule as mentioned in the previous step. We therefore have:

$$\dot{x} = f$$

$$\int_{t_k}^{t_{k+1}} \dot{x}\ dt = \int_{t_k}^{t_{k+1}} f\ dt$$

$$x_{k+1} - x_k \approx \frac{1}{2} h (f_{k+1} + f_k)$$

The last equation leads us the constraint which is applied between every pair of collocation points $k \in \{0, ..., N-1\}$

$$x_{k+1} = x_k + \frac{1}{2} h (f_{k+1} + f_k) \tag{4.3}$$

5. Enforce static constraints

The static constraints are simply enforced at all collocation points such that

$$g(t, x(t), u(t)) \leq 0 \quad \rightarrow \quad g(t_k, x_k, u_k) \leq 0, \quad \forall k \in \{0, ..., N\} \tag{4.4}$$

6. Resulting NLP

Following steps 1-5, we now have the following NLP:

$$\begin{aligned} \min_{\substack{u_0,\ldots,u_N \\ x_0,\ldots x_N}} \quad & J = \phi(t_k, x_k) + \sum_{k=0}^{N-1} \frac{1}{2} h_k (L_k + L_{k+1}) \\ s.t. \quad & x_{k+1} = x_k + \frac{1}{2} h (f_{k+1} + f_k), \qquad \forall k \in \{0, ..., N-1\} \\ & g(t_k, x_k, u_k) \leq 0 \qquad\qquad\qquad\quad \forall k \in \{0, ..., N\} \\ & x_0 = x_{start} \\ & x_N = x_{free} \end{aligned} \tag{4.5}$$

This NLP is then solved using SQP solver.

7. Solution interpolation

Because trapezoidal collocation works by approximating control trajectory as varying linearly between the grid points, it is constructed as

$$u(t) \approx u_k + \frac{\tau}{h}(u_{k+1} - u_k) \qquad \text{for every interval}, t \in [t_k, t_{k+1}] \tag{4.6}$$

where $\tau = t - t_k$.

Next, because the dynamics too are approximated as varying linearly between the grid points, this means that the state approximation, which is an integral of the dynamics, is approximated via quadratic splines. This leads to the following construction of the state trajectory

$$x(t) \approx x_k + f_k \tau + \frac{\tau^2}{2h}(f_{k+1} - f_k) \qquad \text{for every interval}, t \in [t_k, t_{k+1}] \tag{4.7}$$

where $\tau = t - t_k$.

4.5 Pseudospectral Collocation Method

Pseudospectral methods are state-of-the-art collocation methods for solving optimal control problems. Unlike, trapezoidal method which used uniform grid and local polynomials for approximating states and controls , this method uses non-uniform grid and global polynomials

for approximating state and control trajectories. Using these methods, the state and control trajectories are approximated as weighted sum of smooth basis functions, such as Legendre or Chebyshev polynomials in the interval $[-1, 1]$. One of the major advantages of this method over classical optimal control methods (such as the trapezoidal method) is the exponential convergence rate of the solution which is faster than any polynomial rate (typical in case of classical methods). Another advantage of this method is its use of relatively small number of grid points to yield solutions high accuracy [38, 61, 62].

Before we provide the sequence of steps needed to implement this method, we take a quick detour to talk about orthogonal polynomials and types of collocation points, two concepts which lie at the heart of pseudospectral methods.

4.5.1 Orthogonal Polynomials

Orthogonal polynomials $P_n(\tau)$ are a class of polynomials defined over an interval $[t_0, t_f]$ such that they obey the following orthogonality condition:

$$\int_{t_0}^{t_f} w(\tau) P_m(\tau) P_n(\tau) d\tau = \begin{cases} 1, & \text{if } m = n \\ 0, & \text{if } m \neq n \end{cases} \tag{4.8}$$

where $w(\tau)$ is a weighting function.

In our method, we will use Legendre polynomials as the choice for orthogonal polynomials. Other polynomials such as Chebyshev polynomials can also be used, which result in a variant of pseudospectral optimal control method. The Legendre polynomials $P_n(\tau)$ are the eigenfunctions of a singular Sturm-Lioville problem:

$$(1 - \tau^2) \frac{d^2 P_n(\tau)}{d\tau^2} - 2\tau \frac{dP_n(\tau)}{d\tau} + n(n+1) P_n(\tau) = 0 \tag{4.9}$$

where $n(n+1) = \lambda$ is generally called eigenvalue of the problem. The differential equation is solved for particular values of λ to yield the eigenfunctions $P_n(\tau)$. The Legendre polynomials are orthogonal over the interval $[-1, 1]$ with weight function $w = 1$. A Legendre polynomial of order N can be generated from:

$$P_N(\tau) = \frac{1}{2^N N!} \frac{d^N}{d\tau^N} (\tau^2 - 1)^N \tag{4.10}$$

4.5.2 Collocation Points

One can also get variants of pseudospectral methods depending on how the collocation points are chosen. If none of the end points of the interval $[-1, 1]$ are included as collocation points, we get Legendre-Gauss (LG) points. If one of the end points of the interval is included as a collocation point, we get Legendre-Gauss-Radau (LGR) points. If both the end points of the interval are included as collocation points, we get Legendre-Gauss-Lobatto (LGL) points [63].

In this work, LGL points (or nodes) will be used. These nodes are the roots of $\dot{P}_N(\tau)$ together with -1 and 1. For example, if $N = 3$, the collocation points $\tau_k(k = 0, ..., N)$ are

$$[\tau_0, \tau_1, \tau_2, \tau_3] = [-1, -0.4472, 0.4472, 1]$$

where τ_1, τ_2 are the roots of the Legendre polynomial $\dot{P}_3(\tau)$.

4.5.3 Lagrange Polynomials

In the case of pseudospectral methods, Lagrange polynomials are used to approximate state and control trajectories via global polynomials. Given a set of $N + 1$ distinct collocation points $\tau_0, ..., \tau_N$ and the value of a function at those points , i.e., $f(\tau_0), ..., f(\tau_N)$, a unique polynomial $P(\tau)$ of degree at most N exists such that

$$P(\tau_k) = f(\tau_k), \quad \forall k \in \{0, ..., N\}$$

This polynomial is defined as

$$P(\tau) = \sum_{k=0}^{N} f(\tau_k) L_k(\tau)$$

where

$$L_k(\tau) = \prod_{i=0, i\neq k}^{N} \frac{\tau - \tau_i}{\tau_k - \tau_i} \tag{4.11}$$

Here $P(\tau)$ is known as the Lagrange polynomial and $L_k(\tau)$ are called Lagrange basis polynomials.

We now describe the steps involved in implementing Legendre pseudospectral optimal control method.

1. Grid type

A non-uniform grid is chosen for collocation. Because Legendre polynomials are orthonormal in the interval $[-1, 1]$, the independent variable $t \in [t_0, t_f]$ is mapped to a new independent

variable $\tau \in [-1, 1]$ via the following affine transformation:

$$\tau \leftarrow \frac{2}{t_f - t_0} t - \frac{t_f + t_0}{t_f - t_0} \tag{4.12}$$

Because of this transformation, the original OCP is transformed to the following new OCP:

$$\begin{aligned} \min_{u} \quad & J(u) = \phi(t_f, x(1)) + \frac{t_f - t_0}{2} \int_{-1}^{1} L(\tau, x(\tau), u(\tau))\, d\tau \\ s.t. \qquad & \dot{x}(\tau) = \frac{t_f - t_0}{2} f(\tau, x(\tau), u(\tau)), \quad \tau \in [-1, 1] \\ & g(\tau, x(\tau), u(\tau)) \leq 0, \qquad \tau \in [-1, 1] \\ & x(-1) = x_{start} \\ & x(1) = x_{free} \end{aligned} \tag{4.13}$$

The collocation points τ_k associated with the non-uniform grid were described in the previous subsection.

2. Discretized state and control vectors

Based on the collocation points, the state and control trajectories are defined as:

$$u \rightarrow u(\tau_0), ..., u(\tau_k), ..., u(\tau_N)$$

$$x \rightarrow x(\tau_0), ..., x(\tau_k), ..., x(\tau_N)$$

We now use Lagrange polynomials to approximate state trajectories as:

$$x(\tau) \approx x^N(\tau) = \sum_{k=0}^{N} x(\tau_k) L_k(\tau) \tag{4.14}$$

Similarly, we approximate control trajectories as:

$$u(\tau) \approx u^N(\tau) = \sum_{k=0}^{N} u(\tau_k) L_k(\tau) \tag{4.15}$$

where $x^N(\tau), u^N(\tau)$ are Lagrange polynomials of degree N and $L_k(\tau)$ are the Lagrange basis functions as defined by (4.11).

3. Integral approximation for cost function

The cost function is approximated using Gaussian quadrature. Note that if the integrand is a polynomial of degree $\leq 2N - 1$, the integral can be computed exactly over then interval $\tau \in [-1, 1]$.

$$\begin{aligned} J(u) &= \phi(t_f, x(1)) + \frac{t_f - t_0}{2} \int_{-1}^{1} L(\tau, x(\tau), u(\tau))\, d\tau \\ &\approx \phi(t_f, x(1)) + \frac{t_f - t_0}{2} \sum_{k=0}^{N} L(\tau_k, x(\tau_k), u(\tau_k)) w_k \end{aligned} \tag{4.16}$$

$$w_k = \frac{2}{N(N+1)} \frac{1}{P_N(\tau_k)^2} \tag{4.17}$$

where w_k are the quadrature weights associated with the integral approximation and P_N is the Legendre polynomial of degree N.

4. Enforcing system dynamics as dynamic constraint

Taking the derivative of (4.14), we get

$$\dot{x}(\tau) \approx \dot{x}^N(\tau) = \sum_{k=0}^{N} x(\tau_k) \dot{L}_k(\tau) \tag{4.18}$$

This can be written as

$$\dot{x}(\tau_k) \approx \dot{x}^N(\tau_k) = \sum_{i=0}^{N} D_{ki} x^N(\tau_i) \tag{4.19}$$

where D_{ki} is an $(N+1)$x$(N+1)$ differentiation matrix given by

$$D_{ki} = \begin{cases} -\frac{P_N(\tau_k)}{P_N(\tau_i)} \frac{1}{\tau_k - \tau_i}, & \text{if } k \neq i \\ \frac{N(N+1)}{4}, & \text{if } k = i = 0 \\ -\frac{N(N+1)}{4}, & \text{if } k = i = N \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

The constraint enforced at each collocation point $k \in \{0, ...N\}$ is therefore

$$x^N(\tau_k) = \sum_{i=0}^{N} D_{ki} x^N(\tau_i) = \frac{t_f - t_0}{2} f(\tau_k, x(\tau_k), u(\tau_k)) \tag{4.21}$$

5. Enforce static constraints

The static constraints are simply enforced at all collocation points such that

$$g(\tau, x(\tau), u(\tau)) \leq 0 \quad \rightarrow \quad g(\tau_k, x(\tau_k), u(\tau_k)) \leq 0, \quad \forall k \in \{0, ..., N\} \tag{4.22}$$

6. Resulting NLP

Following the transcription steps 1-5, we end up with the following NLP:

$$\begin{aligned} \min_{\substack{u_0,\ldots,u_N \\ x_0,\ldots x_N}} \quad & J = \phi(t_f, x(1)) + \frac{t_f - t_0}{2} \sum_{k=0}^{N} L(\tau_k, x(\tau_k), u(\tau_k)) w_k \\ s.t. \quad & \sum_{i=0}^{N} D_{ki} x^N(\tau_i) = \frac{t_f - t_0}{2} f(\tau_k, x(\tau_k), u(\tau_k)), \qquad \forall k \in \{0, ..., N\} \\ & g(t_k, x_k, u_k) \leq 0 \qquad \forall k \in \{0, ..., N\} \\ & x(-1) = x_{start} \\ & x(1) = x_{free} \end{aligned} \tag{4.23}$$

This NLP is now solved using SQP.

7. Solution interpolation

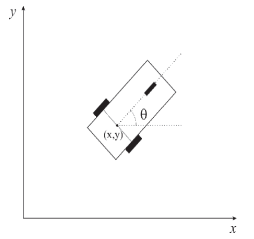
The interpolation of the state and control vectors is done using Lagrange polynomials using (4.14)-(4.15), respectively.

4.6 Simulations

We now present three nonlinear optimal control problems of increasing difficulty. These problems will be used again in the next chapter to demonstrate the application of inverse optimal control.

1) Kinematic unicycle

As our first example, we consider a kinematic unicycle. It is subjected to the following optimal control problem:



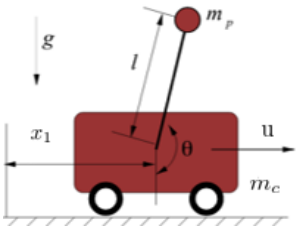
A kinematic unicycle

$$
\begin{aligned}
&\min_u J = \int_0^2 (9x_1^2 + 2x_2^2 + 5x_3^2 + u^2)\, dt \\
&s.t. \quad \dot{x} = \begin{bmatrix} cos(x_3) \\ sin(x_3) \\ u \end{bmatrix} \\
&\qquad x = \begin{bmatrix} x_1, \text{ x position} \\ x_2, \text{ y position} \\ x_3, \text{ heading} \end{bmatrix} \quad u = \Big[\text{angular velocity}\Big] \\
&\qquad u + \frac{x_1}{6} \leq 0 \\
&\qquad x(0) = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \\
&\qquad x(2) = x_{free}
\end{aligned}
\tag{4.24}
$$

Here, $x \in \mathbb{R}^3$ is the state vector of the system and $u \in \mathbb{R}$ is the control input to the system.

