AA203: Optimal and Learning-based Control
Combined Course Notes

James Harrison*

May 16, 2019

Introduction

These notes accompany the newly revised (Spring 2019) version of AA203 at Stanford. The goal of this new course is to present a unified treatment of optimal control and reinforcement learning (RL), with an emphasis on model-based reinforcement learning. The goal of the instructors is to unify the subjects as much as possible, and to concretize connections between these research communities.

How is this course different from a standard class on Optimal Control? First, we will emphasize practical computational tools for real world optimal control problems, such as model predictive control and sequential convex programming. Beyond this, the last third of the course focuses on the case in which an exact model of the system is not available. We will discuss this setting both in the online context (typically referred to as adaptive optimal control) and in the episodic context (the typical setting for reinforcement learning).

How is this course different from a standard class on Reinforcement Learning? Many classes on reinforcement learning focus primarily on the setting of discrete Markov Decision Processes (MDPs), whereas we will focus primarily on continuous MDPs. More importantly, the focus on discrete MDPs leads planning with a known model (which is typically referred to as "planning" or "control" in RL) to be relatively simple. In this course, we will spend considerably more time focusing on planning with a known model in both continuous and discrete time. Finally, the focus of this course will primarily be on model-based methods. We will touch briefly on model-free methods at the end, and combinations of model-free and model-based approaches.

*Contact: jharrison@stanford.edu

A Note on Notation

The notation and language used in the control theory and reinforcement learning communities vary substantially, as so we will state all of the notational choices we make in this section. First, optimal control problems are typically stated in terms of minimizing a cost function, whereas reinforcement learning problems aim to maximize a reward. These are mathematically identical statements, where one is simply the negation of the other. Herein, we will use the control theoretic approach of cost minimization. We write c for the cost function, f for the system dynamics, and denote the state and action at time t as $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ respectively. We write scalars as lower case letters, vectors as bold lower case letters, and matrices as upper case letters. We write a deterministic policy as $\pi(\boldsymbol{x})$, and a stochastic policy as $\pi(\boldsymbol{u} \mid \boldsymbol{x})$. We write the cost-to-go (negation of the value function) associated with policy π at time t and state $\boldsymbol{x}$ as $J_t^\pi(\boldsymbol{x})$. We will also sometimes refer to the cost-to-go as the value, but in these notes we are always referring to the expected sum of future costs. For an in-depth discussion of the notational and language differences between the artificial intelligence and control theory communities, we refer the reader to [Pow12].

For notational convenience, we will write the Hessian of a function $f(x)$, evaluated at x^*, as $\nabla^2 f(x^*)$.

Prerequisites

While these notes aim to be almost entirely self contained, familiarity with undergraduate level calculus, differential equations, and linear algebra (equivalent to CME 102 and EE 263 at Stanford) are assumed. We will briefly review nonlinear optimization in the first section of these notes, but previous experience with optimization (e.g. EE 364A) will be helpful. Finally, previous experience with machine learning (at the level of CS 229) is beneficial.

Contents

1 Nonlinear Optimization

In this section we discuss the generic nonlinear optimization problem that forms the basis for the rest of the material presented in this class. We write the minimization problem as

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad f(\boldsymbol{x})$$

where f is the cost function, usually assumed twice continuously differentiable, $x \in \mathbb{R}^n$ is the optimization variable, and $\mathcal{X} \subset \mathbb{R}^n$ is the constraint set. The special case in which the cost function is linear and the constraint set is specified by linear equations and/or inequalities is *linear optimization*, which we will not discuss.

1.1 Unconstrained Nonlinear Optimization

We will first address the unconstrained case, in which $\mathcal{X} = \mathbb{R}^n$. A vector $\boldsymbol{x}^*$ is said to be an unconstrained *local minimum* if there exists $\epsilon > 0$ such that $f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}^*\| \le \epsilon\}$, and $\boldsymbol{x}^*$ is said to be an unconstrained *global minimum* if $f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$ for all $x \in \mathbb{R}^n$.