2) Cart-pole problem

As our second example case, we present a dynamic system in the form of a cart-pole balancer. Here, a pendulum is attached to a cart and has to be balanced by applying a force as an input to the cart. The optimal control problem in this case is presented as:



A cart-pole balancer

$$\min_u J = \int_0^2 (x^T Q x + u^T R u)\, dt$$

$$s.t. \quad \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1, \text{ cart linear position} \\ x_2, \text{ cart linear velocity} \\ x_3, \text{ pole angular position} \\ x_4, \text{ pole angular velocity} \end{bmatrix} \quad u = \begin{bmatrix} \text{force} \end{bmatrix} \qquad (4.25)$$

$$-0.2 \leq u \leq 0.2$$

$$x(0) = \begin{bmatrix} 1 \\ 0.2 \\ 0 \\ 0 \end{bmatrix} \qquad x(t_f) = free$$

$$\ddot{x}_1 = \frac{1}{m_c + m_p \sin^2(x_2)} [u + m_p \sin(x_2)(l\dot{x}_2^2 + g\cos(x_2))]$$

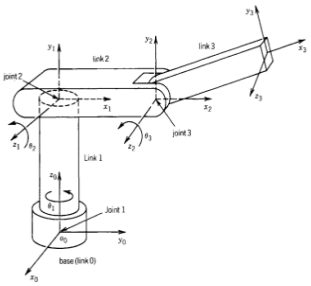
$$\ddot{x}_2 = \frac{1}{l(m_c + m_p \sin^2(x_2))} [-u\cos(x_2) - m_p l \dot{x}_2^2 \cos(x_2) - (m_c + m_p) g \sin(x_2)]$$

$$Q = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \qquad R = 3$$

Here, $x \in \mathbb{R}^4$ are the states of the system and $u \in \mathbb{R}$ is the control input to the system and the Q,R are the weighting matrices associated with state and control trajectories, respectively.

3) Elbow manipulator

As our final example, we present a 3 dof robot manipulator commonly known as an elbow manipulator. The dynamics of the robot is described in robot joint space, although a task space description is also straightforward, using manipulator jacobian. It is actuated by three motors at each of the three joints. The optimal control problem corresponding to the robot is:



An elbow manipulator

$$
\begin{aligned}
&\min_u J = \int_0^3 (x^T Q x + u^T R u)\, dt \\
&s.t. \quad M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \kappa u \\
&\qquad x = \begin{bmatrix} x_{1-3} = \theta_{1-3},\ \text{joint angles}_{1-3} \\ x_{4-6} = \dot{\theta}_{1-3},\ \text{joint velocities}_{1-3} \end{bmatrix} \\
&\qquad u = \begin{bmatrix} u_{1-3},\ \text{joint torques}_{1-3} \end{bmatrix} \\
&\qquad x(0) = \begin{bmatrix} \frac{\pi}{6} \\ \frac{\pi}{8} \\ \frac{\pi}{14} \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&\qquad x(t_f) = free
\end{aligned}
\tag{4.26}
$$

$$
Q = \begin{bmatrix} 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}
$$

Here, $x \in \mathbb{R}^6$ is the state vector of the system and $u \in \mathbb{R}^3$ is the control input to the system. κ is a scaling factor which is used to ensure the numerical stability of the inverse optimal control problem, to be discussed in the next chapter.

4.7 Results and Discussion

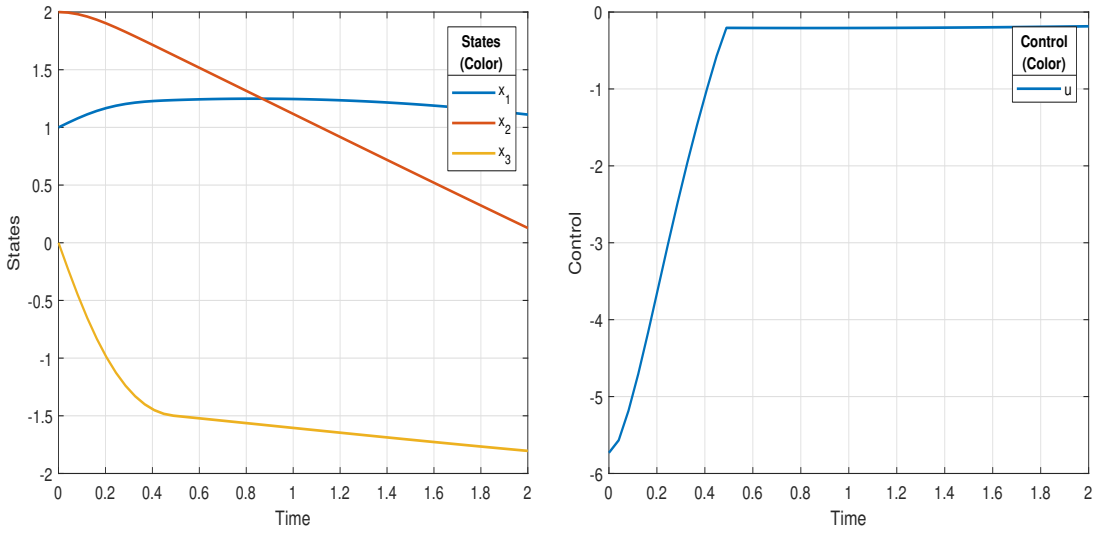
Note: All the example problems presented here are solved using MATLAB's *fmincon* function with SQP solver.

The kinematic unicycle problem as well as the cart-pole problem were solved using trapezoidal transcription method on a grid of 50 and 100 points, respectively. The elbow manipulator problem was solved using pseudospectral method on a grid of 30 nodes. While the unicycle and cart-pole problems have mixed inequality constraint and control saturation constraint, respectively, the elbow manipulator problem was solved without any such constraints. This was done so as to show that the methodology used in solving the first two constrained problems can also be used to solve an unconstrained problem. Because all three example problems are regulation problems, we can see from the plots that both state and control trajectories are trying to regulate to zero value. Also, because both unicycle and cart-pole problems are subject to control constraints, we can see from their respective control plots that the control actions get saturated. As a result, we see non-smooth control trajectories. Elbow manipulator problem, on the other hand, admits smooth control trajectories due to absence of any constraints.

A few additional comments are in order with respect to elbow manipulator problem. Firstly, only the joint angle plots are shown in Figure 4.3 a. This is done so as to avoid clutter in the plot but the joint velocity trajectories do behave expectedly. Secondly, a scaling parameter, κ, is used so as to make the corresponding inverse optimal control problem (which is to be discussed in the next chapter) numerically stable. The use, or lack thereof, of κ, does not affect the numerical stability of the current (forward) optimal control problem. Lastly, no scaling parameter is needed for the inverse problem if gravity compensation is done.

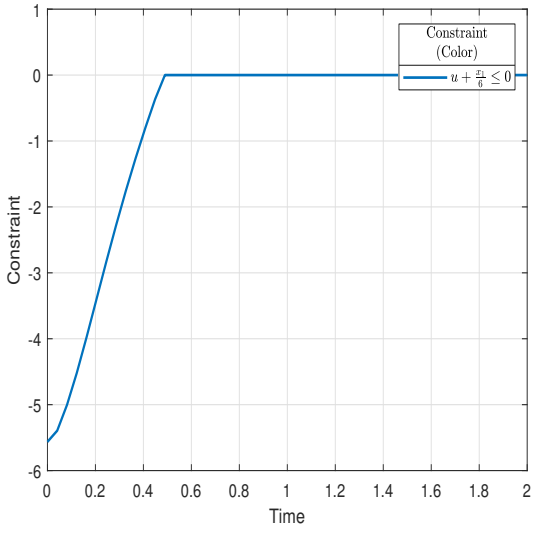
Finally, as a check to validate the accuracy of optimal control solutions, we can see that the constraints in both unicycle and cart-pole are satisfied throughout the trajectories.

As mentioned previously, the state and control trajectory data for each of the three problems will serve as the input to the inverse optimal control method to be presented in the next chapter.



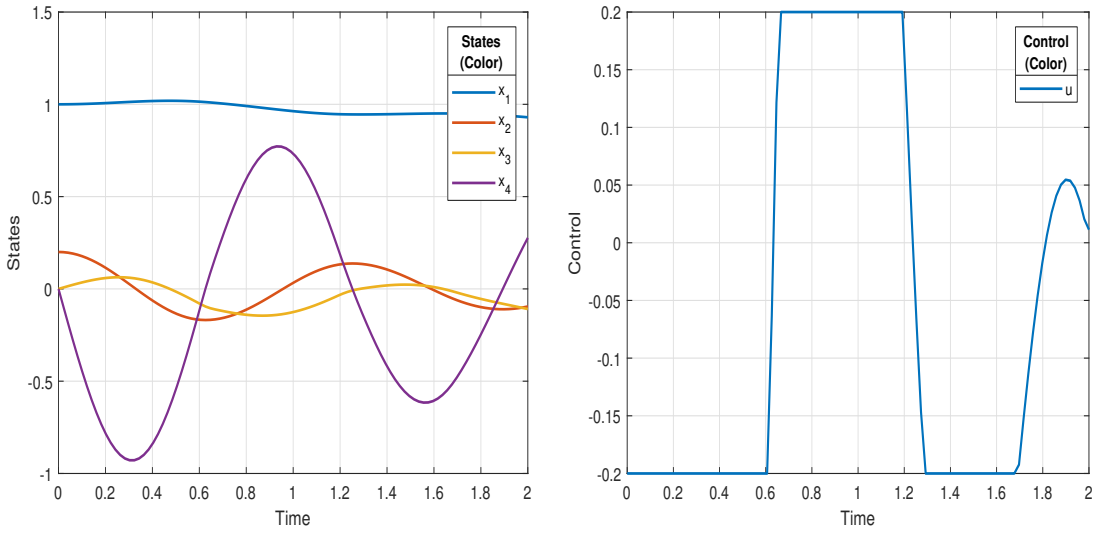
a. States

b. Control



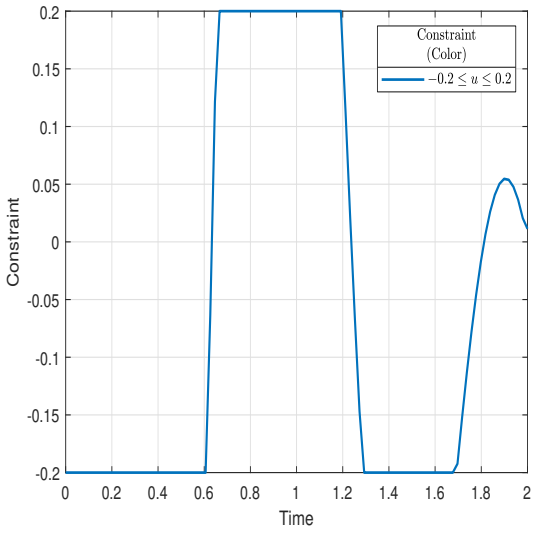
c. Constraint

Figure 4.1: This figure plots state (a.) and control (b.) trajectories for unicycle model. As can be seen, the constraints (c.) are satisfied throughout the trajectory.



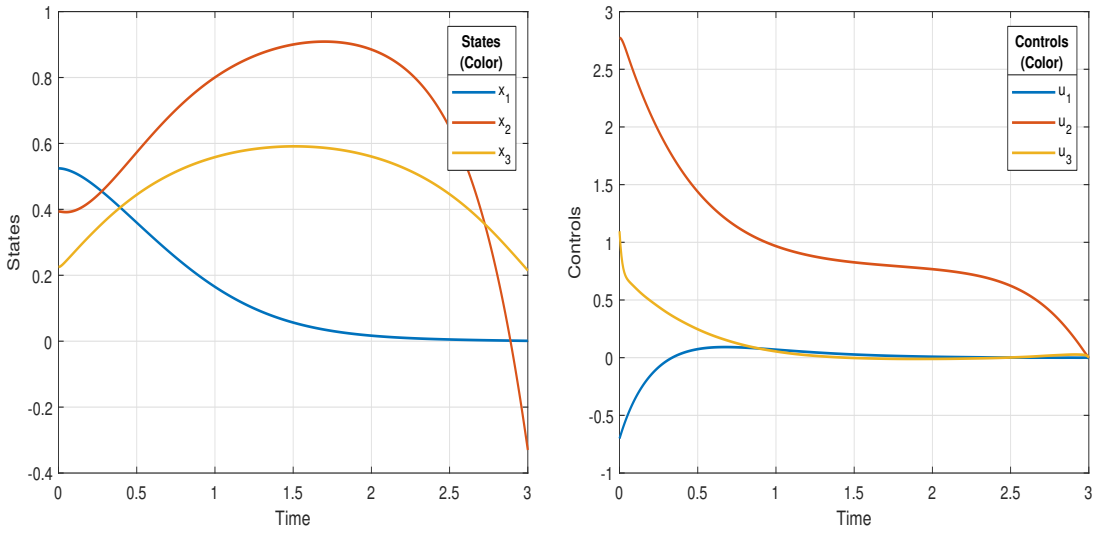
a. States

b. Control



c. Constraint

Figure 4.2: This figure plots state (a.) and control (b.) trajectories for cart-pole model. As can be seen, the constraints (c.) are satisfied throughout the trajectory.



a. States b. Controls

Figure 4.3: This figure plots state (a.) and control (b.) trajectories for elbow manipulator model. This problem does not have any mixed and/or control constraints.

Chapter 5

Inverse Optimal Control

The problem of optimal control is to find a controller which is optimal with respect to performance metric. Commonly, this metric is known as the cost function and the controller tries to minimize this cost function to yield a set of optimal control inputs. Usually, the cost function is an integral cost function and is minimized along the trajectory of the system from some initial state to some final state of the system.

Inverse optimal control (IOC) is the (inverse) problem of trying to find the underlying cost function with respect to which the state and control trajectories are (locally) optimal. In this chapter, we extend the method of Johnson et al. [18] to impute cost functions when the systems involve state and/or control constraints.

5.1 Constrained Inverse Optimal Control

In the case of unconstrained IOC problem with only Lagrange cost, Johnson et al. were able to setup the inverse problem as an LQR. However, when we incorporate mixed inequality constraints (i.e., constraints containing both state and control vectors) and/or control constraints, the LQR structure of the problem no longer exists.

5.1.1 Problem Statement

We now present a generalization of the inverse problem in the light of mixed inequality constraints in the Bolza form.

We now consider optimal control problems of the form

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_{t_0}^{t_f} c^T \phi(t, x(t), u(t))\, dt \\ s.t. \quad & \dot{x}(t) = f(t, x(t), u(t)) \\ & g(t, x(t), u(t)) \leq 0 \\ & x(0) = x_{start} \\ & x(t_f) = free \end{aligned} \tag{5.1}$$

Notation:

$x(t) \in \mathbb{R}^{n_x}$ is the state vector.

$u(t) \in \mathbb{R}^{n_u}$ is the control vector.

$f(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the system dynamics.

$\phi(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^k$ is known feature vector; k is the length of feature vector.

$c \in \mathbb{R}^k$ is weight vector.

$g(t, x(t), u(t)) : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^s$ is path inequality constraint; s is the size of constraint vector.

$c^T \phi(t, x(t), u(t))$ is the scalar cost function

$\nabla_z q$ is gradient of function q with respect to variable z.

We now make following assumptions on the optimal control problem for it to be well posed and amenable to the analysis of this work.

1) $f[x(t), u(t)]$ is assumed to be continuous in time and differentiable in state and control variables.

2) $\nabla_x f[x(t), u(t)]$ is continuous in both time and control variables.

3) The control variable is at least piecewise continuous in time.

4) The feature or basis vector $\phi[x(t), u(t)]$ is differentiable in both state and control variables.

5) The functions that appear in equation (7) are not explicit function of time

6) The inequality path constraint, $g[t, x(t), u(t)]$ is continuous in time, state and control variables and differentiable in state and control variables.

To discuss the problem of optimal control under mixed inequality constraints, we first define the Hamiltonian, $\mathscr{H}$ and the Lagrangian, $\mathscr{L}$ as follows (for brevity, we will, at times, omit the time dependence of variables in the equations to follow):

$$\mathscr{H}(t, x, u, p) = c^T \phi(t, x, u) + p^T f(t, x, u) \tag{5.2}$$

$$\mathscr{L}(t,x,u,p,\mu) = \mathscr{H}(t,x,u,p) + \mu^T g(t,x,u) \tag{5.3}$$

where $p \in \mathbb{R}^{n_x}$ is the adjoint variable and $\mu \in \mathbb{R}^k$ is the Lagrange multiplier. We also define the state-dependent control region as

$$\Lambda(t,x) = \{u \in \mathbb{R}^{n_u} | g[t,x,u] \leq 0\} \subset \mathbb{R}^{n_u} \tag{5.4}$$

We are now in a position to state a formulation of Pontryagin's minimum principle that is used when dealing with optimal control problems with mixed inequality constraints. The results are based on the necessary conditions of optimality derived in Chapter 3 and are presented here again for convenience:

$$\begin{aligned}
&u^*(t) = \underset{u \in \Lambda(t,x^*)}{\operatorname{argmin}} H(t,x^*(t),u,p(t)) \\
&\nabla_u L^*(t,x,u,p(t),\mu(t)) = 0 \\
&\dot{p}(t) = -\nabla_x L^*(t,x,u,p(t),\mu(t)) \\
&p(t_f) = 0 \\
&\mu(t) \geq 0 \\
&\mu(t)^T g^*(t,x,u) = 0
\end{aligned} \tag{5.5}$$

Applying the aforementioned necessary conditions to our problem of optimal control, we have

$$\begin{aligned}
-\dot{p}(t) &= c^T \nabla_x \phi(t,x,u) + p^T \nabla_x f(t,x,u) + \mu^T \nabla_x g(t,x,u) \\
p(t_f) &= 0
\end{aligned} \tag{5.6}$$

$$0 = c^T \nabla_u \phi(t,x,u) + p^T \nabla_u f(t,x,u) + \mu^T \nabla_u g(t,x,u) \tag{5.7}$$

Assuming similar naming convention as [18], we let

$$\dot{p}(t) = v(t) \tag{5.8}$$

$$z(t) = \begin{bmatrix} c \\ p(t) \\ \mu(t) \end{bmatrix} \tag{5.9}$$

The residual function $r[c, p(t), v(t), \mu(t)]$ is then defined by assuming slack in the necessary conditions of optimality as

$$r[z(t), v(t)] = \begin{bmatrix} \nabla_x \phi \Big|_{(x,u)}^T & \nabla_x f \Big|_{(x,u)}^T & \nabla_x g \Big|_{(x,u)}^T \\ \nabla_u \phi \Big|_{(x,u)}^T & \nabla_u f \Big|_{(x,u)}^T & \nabla_u g \Big|_{(x,u)}^T \end{bmatrix} z(t) + \begin{bmatrix} I \\ 0 \end{bmatrix} v(t) \tag{5.10}$$

5.1.2 Residual Function Optimization

We now state the optimization problem which is a result of constrained inverse optimal control. The solution to this residual optimization problem recovers the underlying weight vector.

$$\begin{aligned} \underset{z(t),v(t)}{\text{minimize}} \quad & \int_{t_0}^{t_f} ||r[z(t), v(t)]||^2 \, dt \\ s.t. \quad & \dot{p}(t) = v(t) \\ & p(t_f) = 0 \\ & \mu(t) \geq 0 \\ & \mu(t)^T g(x(t), u(t)) = 0 \end{aligned} \tag{5.11}$$

where the last two conditions are also known as complementary slackness which appear here due to mixed inequality constraint present in forward optimal control.

Note that this is a convex optimization problem with input $v(t)$ and state $z(t)$. Thus, it is amenable to efficient numerical solution via transcription methods such as those described in [31].

5.1.3 Improving Accuracy of IOC

When learning a cost function from observations, the performance of the IOC method improves as the number of trajectories or observations is increased. Consider k trajectories which may begin from different initial conditions, but run for the same length of time t_f. We label these k state and control trajectories as $(x^{(j)}, u^{(j)}) \ \ \forall j \in \{1, ..., k\}$. In this case, the IOC variables of interest, $v(t)$, $z(t)$, and $r(z(t), v(t))$ are extended to include the information from k trajectories

so that:

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

To make the notation succinct, we make the following simplifications in reference to (5.10). Let

$$A = \begin{bmatrix} \nabla_x \phi \Big|_{(x,u)}^T \\ \nabla_u \phi \Big|_{(x,u)}^T \end{bmatrix} \qquad B = \begin{bmatrix} \nabla_x f \Big|_{(x,u)}^T \\ \nabla_u f \Big|_{(x,u)}^T \end{bmatrix}$$

$$C = \begin{bmatrix} \nabla_x g \Big|_{(x,u)}^T \\ \nabla_u g \Big|_{(x,u)}^T \end{bmatrix} \qquad \mathscr{I} = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

Using the above notation, we can rewrite (5.10) as

$$r(z(t), v(t)) = \begin{bmatrix} A & B & C \end{bmatrix} z(t) + \mathscr{I}\, v(t) \tag{5.12}$$

Extending to k trajectories, we get

$$r(z(t), v(t)) =$$

$$\begin{bmatrix} A^{(1)} & B^{(1)} & 0 & 0 & C^{(1)} & 0 & 0 \\ \vdots & 0 & \ddots & 0 & 0 & \ddots & 0 \\ A^{(k)} & 0 & 0 & B^{(k)} & 0 & 0 & C^{(k)} \end{bmatrix} z(t) + \begin{bmatrix} \mathscr{I}^{(1)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathscr{I}^{(k)} \end{bmatrix} v(t) \tag{5.13}$$

As a result of incorporating k observations, note that the resulting residual vector, $r(z(t), v(t))$, has a lot of sparse structure. This structure is utilized when solving the residual minimization problem (5.11).

Remarks

Following remarks are now in order:

1. Convex Optimization: Note that (5.11) is a convex optimal control problem with input $v(t)$

and state $z(t)$. Thus, it is amenable to efficient numerical solution via transcription methods such as those described in [31].

2. Normalization: In the absence of any normalization procedure, (5.11) can be trivially solved by $z = 0$. However, in many applications, we know some part of the cost function from domain specific knowledge. Therefore, we can often choose one of the weights in weight vector c to be equal to 1. More generally, we can impose the constraints that $c_i \geq 0$ and $\sum_{i=1}^{k} c_i = 1$ (see [45]).

3. Sufficiency condition: The present work does not provide any sufficiency result to show that the computed solution is indeed a minimum of the problem, the sufficiency condition is implicitly assumed. This is a common assumption in the literature [18, 59, 60].

5.2 Simulations

This section presents the same three example problems from the previous chapter, but this time they will be used to implement inverse optimal control method discussed in this chapter. A description of the three examples is provided again for convenience.

1) Kinematic Unicycle

$$
\begin{aligned}
\min_{u} J &= \int_0^2 (9x_1^2 + 2x_2^2 + 5x_3^2 + u^2)\, dt \\
s.t. \quad \dot{x} &= \begin{bmatrix} cos(x_3) \\ sin(x_3) \\ u \end{bmatrix} \\
x &= \begin{bmatrix} x_1, \text{ x position} \\ x_2, \text{ y position} \\ x_3, \text{ heading} \end{bmatrix} \quad u = \begin{bmatrix} \text{angular velocity} \end{bmatrix} \\
& u + \frac{x_1}{6} \leq 0 \\
x(0) &= \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \\
x(2) &= x_{free}
\end{aligned}
\tag{5.14}
$$

2) Cart-pole balancing

$$
\begin{aligned}
&\min_u J = \int_0^2 (x^T Q x + u^T R u)\, dt \\
&s.t. \quad \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} \\
&\qquad x = \begin{bmatrix} x_1,\ \text{cart linear position} \\ x_2,\ \text{cart linear velocity} \\ x_3,\ \text{pole angular position} \\ x_4,\ \text{pole angular velocity} \end{bmatrix} \quad u = \begin{bmatrix} \text{force} \end{bmatrix} \\
&\qquad -0.2 \le u \le 0.2 \\
&\qquad x(0) = \begin{bmatrix} 1 \\ 0.2 \\ 0 \\ 0 \end{bmatrix} \\
&\qquad x(t_f) = free
\end{aligned}
\tag{5.15}
$$

where,

$$
\ddot{x}_1 = \frac{1}{m_c + m_p \sin^2(x_2)} [u + m_p \sin(x_2)(l\dot{x}_2^2 + g\cos(x_2))]
$$

$$
\ddot{x}_2 = \frac{1}{l(m_c + m_p \sin^2(x_2))} [-u\cos(x_2) - m_p l \dot{x}_2^2 \cos(x_2) - (m_c + m_p) g \sin(x_2)]
$$

$$
Q = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \qquad R = 3
$$

3) Elbow manipulator

$$
\begin{aligned}
&\min_{u} J = \int_0^3 (x^T Q x + u^T R u)\, dt \\
&s.t. \quad M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \kappa u \\
&\qquad x = \begin{bmatrix} x_{1-3} = \theta_{1-3},\ \text{joint angles}_{1-3} \\ x_{4-6} = \dot{\theta}_{1-3},\ \text{joint velocities}_{1-3} \end{bmatrix} \qquad u = \begin{bmatrix} u_{1-3},\ \text{joint torques}_{1-3} \end{bmatrix} \\
&\qquad x(0) = \begin{bmatrix} \frac{\pi}{6} \\ \frac{\pi}{8} \\ \frac{\pi}{14} \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&\qquad x(t_f) = free \\
&\qquad\qquad Q = \begin{bmatrix} 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}
\end{aligned}
\tag{5.16}
$$

5.3 Results and Discussion

Note: All the example problems presented here are solved using MATLAB's *fmincon* function with SQP solver.