1.1.1 Necessary Conditions for Optimality

For a differentiable cost function, we can compare the cost of a point to its neighbors by considering a small variation $\Delta\boldsymbol{x}$ from $\boldsymbol{x}^*$. By using Taylor expansions, this yields a first order cost variation

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)^T \Delta\boldsymbol{x} \tag{1}$$

and a second order cost variation

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)^T \Delta\boldsymbol{x} + \frac{1}{2}\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}^*) \Delta\boldsymbol{x}. \tag{2}$$

Setting $\Delta\boldsymbol{x}$ to be equal to positive and negative multiples of the unit coordinate vector, we have

$$\frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \ge 0 \tag{3}$$

where x_i denotes the i'th coordinate of $\boldsymbol{x}$, and

$$\frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \le 0 \tag{4}$$

for all i, which is only satisfied by $\nabla f(\boldsymbol{x}^*) = 0$. This is referred to as the *first order necessary condition for optimality.* Looking at the second order variation, and noting that $\nabla f(\boldsymbol{x}^*) = 0$, we expect

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \ge 0 \tag{5}$$

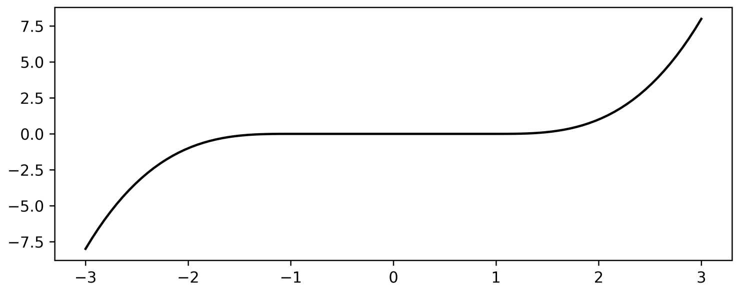


Figure 1: An example of a function for which the necessary conditions of optimality are satisfied but the sufficient conditions are not.

and thus

$$\Delta \boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}^*) \Delta \boldsymbol{x} \geq 0 \tag{6}$$

which implies $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite. This is referred to as the *second order necessary condition for optimality.* Stating these conditons formally,

Theorem 1.1 (Necessary Conditions for Optimality (NOC))**.** *Let $\boldsymbol{x}^*$ be an unconstrained local minimum of $f : \mathbb{R}^n \to \mathbb{R}$ and $f \in C^1$ in an open set S containing $\boldsymbol{x}^*$. Then,*

$$\nabla f(\boldsymbol{x}^*) = 0. \tag{7}$$

If $f \in C^2$ within S, $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite.

Proof. See section 1.1 of [Ber16].

1.1.2 Sufficient Conditions for Optimality

If we strengthen the second order condition to $\nabla^2 f(\boldsymbol{x}^*)$ being positive definite, we have the sufficient conditions for $\boldsymbol{x}^*$ being a local minimum. Why is the second order necessary conditions not sufficient? An example function is given in figure 1. Formally,

Theorem 1.2 (Sufficient Conditions for Optimality (SOC))**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be C^2 in an open set S. Suppose a vector $\boldsymbol{x}^*$ satisfies the conditions $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then $\boldsymbol{x}^*$ is a strict unconstrained local minimum of f.*

Proof is again given in Section 1.1 of [Ber16]. There are several reasons why the optimality conditions are important. In a general nonlinear optimization setting, they can be used to filter candidates for global minima. They can be used for sensitivity analysis, in which the sensitivity of $\boldsymbol{x}^*$ to model parameters can be quantified [Ber16]. This is common in e.g. microeconomics. Finally, these conditions often provide the basis for the design and analysis of optimization algorithms.

1.1.3 Special case: Convex Optimization

A special case within nonlinear optimization is the set of *convex optimization* problems. A set $S \subset \mathbb{R}^n$ is called *convex* if

$$\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y} \in S, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in S, \forall \alpha \in [0,1]. \tag{8}$$

For S convex, a function $f : S \to \mathbb{R}$ is called convex if

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}). \tag{9}$$

This class of problems has several important characteristics. If f is convex, then

- A local minimum of f over S is also a global minimum over S. If in addition f is strictly convex (the inequality in (9) is strict), there exists at most one global minimum of f.
- If $f \in C^1$ and convex, and the set S is open, $\nabla f(\boldsymbol{x}^*) = 0$ is a necessary and sufficient condition for a vector $\boldsymbol{x}^* \in S$ to be a global minimum over S.