Figure 5.1 a,b. show the state and control trajectories for the kinematic unicycle. As we can see, the predicted state and control trajectories virtually lie on top of the actual simulated trajectories. As a check on the validity of the IOC method, we can also see that dual constraint (c.) as well as complementary slackness condition (d.) are satisfied.

Figure 5.2 a,b. show the state and control trajectories for the cart-pole problem. Here again, the predicted trajectories are in close agreement with the actual simulated trajectories. As a check on the validity of the IOC method, we can also see that dual constraint (c.) as well as

Table 5.1: Comparison of weight vector from forward and inverse optimal control

System	**Feature vector (ϕ)**	**Weight vector (from forward method) ($c = c_{true}$)**	**Learned weight vector (from inverse method) ($c = c_{learned}$)**	**Minimized residual function (r)**
Kinematic unicycle	$\phi = [x_1^2, x_2^2, x_3^2, u^2]$	[9, 2, 5, 1]	[8.9867, 1.9733, 4.9898,1]	2.767×0^{-04}
Cart-pole balancing	$\phi = [x_1^2, x_2^2, x_3^2, x_4^2, u^2]$	[0.5, 9, 1, 6, 3]	[0.5, 9.2268, 1, 5.9829, 2.9952]	3.258×10^{-05}
Elbow manipulator	$\phi = [x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, x_6^2, u_1^2, u_2^2, u_3^2]$	[2.5, 4, 1.5, 0, 0.5 1, 1, 3, 2]	[2.4751, 3.9029, 1.4668, 0.0083, 0.4892 0.9724, 1, 2.9299, 1.9573]	5.070×10^{-05}

complementary slackness condition (d.) are satisfied. Another important thing to note is that the box constraint in this problem is handled by breaking down the constraint into two separate constraints g_1, g_2.

Finally, Figure 5.3 a,b. show the state and control trajectories for the robot problem. Like in the previous two cases, the predicted trajectories are indistinguishable from the actual trajectories. However, unlike the last two cases, this problem does not have state and/or control constraints.

While Figures 5.1-5.3 provide visual description of the results, Table 5.1 provides a quantitative measure of the performance of the inverse optimal control method. It compares the true cost functions and the learned cost functions and also provides a value for the residual function. The quality of the solution can be judged from its value of final residual function: the lower the value of the residual, the better IOC method was able to recover the underlying cost function.

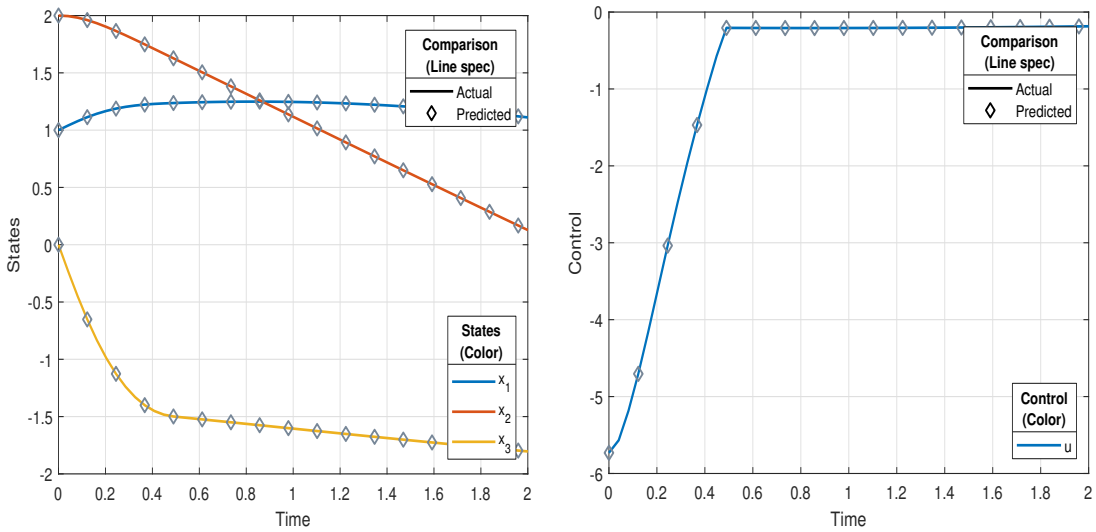
While the inverse optimal control method perform well in recovering the underlying cost functions in all example cases, the feature errors can be further reduced by either sampling the observations at a faster rate, or by observing more trajectories. In a real life scenario, if the sensors recording the observational data cannot sample at a faster rate, simply more trajectories can be observed. On the other hand, if recording data for several trajectories is cumbersome, high resolution sensors can be used to sample more in a given time for a given trajectory.

Cost function, as mentioned in this work, has sometimes interchangeably been used with the weight vector. However, it is important to note that they are distinct. A cost function is composed of a weight vector and a feature vector. Feature vectors are generally domain dependent and require knowledge of the expert to construct it from experience and sound judgment. Once

a feature vector is passed to the inverse optimal control, the inverse methods then try to learn the weight vector. The weight vector indicates the relative importance of the different features present in the feature vector. In this paper, the weight vector is always assumed to be positive.

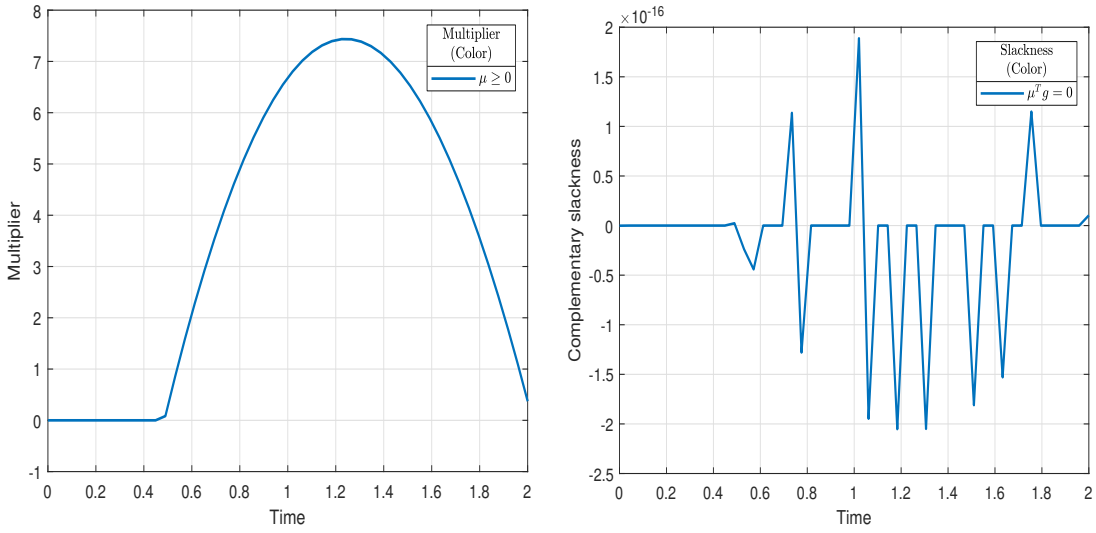
In reality, one will never have access to true cost function so that it can be compared against learned cost function. The best one can do is to learn an underlying cost function which respects the constraints and compare the predicted trajectories from that cost function with the actual trajectories generated by the system of interest. Typically, the expert will have some knowledge about the ways in which a given system is constrained. These constraints should be passed to the inverse methods so that a correct cost function can be learned. If the expert fails to account for the constraints that may be present in a system, it is very likely that the learned cost function will perform poorly. Also, it is possible that if enough trajectories are not observed, the cost function learned maybe incorrect. This is because the inverse methods may not have been able to sample the feasible state/control space completely. Another point to note is that although the feature vectors in the examples provided were chosen to be quadratic for reasons of physical intuition, it is possible to choose the feature vectors to be non-quadratic, if the problem so desires. The inverse methods will still be able to provide locally optimal solution to the residual minimization problem in (5.11)

In the case of unicycle and cart-pole problem, we estimated the weight vector using two trajectories, while in the case of elbow manipulator, we only used a single trajectory. For unconstrained problems, a single trajectory is typically sufficient to learn the cost weights. However, for constrained systems, a single trajectory often does not give sufficient information to learn the cost. In particular, optimal solutions for problems with input constraints often exhibit control saturation throughout the trajectory. A saturated control constraint implies that the corresponding cost is sufficiently small to allow saturation, but the precise cost weight cannot be obtained. We can improve the likelihood of observing unsaturated trajectories by simply observing more trajectories.



a. States

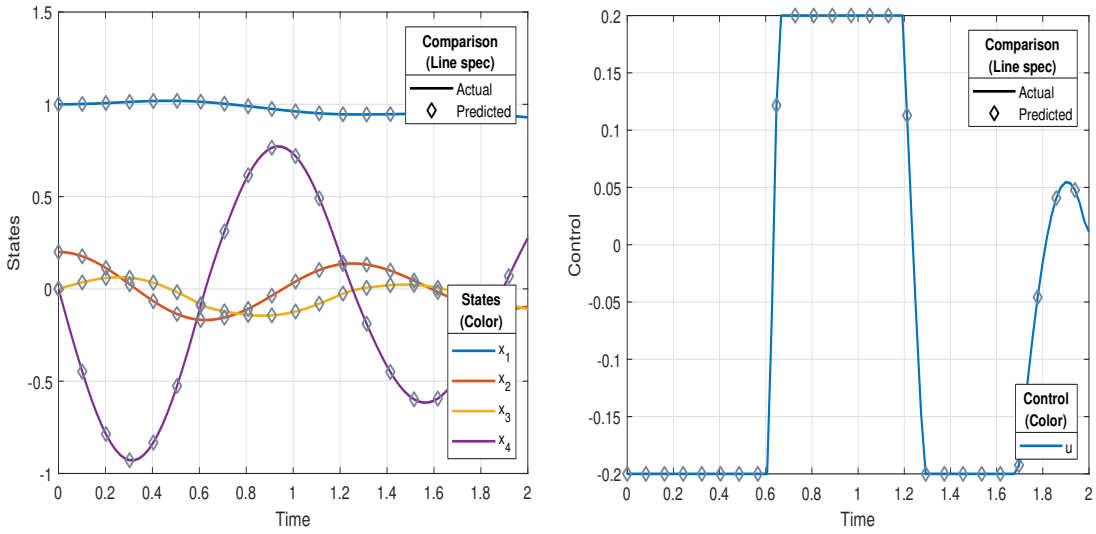
b. Control



c. Dual feasibility

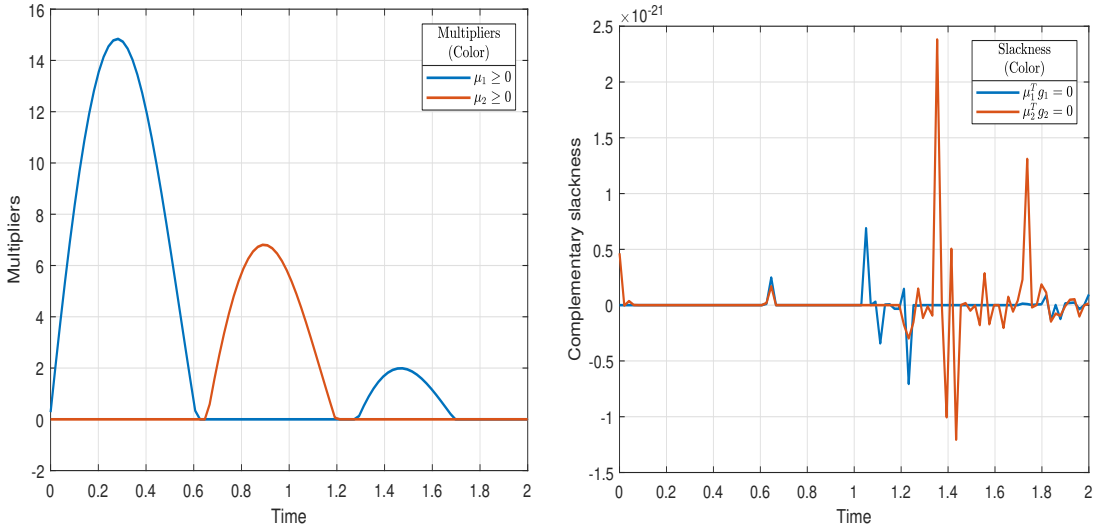
d. Complementary slackness

Figure 5.1: This figure compares the state (a.) and control (b.) trajectories for the unicycle model computed using true (actual) and learned (predicted) cost functions. As can be seen, the computed costs are sufficiently accurate to reproduce the original trajectories. We also see that the dual feasibility constraint (c.) as well complementary slackness (d.) are satisfied. Here constraint $g = u + \frac{x_1}{6}$.



a. States

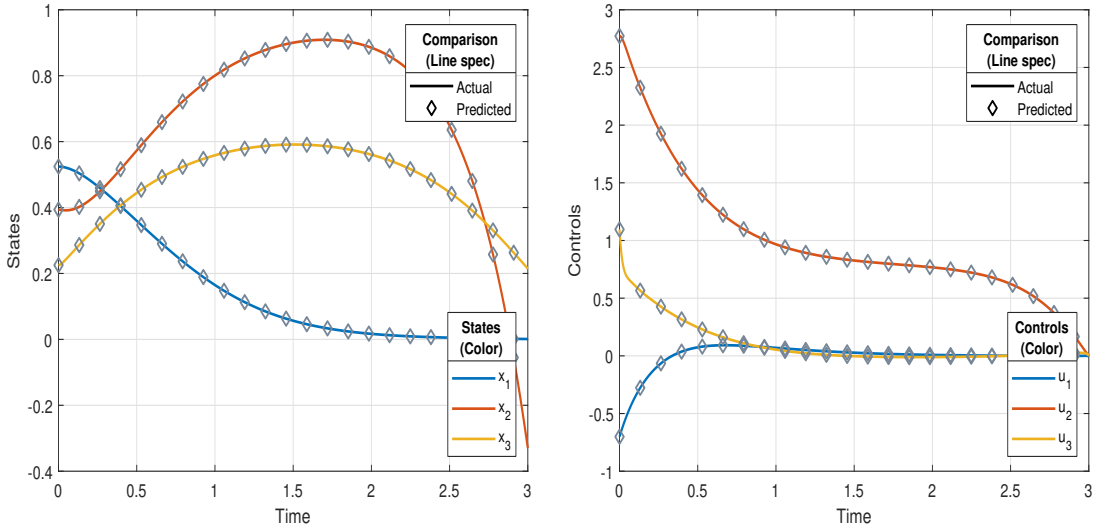
b. Control



c. Dual feasibility

d. Complementary slackness

Figure 5.2: This figure compares the state (a.) and control (b.) trajectories for the cart-pole model computed using true (actual) and learned (predicted) cost functions. As can be seen, the computed costs are sufficiently accurate to reproduce the original trajectories. We also see that the dual feasibility constraint (c.) as well complementary slackness (d.) are satisfied. Here constraints $g_1 = -u - 0.2$ and $g_2 = u - 0.2$, respectively.



a. States b. Controls

Figure 5.3: This figure compares the state (a.) and control (b.) trajectories for the elbow manipulator model computed using true (actual) and learned (predicted) cost functions. As can be seen, the computed costs are sufficiently accurate to reproduce the original trajectories. This problem does not have any mixed and/or control constraints.

Chapter 6

Dynamic Games

Dynamic games arise when multiple agents with differing objectives interact over time. They can be thought of as an obvious extension to single objective optimization problems (such as optimal control problems). Dynamic games are a natural framework for studying interactive behaviors which are not fully cooperative, and not fully competitive. Such games are called non-zero sum. Consider the case of vehicle driving. Drivers prioritize individual speed, safety, and comfort. While drivers are largely indifferent to the behavior and costs of others, they will cooperate to prevent collisions. Furthermore, different individuals have varying behaviors based on their particular objectives. For example, some prioritize speed, while others prioritize safety. In nature, non-zero sum behavior arises when animals are largely indifferent to one-another, unless they need to compete for a resource.

In this chapter, we present a novel method for solving Nash equilibrium solution of a dynamic game. We also present two example cases where the solution is computed using this method.

6.1 Problem Formulation of Dynamic Games

Consider an N player nonzero-sum open-loop dynamic game where every player i $\in \{1, ..., N\}$ has an associated performance index J_i. The dynamics associated with state of the game evolve via a coupled nonlinear set of differential equations $\dot{x} = f(t, x, u_1, ..., u_N)$. The goal of each player is to minimize their own performance index by an appropriate choice of control action

u_i. The actions of the other agents are denoted by u_{-i}. The problem is stated formally as:

$$\begin{aligned} J_i(u_i, u_{-i}) = \underset{u_i}{\text{minimize}} & \int_{t_0}^{t_f} c_i^T \phi_i(t, x(t), u_1(t), ..., u_N(t))\, dt \\ s.t. \qquad & \dot{x}(t) = f(t, x(t), u_i(t), u_{-i}(t)) \\ & g_i(t, x(t), u_i(t), u_{-i}(t)) \leq 0 \\ & x(0) = x_{start} \\ & x(t_f) = free \end{aligned} \tag{6.1}$$

Because we are considering deterministic dynamics and open-loop trajectories, the performance index, $J_i(u_i, u_{-i})$ can be viewed as function of the inputs. Before we proceed further, we provide some useful definitions which will be relevant to the present work.

Definition (***Zero-sum dynamic game***): A dynamic game in which each player's gains or losses is exactly balanced by the gains or losses of other players. This leads to the overall gains and losses to sum exactly to zero. This game is also called as strictly competitive.

In contrast with a zero-sum dynamic game, a nonzero-sum game can be seen as a less restrictive game setting.

Definition (***Nonzero-sum dynamic game***): A dynamic game in which the overall gains and losses can be less than or more than zero. In a nonzero-sum dynamic game, every player tries to minimize its own performance index, by choosing an optimal u_i. This setting can be either competitive or cooperative or a mixture of both.

The optimality of the control input u_i leads to the concept of an (open-loop) Nash equilibrium.

Definition (***Nash equilibrium***): A dynamic game involving N players is said to be in a local open-loop Nash equilibrium if any admissible control action u_i in a neighborhood of u_i^* of the i^{th} player is such that

$$J(u_i^*, u_{-i}^*) \leq J(u_i, u_{-i}^*). \tag{6.2}$$

We will often refer to a solution satisfying (6.2) in a neighborhood simply as a Nash equilibrium. Indeed, it is a Nash equilibrium of the game in a restricted neighborhood.

In the present work, we focus our attention to solving only nonzero-sum dynamic games.

Following similar development as in the case of optimal control, the Hamiltonian, $\mathscr{H}$ and the Lagrangian, $\mathscr{L}$ associated with the i^{th} player is

$$\mathscr{H}_i(t, x, u_i, u_{-i}, p_i) = c_i^T \phi_i(t, x, u_i, u_{-i}) + p_i^T f(t, x, u_i, u_{-i}) \tag{6.3}$$

$$\mathscr{L}_i(t, x, u_i, u_{-i}, p_i, \mu_i) = H_i(t, x, u_i, u_{-i}, p_i) + \mu_i^T g_i(t, x, u_i, u_{-i}) \tag{6.4}$$

At a Nash equilibrium, every player simultaneously minimizes its own performance index subject to dynamics of the game and player constraints. Thus, every player $i \in \{1, ..., N\}$ satisfies the following set of necessary conditions:

Stationarity

$$\nabla_{u_i} L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) = 0 \tag{P1}$$

$$\begin{aligned} &\dot{p}_i(t) = -\nabla_x L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) \\ &p_i(t_f) = 0 \end{aligned} \tag{P2}$$

Primal feasibility

$$\begin{aligned} &\dot{x}(t) = f(t, x(t), u_i(t), u_{-i}(t) \\ &x(0) = x_0 \end{aligned} \tag{P3}$$

$$g_i(t, x(t), u_i(t), u_{-i}(t)) \leq 0 \tag{P4}$$

Dual feasibility

$$\mu_i(t) \geq 0 \tag{P5}$$

Complementary slackness

$$\mu_i(t)^T g_i^*(t, x, u_i, u_{-i}) = 0 \tag{P6}$$

6.2 Semi-direct Method

We now present a solution strategy to deal with non-zero sum dynamic games. To our knowledge, the method we present here is novel and is now described below.

6.2.1 Semi-Direct Method for Unconstrained Dynamic Games

Recall that (6.1) is subject to mixed inequality constraint $g_i(t, x(t), u_i(t), u_{-i}(t)) \leq 0$, for every player i. Let us ignore this constraint so we can focus on the unconstrained problem. This problem results in necessary conditions of optimality given by eqns. (P1)-(P3). As such, these set of equations result in a two-point boundary value problem (TPBVP) and can be solved using solvers such as bvp4c provided by MATLAB. However, in this paper, we provide a solution technique which we call semi-direct method. The method can be described as recasting the TPBVP as an optimal control feasibility problem. Formally, the problem can be described as:

$$
\begin{aligned}
& \min J = 0 \\
s.t. \quad & \dot{x}(t) = f(t, x(t), u_i(t), u_{-i}(t)) \\
& x(0) = x_0 \\
& \dot{p}_i(t) = -\nabla_x L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) \\
& p_i(t_f) = 0 \\
& \nabla_{u_i} L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) = 0 \\
& \forall i \in \{1, ..., N\}
\end{aligned} \tag{6.5}
$$

It is important to note that (6.5) is solved simultaneously for every player i.

We now see that the necessary conditions of optimality (P1)-(P3), comprising of a set of differential equations, can be seen as the constraints in an optimal control problem.

6.2.2 Semi-Direct Method for Constrained Dynamic Games

The approach we developed for the unconstrained case has powerful consequences because it can be easily extended to the constrained case. This generalization is not possible with TP-BVP solvers. In the constrained case, we add the remaining necessary conditions of optimality (P4)-(P6) (related to the constraints). The problem can again be posed as an optimal control feasibility problem:

$$
\begin{aligned}
& \min J = 0 \\
s.t. \quad & \dot{x}(t) = f(t, x(t), u_i(t), u_{-i}(t)) \\
& x(0) = x_0 \\
& \dot{p}_i(t) = -\nabla_x L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) \\
& p_i(t_f) = 0 \\
& \nabla_{u_i} L_i^*(t, x, u_i, u_{-i}, p_i(t), \mu_i(t)) = 0 \\
& g_i(t, x(t), u_i(t), u_{-i}(t)) \leq 0 \\
& \mu_i(t) \geq 0 \\
& \mu_i(t)^T g_i(t, x, u_i, u_{-i}) = 0 \\
& \forall i \in \{1, ..., N\}
\end{aligned}
\tag{6.6}
$$

As noted previously for the unconstrained dynamic game, (6.6) is solved simultaneously for every player i.

Therefore, the cost of going from unconstrained case to a constrained is only addition of extra constraints to the feasibility problem which could be solved using standard optimization solvers.

We have thus shown that any dynamic game consisting of multiple players, with each player optimizing their own performance index, as well as being subjected to game dynamics and constraints, can be reduced to a single (trivial, since J=0) objective optimization problem subject to dynamic and path constraints.

Remarks about Semi-Direct Method

1. The reason we call our method semi-direct is due to the fact that we combine both, direct and indirect methods used for solving optimal control problems, to solve the dynamic game problem given by (6.1). This problem essentially reduces to a feasibility problem because there is no objective function to optimize over. Any solution which lies in the feasible space of (6.6) is an optimal solution to the original problem (6.1).

2. The present work does not provide any existence or uniqueness results for Nash equilibrium solution, nor does it provide any sufficiency results to prove that the equilibrium is necessarily a

minimum of the dynamic game. The work is only concerned with showing that if a Nash equilibrium exists, then the semi-direct method can be used to compute state and control trajectories of the players that are in Nash equilibrium.

6.3 Simulations

In this section, we present two example cases of dynamic games to demonstrate and evaluate the performance of the semi-direct method. The example cases include duopolistic competition scenario and a nonlinear polynomial game, both problems were inspired from [26].

1) Duopolistic competition

As our first example case, we consider the following linear-quadratic nonzero-sum differential game:

$$\begin{aligned}
&\min_{u_1} J_1 = \frac{1}{2}\int_0^3 (x^2 + 2u_1^2)\, dt \\
&\min_{u_2} J_2 = \frac{1}{2}\int_0^3 (4x^2 + u_2^2) \\
&s.t. \qquad \dot{x} = x + u_1 + u_2 \\
&\qquad\quad x + u_1 + 5 \leq 0 \\
&\qquad\quad x(0) = x_0 \\
&\qquad\quad x(3) = x_{free}
\end{aligned} \tag{6.7}$$

Here, $x \in \mathbb{R}$ is the state vector of the game and $u \in \mathbb{R}^2$ is the control vector whose elements consists of the control actions for Player 1 and Player 2.

2) Nonlinear polynomial game

As our second example, we consider a model for a nonlinear electric circuit managed by two companies employing different cost metrics for the consumed electricity. The objective of the game is to minimize the cost incurred by each electric company:

$$
\begin{aligned}
&\min_{u_1} J_1 = \frac{1}{2}\int_0^3 (2x_1^2 + 7x_2^2 + u_1^2 + u_2^2)\, dt \\
&\min_{u_2} J_2 = \frac{1}{2}\int_0^3 (3x_1^2 + 8x_2^2 + u_1^2 + u_2^2)\, dt \\
&s.t. \qquad \dot{x} = \begin{bmatrix} x_2 \\ x_1^2 + u_1 + u_2 \end{bmatrix} \\
&\qquad -1 \leq u_1 \leq 1 \\
&\qquad x_2 + u_2 \leq 1 \\
&\qquad x(0) = x_0 \\
&\qquad x(3) = x_{free}
\end{aligned}
\tag{6.8}
$$

Here, $x \in \mathbb{R}^2$ is the state vector of the game and $u \in \mathbb{R}^2$ is the control vector whose elements consists of the control actions for Player 1 and Player 2.