Convex optimization problems have several nice properties that make them (usually) computationally efficient to solve, and the first property above gives a certificate of having obtained global optimality that is difficult or impossible to obtain in the general nonlinear optimization setting. For a thorough treatment of convex optimization theory and algorithms, see [BV04].

1.1.4 Computational Methods

In this subsection we will discuss the class of algorithms known as *gradient methods* for finding local minima in nonlinear optimization problems. These approaches, rely (roughly) on following the gradient of the function "downhill", toward the minima. More concretely, these algorithms rely on taking steps of the form

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k \tag{10}$$

where if $\nabla f(\boldsymbol{x}) \neq 0$, $\boldsymbol{d}^k$ is chosen so that

$$\nabla f(\boldsymbol{x})^T \boldsymbol{d}^k < 0 \tag{11}$$

and $\alpha > 0$. Typically, the step size α^k is chosen such that

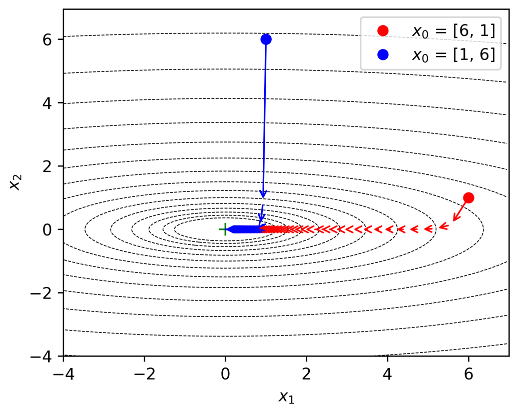
$$f(\boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k) < f(\boldsymbol{x}^k), \tag{12}$$

but generally, the step size and the direction of descent ($\boldsymbol{d}^k$) are tuning parameters.

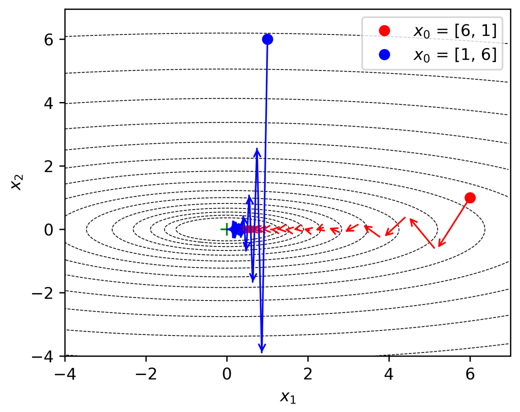
We will look at the general class of descent directions of the form

$$\boldsymbol{d}^k = -D^k \nabla f(\boldsymbol{x}^k) \tag{13}$$

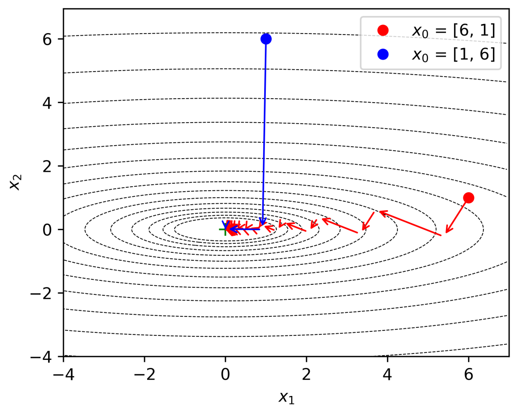
where $D^k > 0$ (note that this guarantees $\nabla f(\boldsymbol{x}^k)^T \boldsymbol{d}^k < 0$).