6.4 Results and Discussion

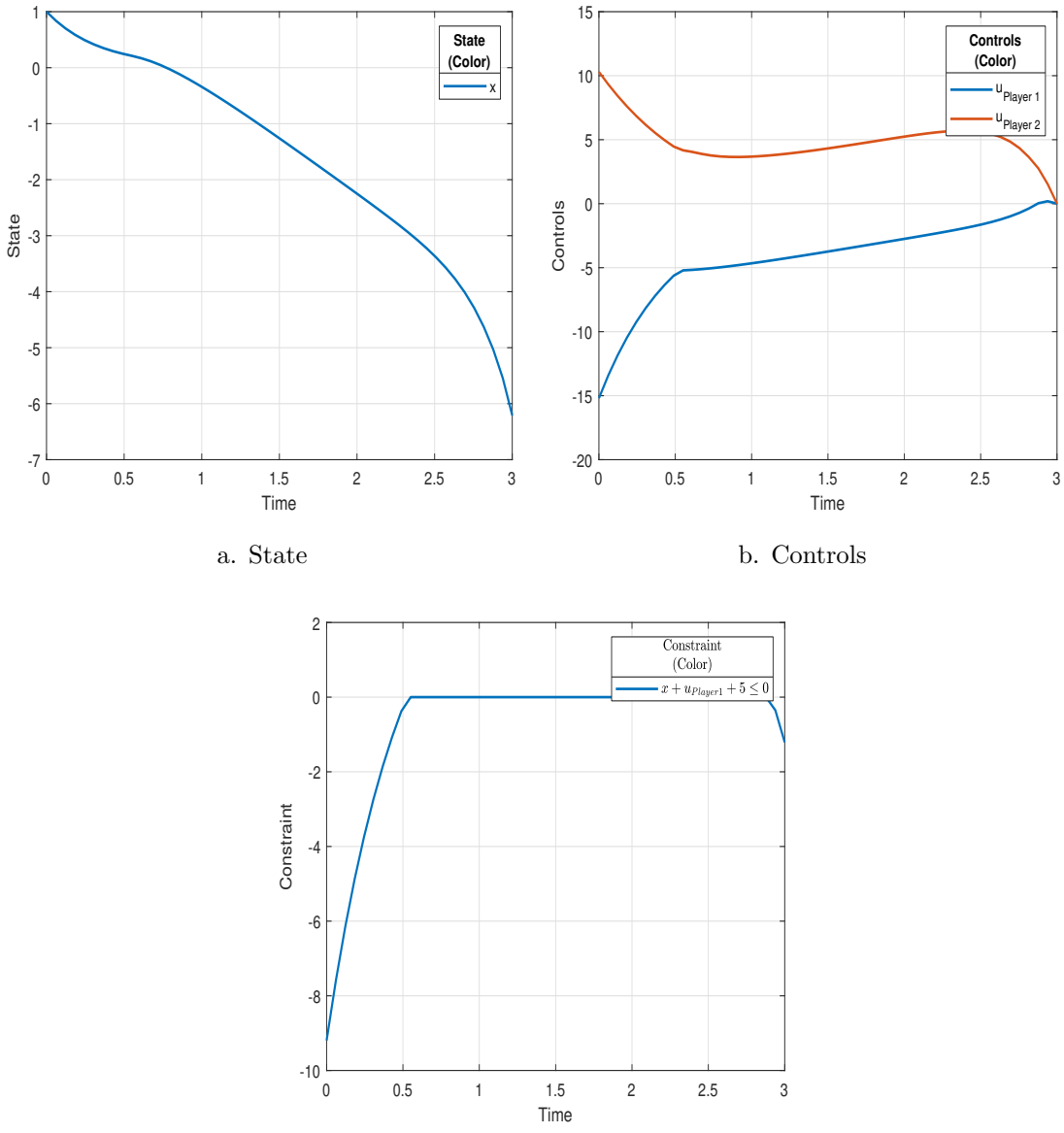
Note: Both example problems presented here are solved using MATLAB's *fmincon* function with SQP solver.

The duopolistic competition problem as well as the nonlinear polynomial game problem were solved using trapezoidal transcription method on a grid of 50 points, each. Both problems are subjected to either mixed inequality constraints and/or control constraints.

Figure 7.1 a,b. show state and control trajectories of the two players in the duopolistic competition. The control action of player 1 shows some non-smooth behavior due to control saturation imposed via the mixed-inequality constraint in the system. As a check to validate solution accuracy, we see that the control constraint in Figure 7.1 c. is respected at all times.

Figure 7.2 a,b. show state and control trajectories of the two players in the nonlinear polynomial game. Unlike the previous game, this game is 2-dimensional. It also has multiple constraints. The control actions of neither player shows any non-smooth behavior, which means that the two players did not saturate their control. As a check to validate the accuracy of the solution, we see that the control constraint in Figure 7.1 c. is again respected at all times.

The state and control trajectory data for both these problems will serve as the input to the inverse dynamic game method to be presented next.

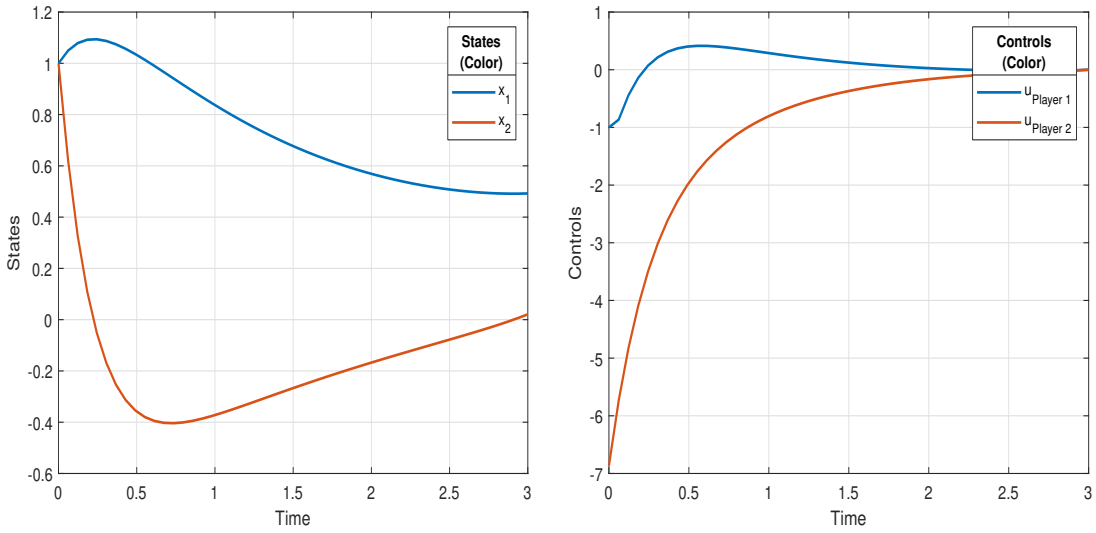


a. State

b. Controls

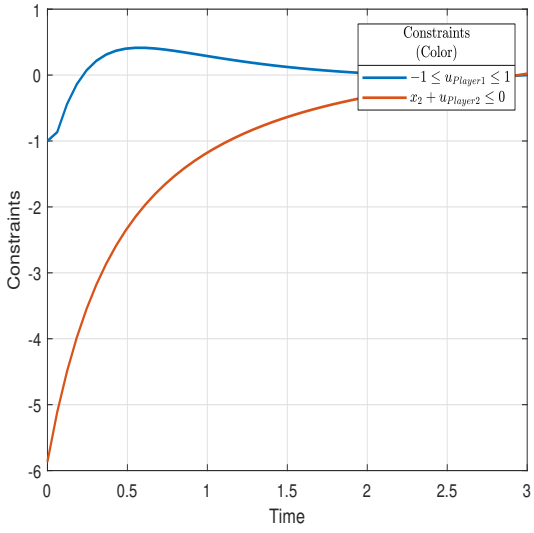
c. Constraint

Figure 6.1: This figure plots the state (a.) and control (b.) trajectories for the duopolistic competition. As can be seen, the constraint (c.) is satisfied throughout the trajectory.



a. States

b. Controls



c. Constraints

Figure 6.2: This figure plots the state (a.) and control (b.) trajectories for the nonlinear polynomial game. As can be seen, the constraints (c.) are satisfied throughout the trajectory.

Chapter 7

Inverse Dynamic Games

Since non-zero sum behaviors often arise in real-world settings, control systems that interact with humans should ideally account for this non-zero sum behavior. In particular, ideal controllers would exploit cooperation while protecting against adversarial behavior. However, a challenge that immediately arises is that the costs of interacting agents are often not known. In order to design control systems that account for cooperation and competition, systematic methods for modeling the costs would be beneficial.

Inverse dynamic games is the analogous problem to inverse optimal control, only in this, multiple objectives as well as multiple players may be involved. In this chapter, we present a general framework to solve nonzero-sum inverse dynamic game problems by showing that they can be reduced to decoupled residual minimization problems of individual players.

7.1 Problem Formulation for Inverse Differential Games

Recall from the previous chapter that a dynamic game problem can be stated as:

$$\begin{aligned} J_i(u_i, u_{-i}) = \underset{u_i}{\text{minimize}} & \int_{t_0}^{t_f} c_i^T \phi_i(t, x(t), u_1(t), ..., u_N(t))\, dt \\ s.t. \qquad & \dot{x}(t) = f(t, x(t), u_i(t), u_{-i}(t)) \\ & g_i(t, x(t), u_i(t), u_{-i}(t)) \leq 0 \\ & x(0) = x_{start} \\ & x(t_f) = free \end{aligned} \tag{7.1}$$

where $i \in \{1, ..., N\}$ are the N players with their associated cost functions J_i, u_i is the control action of player i and u_{-i} is the control action of every other player. All the players are subjected to the dynamics f and constraints g_i.

Since at a Nash equilibrium, every player i must simultaneously minimize their own performance index J_i, we use the necessary conditions (P1-P6) listed in the previous chapter to formulate an inverse dynamic game problem.

The co-state equations for each player can be written more explicitly as

$$\begin{aligned} -\dot{p}_i(t) &= c_i^T \nabla_x \phi(t, x, u_i, u_{-i}) + p_i^T \nabla_x f(t, x, u_i, u_{-i}) \mu_i^T \nabla_x g(t, x, u_i, u_{-i}) \\ p_i(t_f) &= 0 \end{aligned} \tag{7.2}$$

$$0 = c_i^T \nabla_{u_i} \phi_i(t, x, u_i, u_{-i}) + p_i^T \nabla_{u_i} f(t, x, u_i, u_{-i}) + \mu_i^T \nabla_{u_i} g_i(t, x, u_i, u_{-i}) \tag{7.3}$$

Now that we have necessary optimality conditions for every player in the game, setting

$$\dot{p}_i = v_i(t) \qquad z_i(t) = \begin{bmatrix} c_i \\ p_i(t) \\ \mu_i(t) \end{bmatrix} \tag{7.4}$$

the residual function in the case of dynamic games can be defined as (analogous to the case of optimal control)

$$r_i(z_i(t), v_i(t)) = \begin{bmatrix} \nabla_x \phi_i \Big|^T_{(x,u_i,u_{-i})} & \nabla_x f \Big|^T_{(x,u_i,u_{-i})} & \nabla_x g_i \Big|^T_{(x,u_i,u_{-i})} \\ \nabla_{u_i} \phi \Big|^T_{(x,u,u_{-i})} & \nabla_{u_i} f \Big|^T_{(x,u_i,u-i)} & \nabla_{u_i} g_i \Big|^T_{(x,u_i,u_{-i})} \end{bmatrix} z_i(t) + \begin{bmatrix} I \\ 0 \end{bmatrix} v_i(t) \tag{7.5}$$

Since each player independently minimizes its own performance index, the residual function of every player, as defined by (7.5), is minimized independently too. This is a key insight in our development of the inverse dynamic games method. *No matter how strongly coupled the dynamic game is, the performance index of every player can be computed by solving decoupled residual minimization problems.*

More formally, the performance index of every player i can be imputed by solving the corresponding residual optimization problem:

$$
\begin{aligned}
&\underset{z_i(t),v_i(t)}{\text{minimize}} \int_{t_0}^{t_f} ||r_i(z_i(t), v_i(t))||^2 \, dt \\
&\quad s.t. \quad \dot{p}_i(t) = v_i(t) \\
&\qquad\quad p_i(t_f) = 0 \\
&\qquad\quad \mu_i(t) \geq 0 \\
&\qquad\quad \mu_i(t)^T g_i(t, x(t), u_i(t)) = 0
\end{aligned}
\tag{7.6}
$$

Note again that this is a convex optimal control problem with input $v_i(t)$ and state $z_i(t)$ and can be solved efficiently. This residual minimization problem can also be extended to handle multiple observations of a given player by following similar developments as shown in the case of inverse optimal control method (5.13).