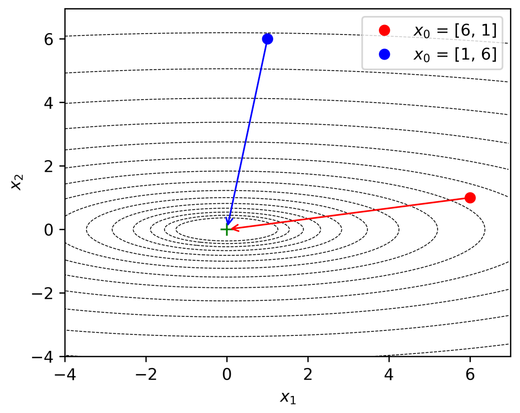
(a) Steepest descent, small fixed step size.



(b) Steepest descent, large fixed step size.



(c) Steepest descent, step size chosen via line search.



(d) Newton's method. Note that the method converges in one step.

Figure 2: Comparison of steepest descent methods with various step sizes, and Newton's method, on the same quadratic cost function.

Steepest descent, $D^k = I$. The simplest choice of descent direction is directly following the gradient, and ignoring second order function information. In practice, this often leads to slow convergence (figure 2a) and possible oscillation (figure 2b).

Newton's Method, $D^k = (\nabla^2 f(\boldsymbol{x}^k))^{-1}$. The underlying idea of this approach is to at each iteration, minimize the quadratic approximation of f around $\boldsymbol{x}^k$,

$$f^k(\boldsymbol{x}) = f(\boldsymbol{x}^k) + \nabla f(\boldsymbol{x}^k)^T(\boldsymbol{x} - \boldsymbol{x}^k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^k)^T \nabla^2 f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k). \tag{14}$$

Setting the derivative of this to zero, we obtain

$$\nabla f(\boldsymbol{x}^k) + \nabla^2 f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) = 0 \tag{15}$$

and thus, by setting $\boldsymbol{x}^{k+1}$ to be the $\boldsymbol{x}$ that satisfies the above, we get the

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - (\nabla^2 f(\boldsymbol{x}^k))^{-1}\nabla f(\boldsymbol{x}^k) \tag{16}$$

or more generally,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha(\nabla^2 f(\boldsymbol{x}^k))^{-1}\nabla f(\boldsymbol{x}^k). \tag{17}$$

Note that this update is only valid for $\nabla^2 f(\boldsymbol{x}^k) > 0$. When this condition doesn't hold, $\boldsymbol{x}^{k+1}$ is not a minimizer of the second order approximation (as a result of the SOCs). See figure 2d for an example where Newton's method converges in one step, as a result of the cost function being quadratic.

Diagonally scaled steepest descent, $D^k = \mathbf{diag}(d_1^k, \ldots, d_n^k)$. Have $d_i^k > 0 \forall i$. A popular choice is

$$d_i^k = \left(\frac{\partial^2 f(\boldsymbol{x}^k)}{\partial x_i^2}\right)^{-1} \tag{18}$$

which is a diagonal approximation of the Hessian.

Modified Newton's method, $D^k = (\nabla^2 f(\boldsymbol{x}^0))^{-1}$. Requires $\nabla^2 f(\boldsymbol{x}^0) > 0$. For cases in which one expects $\nabla^2 f(\boldsymbol{x}^0) \approx \nabla^2 f(\boldsymbol{x}^k)$, this removes having to compute the Hessian at each step.

In addition to choosing the descent direction, there also exist a variety of methods to choose the step size α. A computationally intensive but efficient (in terms of the number of steps taken) is using a minimization rule of the form

$$\alpha^k = \text{argmin}_{\alpha \geq 0} f(\boldsymbol{x}^k + \alpha \boldsymbol{d}^k) \tag{19}$$

which is usually solved via line search (figure 2c). Alternative approaches include a limited minimization rule, in which you constrain $\alpha^k \in [0, s]$ during the line search, or simpler approach such as a constant step size (which may not guarantee convergence), or a diminishing scheduled step size. In this last case, schedules are typically chosen such that $\alpha^k \to 0$ as $k \to \infty$, while $\sum_{k=0}^{\infty} \alpha^k = +\infty$.