7.2 Simulations

This section presents the same example problems from the previous chapter, but this time they will be used to implement inverse dynamic game method discussed in this chapter. A description of the two examples is provided again for convenience.

1) Duopolistic competition

$$
\begin{aligned}
&\min_{u_1} J_1 = \frac{1}{2} \int_0^3 (x^2 + 2u_1^2) \, dt \\
&\min_{u_2} J_2 = \frac{1}{2} \int_0^3 (4x^2 + u_2^2) \\
&s.t. \quad \dot{x} = x + u_1 + u_2 \\
&\qquad x + u_1 + 5 \leq 0 \\
&\qquad x(0) = x_0 \\
&\qquad x(3) = x_{free}
\end{aligned}
\tag{7.7}
$$

2) Nonlinear polynomial game

$$\begin{aligned}
&\min_{u_1} J_1 = \frac{1}{2}\int_0^3 (2x_1^2 + 7x_2^2 + u_1^2 + u_2^2)\, dt \\
&\min_{u_2} J_2 = \frac{1}{2}\int_0^3 (3x_1^2 + 8x_2^2 + u_1^2 + u_2^2)\, dt \\
&s.t. \qquad \dot{x} = \begin{bmatrix} x_2 \\ x_1^2 + u_1 + u_2 \end{bmatrix} \\
&\qquad -1 \leq u_1 \leq 1 \\
&\qquad x_2 + u_2 \leq 1 \\
&\qquad x(0) = x_0 \\
&\qquad x(3) = x_{free}
\end{aligned} \tag{7.8}$$

7.3 Results and Discussion

Note: Both example problems presented here are solved using MATLAB's *fmincon* function with SQP solver.

Figure 7.1 a,b. show the state and control trajectories for the duopolistic game. As we can see, the predicted state and control trajectories virtually lie on top of the actual simulated trajectories. As a check on the validity of the inverse dynamic game method, we can also see that dual constraint (c.) as well as complementary slackness condition (d.) are satisfied.

Figure 7.2 a,b. show the state and control trajectories for the non-linear polynomial game. Here again, the predicted trajectories are in close agreement with the actual simulated trajectories. As a check on the validity of the IOC method, we can see that dual constraint condition (c.) is satisfied. However, the same cannot be said for the complementary slackness condition (d.) as it is violated in the very beginning. Looking at how accurately the inverse game method predicts the state and control trajectory, as well as the vanishing value of the residual function (Table 7.1) this constraint violation can be attributed to numerical inaccuracies in the implementation. Another important thing to note is that the box constraint in this problem is handled by breaking down the constraint $g_{Player1}$into two separate constraints $g_{1,Player1}, g_{2,Player2}$.

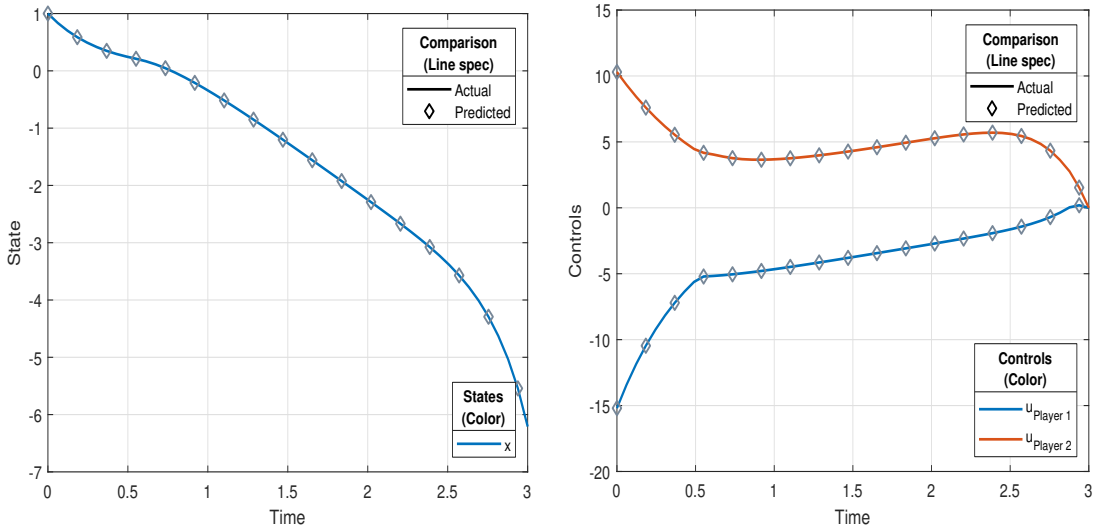
While Figures 7.1-7.2 provide visual description of the results, Table 7.1 provides a quantitative measure of the performance of the inverse dynamic game method, which is basically multiple call to the inverse optimal control method as had been shown in the chapter. It compares the

Table 7.1: Comparison of weight vector from forward and inverse dynamic games

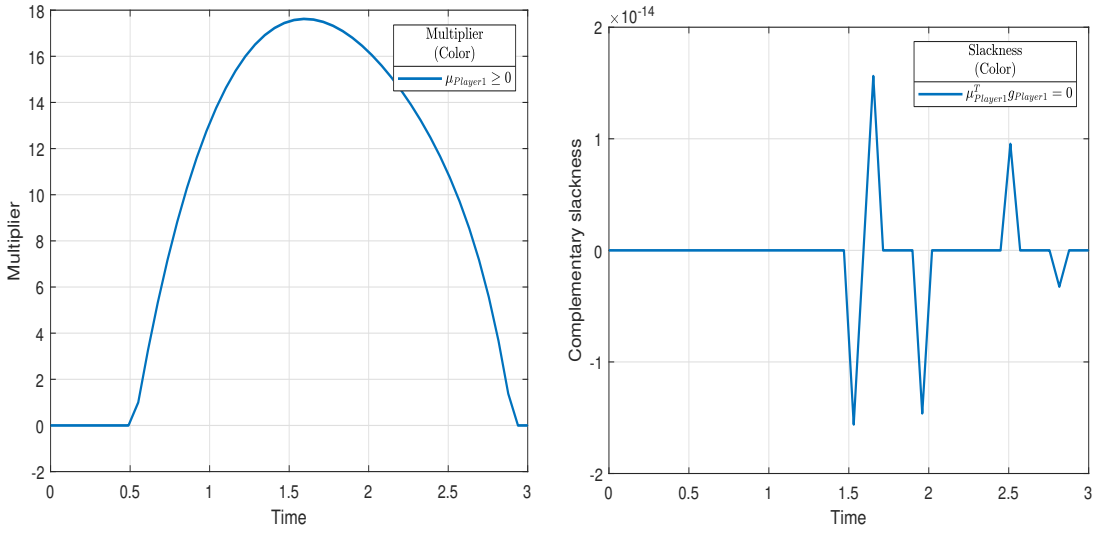
System	**Feature vector (ϕ)**	**Weight vector (from forward method) ($c = c_{true}$)**	**Learned weight vector (from inverse method) ($c = c_{learned}$)**	**Residual function minimization (r)**
Duopolistic competition	Player 1: $\phi_1 = [x^2, u_1^2]$ Player 2: $\phi_2 = [x^2, u_2^2]$	Player 1: [1, 2] Player 2: [4, 1]	Player 1: [1, 2] Player 2: [3.9997, 1]	Player 1: 1.04×10^{-10} Player 2: 9.702×10^{-07}
Nonlinear polynomial game	Player 1: $\phi_1 = [x_1^2, x_2^2, u_1^2, u_2^2]$ Player 2: $\phi_2 = [x_1^2, x_2^2, u_1^2, u_2^2]$	Player 1: [2, 7, 1, 1] Player 2: [3, 8, 1, 1]	Player 2: [2, 7, 1, 1] Player 2: [3, 8, 1, 1]	Player 1: 6.524×10^{-11} Player 2: 3.443×10^{-12}

true cost functions and the learned cost functions and also provides a value for the residual function. The quality of the solution can be judged from its value of final residual function: the lower the value of the residual, the better IOC method was able to recover the underlying cost function.

Much of discussion from Chapter 5 about the inverse optimal control method applies equally well in the case of inverse dynamic game method and is not repeated here to minimize redundancy.

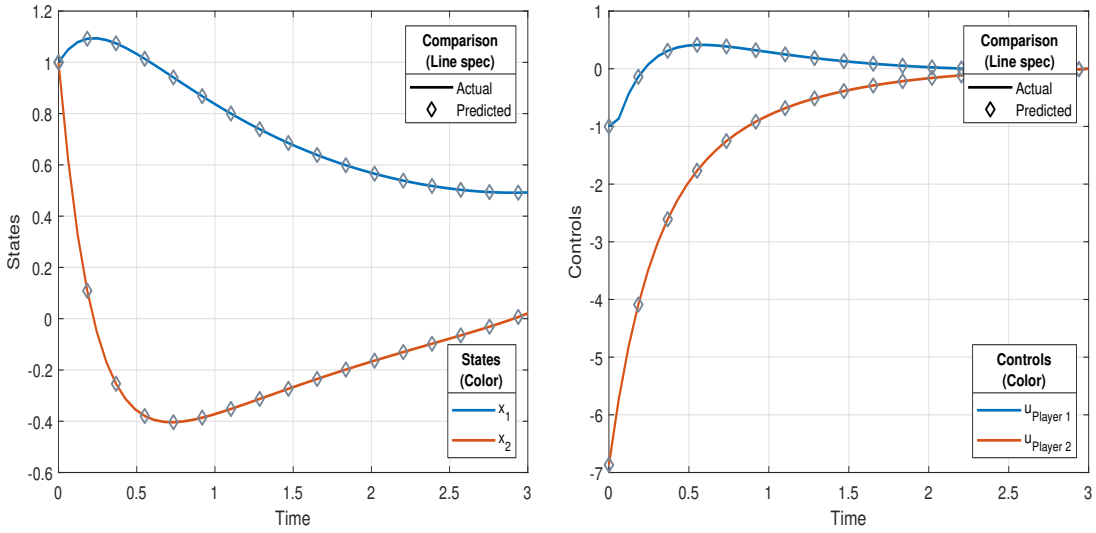


a. States b. Control



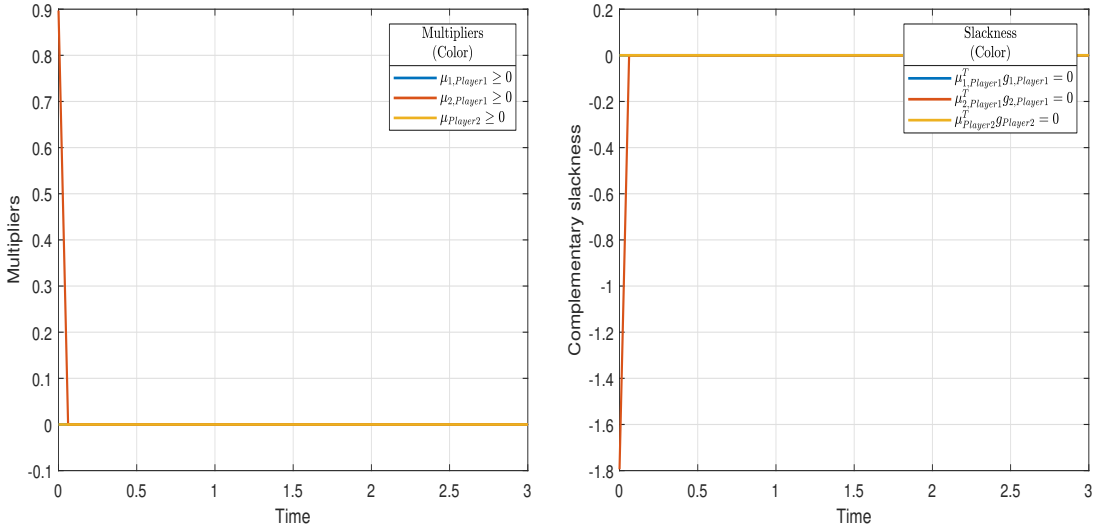
c. Dual feasibility d. Complementary slackness

Figure 7.1: This figure compares the state (a.) and control (b.) trajectories for the duopolistic competition problem computed using true (actual) and learned (predicted) cost functions. As can be seen, the computed costs are sufficiently accurate to reproduce the original trajectories. We also see that the dual feasibility constraint (c.) as well complementary slackness (d.) are satisfied. Here constraint $g = x + u_1 + 5$.



a. States

b. Control



c. Dual feasibility

d. Complementary slackness

Figure 7.2: This figure compares the state (a.) and control (b.) trajectories for the nonlinear polynomial game computed using true (actual) and learned (predicted) cost functions. As can be seen, the computed costs are sufficiently accurate to reproduce the original trajectories. We see that while the dual feasibility constraint (c.) is satisfied throughout the trajectory, the same cannot be said for complementary slackness (d.), which is violated at the very beginning of the trajectory. Here constraint $g_{1,Player1} = -u_{Player1} - 1$, $g_{2,Player2} = u_{Player1} - 1$, $g_{Player2} = x_2 + u_{Player2} - 1$

Chapter 8

Conclusion and Future Work

In this thesis, we presented a control-theoretic method of solving constrained inverse optimal control problems. We also provided a solution strategy, in the form of novel semi-direct method, to solve constrained dynamic games. Finally, we extended the constrained optimal control method to solve constrained inverse dynamic game problems. Throughout our work, we provided several examples to test the efficacy of our methods, both visually (via plots) and quantitatively (via tables).