1.2 Constrained Nonlinear Optimization

In this section we will address the general constrained nonlinear optimization problem,

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad f(\boldsymbol{x})$$

which may equivalently be written

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{X} \end{aligned}$$

where the set $\mathcal{X}$ is usually specified in terms of equality and inequality constraints. To operate within this problem structure, we will develop a set of optimality conditions involving auxiliary variables called *Lagrange multipliers*.

1.2.1 Equality Constrained Optimization

We will first look at optimization with equality constraints of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & h_i(\boldsymbol{x}) = 0, \quad i = 1, \dots, m \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h_i : \mathbb{R}^n \to \mathbb{R}$ are C^1. We will write $\boldsymbol{h} = [h_1, \dots, h_m]^T$. For a given local minimum $\boldsymbol{x}^*$, there exist scalars $\lambda_1, \dots, \lambda_m$ called Lagrange multipliers such that

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\boldsymbol{x}^*) = 0. \tag{20}$$

There are several possible interpretations for Lagrange multipliers. First, note that the cost gradient $f(\boldsymbol{x}^*)$ is in the subspace spanned by the constraint gradients at $\boldsymbol{x}^*$. Equivalently, $f(\boldsymbol{x}^*)$ is orthogonal to the subspace of first order feasible variations

$$V(\boldsymbol{x}^*) = \{\Delta \boldsymbol{x} \mid \nabla h_i(\boldsymbol{x}^*)^T \Delta \boldsymbol{x} = 0, i = 1, \dots, m\}. \tag{21}$$

This subspace is the space of variations $\Delta \boldsymbol{x}$ for which $\boldsymbol{x} = \boldsymbol{x}^* + \Delta \boldsymbol{x}$ satisfies the constraint $\boldsymbol{h}(\boldsymbol{x}) = 0$ up to first order. Therefore, at a local minimum, the first order cost variation $\nabla f(\boldsymbol{x}^*)^T \Delta \boldsymbol{x}$ is zero for all variations $\Delta \boldsymbol{x}$ in this space.

Given this informal understanding, we may now precisely state the necessary conditions for optimality in constrained optimization.

Theorem 1.3 (NOC for equality constrained optimization)**.** *Let $\boldsymbol{x}^*$ be a local minimum of f subject to $\boldsymbol{h}(\boldsymbol{x}) = 0$, and assume that the constraint gradients $\nabla h_1(\boldsymbol{x}^*), \dots, \nabla h_m(\boldsymbol{x}^*)$ are linearly independent. Then there exists a unique vector $\boldsymbol{\lambda}^* = [\lambda_1^*, \dots, \lambda_m^*]^T$ called a Lagrange multiplier vector, such that*

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\boldsymbol{x}^*) = 0. \tag{22}$$

If in addition f and $\boldsymbol{h}$ are C^2, we have

$$\boldsymbol{y}^T(\nabla^2 f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla^2 h_i(\boldsymbol{x}^*))\boldsymbol{y} \geq 0, \quad \forall \boldsymbol{y} \in V(\boldsymbol{x}^*) \tag{23}$$

where

$$V(\boldsymbol{x}^*) = \{\boldsymbol{y} \mid \nabla h_i(\boldsymbol{x}^*)^T \boldsymbol{y} = 0, i = 1, \dots, m\}. \tag{24}$$

Proof. See [Ber16] Section 3.1.1 and 3.1.2. □

We will sketch two possible proofs for the NOC for equality constrained optimization.

Penalty approach. This approach relies on adding a to the cost function a large penalty term for constraint violation. This is the same approach that will be used in proving the necessary conditions for inequality constrained optimization, and is the basis of a variety of practical numerical algorithms.

Elimination approach. This approach views the constraints as a system of m equations with n unknowns, for which m variables can be expressed in terms of the remaining $m-n$ variables. This reduces the problem to an unconstrained optimization problem.

Note that in theorem 1.3, we assumed the gradients of the constraint functions were linearly independent. A feasible vector for which this holds is called *regular*. If this condition is violated, a Lagrange multiplier for a local minimum may not exist.