We began by deriving necessary conditions of optimality for a nonlinear optimal control problem using calculus of variations in Chapter 3. We then brought up the fact that almost all nonlinear optimal control problems cannot be solved analytically or in closed-form, so we presented two methods, namely, trapezoidal collocation and pseudospectral collocation, for numerical solutions of optimal control problems in Chapter 4. These two chapters had in mind a beginner student or a non-expert who'll able to read and understand the text without having to spend an enormous amount of time understanding technicalities and heavy jargon present in the current literature. The examples provided at the end of Chapter 4 were not only used for demonstrating the results from implementing the two numerical schemes, but were also used for generating optimal trajectory data which was used in Chapter 5. In Chapter 5, we extended the method of Johnson et al. [18] to impute cost functions from optimal trajectory data when the dynamic systems were subjected to mixed and/or control constraints. The examples provided at the end of the chapter not only accurately predicted the underlying cost function, but the residual function associated with problem gave us a metric to judge the quality of the solution.

Chapter 6 extended single objective optimal control problem to multi-objective dynamic game problem and provided a novel method for computing Nash equilibrium, which is regarded as an optimal strategy. This method, called as semi-direct method, used necessary conditions of optimality developed in Chapter 3 and turned them into a feasibility problem for the dynamic game. The feasibility problem is then solved using a standard NLP solver. We again employed the examples used in this chapter not only for the purpose of demonstration of the efficacy of the method, but also to generate optimal trajectory data to be used in Chapter 7. Finally, in Chapter 7, we showed that the inverse dynamic game problem can be reduced to decoupled inverse optimal control problems. These problems can then be solved using our extended inverse optimal control method developed in Chapter 5.

While the work presented in this thesis addresses some of the gaps mentioned in the introductory chapter (Chapter 1), there are some limitations to it. Firstly, in Chapter 4, we assume that the solution which results from solving an optimal control problem is a minimal solution. However, because we only used the necessary conditions of optimality without taking into consideration sufficient conditions, we cannot guarantee that this solution is necessarily a minimum. Again, in Chapter 6, we assume that the solution that results from solving the optimization problem is a Nash equilibrium. However, without establishing the sufficient conditions, we cannot guarantee that this is equilibrium is necessarily a minimum of the problem. Moreover, we do not discuss whether this Nash equilibrium is always possible nor do we talk about its uniqueness. Finally, the Chapters 6-7 only talk about nonzero-sum dynamic games.

For future work, we would like to study stochastic systems as well as systems with partially known dynamics, neither of which is dealt in this work. We would also like to use the methods we developed in this thesis to important systems such as swarm navigation of human drivers. Lastly, we would like to extend our nonzero-sum dynamic framework to solve zero-sum dynamic game problems.

References

[1] Nazareth S Bedrossian, Sagar Bhatt, Wei Kang, and I Michael Ross. Zero-propellant maneuver guidance. *IEEE Control Systems Magazine*, 29(5):53–73, 2009.

[2] Christopher L Darby and Anil V Rao. Minimum-fuel low-earth orbit aeroassisted orbital transfer of small spacecraft. *Journal of Spacecraft and Rockets*, 48(4):618–628, 2011.

[3] Ilse Y Smets, Johan E Claes, Eva J November, Georges P Bastin, and Jan F Van Impe. Optimal adaptive control of (bio) chemical reactors: past, present and future. *Journal of process control*, 14(7):795–805, 2004.

[4] Filip Logist, PMM Van Erdeghem, and JF Van Impe. Efficient deterministic multiple objective optimal control of (bio) chemical processes. *Chemical Engineering Science*, 64(11):2527–2538, 2009.

[5] Sterling J Anderson, Steven C Peters, Tom E Pilutti, and Karl Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *International Journal of Vehicle Autonomous Systems*, 8(2-4):190–216, 2010.

[6] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[7] Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143. Springer, 1993.

[8] Divya Garg, Michael A Patterson, Camila Francolin, Christopher L Darby, Geoffrey T Huntington, William W Hager, and Anil V Rao. Direct trajectory optimization and

costate estimation of finite-horizon and infinite-horizon optimal control problems using a radau pseudospectral method. *Computational Optimization and Applications*, 49(2):335–358, 2011.

[9] AE Bryson, YC Ho, and GM Siouris. Applied optimal control: optimization, estimation, and control, ieee trans. syst., man. *Cybernet*, 9(6):366–367, 1979.

[10] Angelo Miele. Gradient algorithms for the optimization of dynamic systems. In *control and Dynamic systems*, volume 16, pages 1–52. Elsevier, 1980.

[11] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[12] Francesco Borrelli, Mato Baotić, Alberto Bemporad, and Manfred Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, 2005.

[13] Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous robots*, 28(3):369–383, 2010.

[14] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[15] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

[16] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[17] Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *Intelligent Control (ISIC), 2011 IEEE International Symposium on*, pages 613–619. IEEE, 2011.

[18] Miles Johnson, Navid Aghasadeghi, and Timothy Bretl. Inverse optimal control for deterministic continuous-time nonlinear systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 2906–2913. IEEE, 2013.

[19] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[20] Finn Kydland. Noncooperative and dominant player solutions in discrete dynamic games. *International economic review*, pages 321–335, 1975.

[21] Y Ho, A Bryson, and Sheldon Baron. Differential games and optimal pursuit-evasion strategies. *IEEE Transactions on Automatic Control*, 10(4):385–389, 1965.

[22] Tal Shima and Oded M Golan. Linear quadratic differential games guidance law for dual controlled missiles. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):834–842, 2007.

[23] Immanuel M Bomze. Non-cooperative two-person games in biology: A classification. *International journal of game theory*, 15(1):31–57, 1986.

[24] Dario Bauso. *Game theory with engineering applications*, volume 30. Siam, 2016.

[25] JC Engwerda et al. Feedback nash equilibria for linear quadratic descriptor differential games. *Automatica*, 48(4):625–631, 2012.

[26] Z Nikooeinejad, A Delavarkhalafi, and M Heydari. Journal of Computational and Applied A numerical solution of open-loop Nash equilibrium in nonlinear differential games based on Chebyshev pseudospectral method. *Journal of Computational and Applied Mathematics*, 300:369–384, 2016.

[27] M Cody Priess, Richard Conway, Jongeun Choi, John M Popovich, and Clark Radcliffe. Solutions to the inverse lqr problem with application to biological systems analysis. *IEEE Transactions on control systems technology*, 23(2):770–777, 2014.

[28] Jairo Inga, Esther Bischoff, Timothy L Molloy, Michael Flad, and Sören Hohmann. Solution sets for inverse non-cooperative linear-quadratic differential games. *IEEE Control Systems Letters*, 2019.

[29] Timothy L Molloy, Jason J Ford, and Tristan Perez. Inverse noncooperative differential games. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5602–5608. IEEE, 2017.

[30] Anil V Rao. Trajectory optimization: a survey. In *Optimization and optimal control in automotive systems*, pages 3–21. Springer, 2014.

[31] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance control and dynamics*, 21(2):193–207, 1998.

[32] JT Betts, SL Campbell, and A Engelsone. Direct transcription solution of optimal control problems with higher order state constraints: theory vs practice. *Optimization and Engineering*, 8(1):1–19, 2007.

[33] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[34] Douglas G Fox and Steven A Orszag. Pseudospectral approximation to two-dimensional turbulence. *Journal of Computational Physics*, 11(4):612–619, 1973.

[35] Steven A Orszag. Comparison of pseudospectral and spectral approximation. *Studies in Applied Mathematics*, 51(3):253–259, 1972.

[36] Gamal Elnagar, Mohammad A Kazemi, and Mohsen Razzaghi. The pseudospectral legendre method for discretizing optimal control problems. *IEEE transactions on Automatic Control*, 40(10):1793–1796, 1995.

[37] Gamal N Elnagar and Mohammad A Kazemi. Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems. *Computational Optimization and Applications*, 11(2):195–217, 1998.

[38] I Michael Ross and Fariba Fahroo. Legendre pseudospectral approximations of optimal control problems. In *New trends in nonlinear dynamics and control and their applications*, pages 327–342. Springer, 2003.

[39] I Michael Ross and Fariba Fahroo. Pseudospectral knotting methods for solving nonsmooth optimal control problems. *Journal of Guidance, Control, and Dynamics*, 27(3):397–405, 2004.

[40] I Michael Ross and Mark Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182–197, 2012.

[41] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[42] Rudolf Emil Kalman. When is a linear control system optimal. *Journal of Basic Engineering*, 86(1):51–60, 1964.

[43] Nathan Ratliff, Brian Ziebart, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Inverse optimal heuristic control for imitation learning. In *Artificial Intelligence and Statistics*, pages 424–431, 2009.

[44] Adina M Panchea and Nacim Ramdani. Towards solving inverse optimal control in a bounded-error framework. In *American Control Conference (ACC), 2015*, pages 4910–4915. IEEE, 2015.

[45] Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *The International Journal of Robotics Research*, 36(13-14):1474–1488, 2017.

[46] Ilan Rusnak. The lady, the bandits, and the bodyguards–a two team dynamic game. In *Proceedings of the 16th world IFAC congress*, pages 934–939, 2005.

[47] Oleg Prokopov and Tal Shima. Linear quadratic optimal cooperative strategies for active aircraft protection. *Journal of Guidance, Control, and Dynamics*, 36(3):753–764, 2013.

[48] Eloy Garcia, David W Casbeer, Khanh Pham, and Meir Pachter. Cooperative aircraft defense from an attacking missile. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 2926–2931. IEEE, 2014.

[49] Fouad El Ouardighi, Steffen Jørgensen, and Federico Pasin. A dynamic game with monopolist manufacturer and price-competing duopolist retailers. *OR spectrum*, 35(4):1059–1084, 2013.

[50] Nathanaël Jarrassé, Themistoklis Charalambous, and Etienne Burdet. A framework to describe, analyze and generate interactive motor behaviors. *PloS one*, 7(11):e49945, 2012.

[51] Y Li, G Carboni, F Gonzalez, D Campolo, and E Burdet. Differential game theory for versatile physical human–robot interaction. *Nature Machine Intelligence*, 1(1):36, 2019.

[52] Nan Li, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck Girard. Game theory-based traffic modeling for calibration of automated driving algorithms. In *Control Strategies for Advanced Driver Assistance Systems and Autonomous Driving Functions*, pages 89–106. Springer, 2019.

[53] Quanyan Zhu, Zhu Han, and Tamer Başar. A differential game approach to distributed demand side management in smart grid. In *Communications (ICC), 2012 IEEE International Conference on*, pages 3345–3350. IEEE, 2012.

[54] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pages 3909–3917, 2016.

[55] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, 59(4):1966–1980, 2013.

[56] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[57] Bernard Dacorogna. *Introduction to the Calculus of Variations*. World Scientific Publishing Company, 2014.

[58] Richard F Hartl, Suresh P Sethi, and Raymond G Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM review*, 37(2):181–218, 1995.

[59] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[60] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[61] Divya Garg, Michael Patterson, William W Hager, Anil V Rao, David A Benson, and Geoffrey T Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851, 2010.

[62] Victor M Becerra. Psopt optimal control solver user manual. *University of Reading*, 2010.

[63] Divya Garg, Michael Patterson, William Hager, Anil Rao, David Benson, and Geoffrey Huntington. An overview of three pseudospectral methods for the numerical solution of optimal control problems. 2017.

Appendix A

Glossary

While care has been taken in this thesis to not excessively rely on the use of jargon, this cannot always be achieved. This appendix defines jargon terms in a glossary.

- **Calculus of variations** – A branch of mathematics that deals with function minimization.
- **Direct transcription** – A procedure to convert a continuous time infinite dimensional functional optimization problem to a finite dimensional parameter optimization problem.
- **Feature vector** – Part of the cost function that contains different features such as squares of states, controls, etc.
- **Functional** – A functional whose input is another function (in contrast to a function whose input is a point)
- **Nash equilibrium** – A strategy in non-cooperative zero-sum games where no player can do better by unilaterally changing their strategy given the strategy of every other player.
- **Parameter optimization** – A problem which involves optimization over decision variables which are typically real numbers (in contrast to optimization over functions).
- **Sequential Quadratic Programming** – A nonlinear optimization solver which is used after a problem is transcribed from functional optimization to parameter optimization.
- **Weight vector** – Part of the cost function that indicates relative importance of different features present in the feature vector.