For convenience, we will write the necessary conditions in terms of the Lagrangian function $L : \mathbb{R}^{m+n} \to \mathbb{R}$,

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}). \tag{25}$$

This function allows the NOC conditions to be succinctly stated as

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{26}$$
$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{27}$$
$$\boldsymbol{y}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \boldsymbol{y} \geq 0, \quad \forall \boldsymbol{y} \in V(\boldsymbol{x}^*). \tag{28}$$

which form a system of $n+m$ equations with $n+m$ unknowns. Given this notation, we can state the sufficient conditions.

Theorem 1.4 (SOC for equality constrained optimization)**.** *Assume that f and $\boldsymbol{h}$ are C^2 and let $\boldsymbol{x}^* \in \mathbb{R}^n$ and $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ satisfy*

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{29}$$
$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{30}$$
$$\boldsymbol{y}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \boldsymbol{y} > 0, \quad \forall \boldsymbol{y} \neq 0, \boldsymbol{y} \in V(\boldsymbol{x}^*). \tag{31}$$

Proof. See [Ber16] Section 3.2. □

Note that the SOC does not include regularity of $\boldsymbol{x}^*$.

1.2.2 Inequality Constrained Optimization

We will now address the general case, including inequality constraints,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & h_i(\boldsymbol{x}) = 0, \quad i = 1, \dots, m \\ & g_j(\boldsymbol{x}) \leq 0, \quad j = 1, \dots, r \end{aligned}$$

where f, h_i, g_i are C^1. The key intuition for the case of inequality constraints is based on realizing that for any feasible point, some subset of the constraints will be active (for which $g_j(\boldsymbol{x}) = 0$), while the complement of this set will be inactive. We define the active set of inequality constraints, which we denote

$$A(\boldsymbol{x}) = \{j \mid g_j(\boldsymbol{x}) = 0\}. \tag{32}$$

A constraint is active at $\boldsymbol{x}$ if it is in $A(\boldsymbol{x})$, otherwise it is inactive. Note that if $\boldsymbol{x}^*$i is a local minimum of the inequality constrained problem, then $\boldsymbol{x}^*$ is a local minimum of the identical problem with the inactive constraints removed. Moreover, at this local minimum, the constraints may be treated as equality constraints. Thus, if $\boldsymbol{x}^*$ is regular, there exists Lagrange multipliers $\lambda_1^*, \dots, \lambda_m^*$ and $\mu_j^*, j \in A(\boldsymbol{x}^*)$ such that

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i \nabla h_i(\boldsymbol{x}^*) + \sum_{j \in A(\boldsymbol{x}^*)} \mu_j^* \nabla g_j(\boldsymbol{x}^*) = 0. \tag{33}$$

We will define the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^r \mu_j g_j(\boldsymbol{x}), \tag{34}$$

which we will use to state the necessary and sufficient conditions.

Theorem 1.5 (Karush-Kuhn-Tucker NOC)**.** *Let $\boldsymbol{x}^*$ be a local minimum for the inequality constrained problem where f, h_i, g_j are C^1 and assume $\boldsymbol{x}^*$ is regular (equality and active inequality constraint gradients are linearly independent). Then, there exists unique Lagrange multiplier vectors $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ such that*

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0 \tag{35}$$
$$\boldsymbol{\mu} \geq 0 \tag{36}$$
$$\mu_j^* = 0, \quad \forall j \notin A(\boldsymbol{x}^*) \tag{37}$$

If in addition, $f, \boldsymbol{h}, \boldsymbol{g}$ are C^2, we have

$$\boldsymbol{y}^T \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \boldsymbol{y} \geq 0 \tag{38}$$

for all $\boldsymbol{y}$ such that

$$\nabla h_i(\boldsymbol{x}^*)^T \boldsymbol{y} = 0, \quad i = 1, \dots, m \tag{39}$$
$$\nabla g_j(\boldsymbol{x}^*)^T \boldsymbol{y} = 0, \quad j \in A(\boldsymbol{x}^*) \tag{40}$$

Proof. See [Ber16] Section 3.3.1. □

The SOC are obtained similarly to the equality constrained case.

1.3 Further Reading

In this section we have addressed the necessary and sufficient conditions for constrained and unconstrained nonlinear optimization. This section is based heavily on [Ber16], and we refer the reader to this book for further details. We have avoided discussing linear programming, which is itself a large topic of study, about which many books have been written (we refer the reader to [BT97] as a good reference on the subject).

Convex optimization has become a powerful and widespread tool in modern optimal control. While we have only addressed it briefly here, [BV04] offers a fairly comprehensive treatment of the theory and practice of convex optimization. For a succinct overview with a focus on machine learning, we refer the reader to [Kol08].

2 Dynamic Programming and the Linear Quadratic Regulator

2.1 The Optimal Control Problem

In this section, we will outline the deterministic continuous-time optimal control problem that we will aim to solve. We will denote the state at time t as $\boldsymbol{x}(t) \in \mathbb{R}^n$, and the control as $\boldsymbol{u}(t) \in \mathbb{R}^m$. We will also occasionally write these as $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$, respectively. We will write the continuous-time systems dynamics as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t). \tag{41}$$

We will refer to a history of control input values during an interval $[t_0, t_f]$ as a control history, and we will refer to a history of state values over this interval as a state trajectory.

Different control problems may call for various constraints. For example, we may constrain a quadrotor to only fly in space not occupied by obstacles. Examples of constraints we will see are

- Initial and final conditions, $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$
- Trajectory constraints, $\underline{\boldsymbol{x}} \leq \boldsymbol{x}(t) \leq \bar{\boldsymbol{x}}$
- Control limits, $\underline{\boldsymbol{u}} \leq \boldsymbol{u}(t) \leq \bar{\boldsymbol{u}}$.

A state trajectory and control history that satisfy the constraints during the entire time interval $[t_0, t_f]$ are called admissible trajectories and admissible controls, respectively.

Finally, we will define the performance measure,

$$J = c_f(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{42}$$

where c is the instantaneous cost function, and c_f is the terminal state cost. We are now able to state the continuous-time optimal control problem. We aim to find an admissible control,

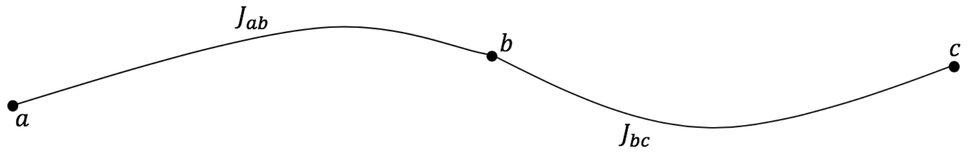


Figure 3: An optimal trajectory connecting point a to point c. There are no better (lower cost) trajectories than the sub-trajectory connecting b and c, by the principle of optimality.

$\boldsymbol{u}^*$, which causes the system (41) to follow an admissible trajectory, $\boldsymbol{x}^*$, that minimizes the performance measure given by (42). The minimizer $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ is called an optimal trajectory-control pair.

Note, first of all, that this is an extremely general problem formulation. We have not fixed our system dynamics, cost function, or specific constraints. We can't, in general, guarantee the existence or uniqueness of the optimal solution.

There are two possible solution forms for the optimal control. The first, $\boldsymbol{u}^* = e(\boldsymbol{x}(t_0), t)$ is referred to as an open-loop solution. This is an input function that is applied to the system, without using feedback. Practically, such solutions usually require augmentation with a feedback controller, as small model mismatch may lead to compounding errors. The second possible solution form is a feedback policy, $\boldsymbol{u}^* = \pi(\boldsymbol{x}(t), t)$. This feedback law maps all state-time pairs to an action and thus is usually more robust to possible model mismatch. However, depending on the particular problem formulation, open-loop solutions may be easier to compute.

2.2 Dynamic Programming and the Principle of Optimality

In this chapter we will outline the principle of optimality, and the method of dynamic programming (DP), one of two main approaches to solving the optimal control problem. The second, so-called variational approaches based on Pontryagin's Maximum Principle (PMP) will be discussed in future chapters. Dynamic programming has the strong advantage of yielding a feedback policy, however, exactly solving the dynamic programming problem is infeasible for many systems. We will address special cases in which the DP problem can be solved exactly, and approximate methods that work for a wide variety of systems.

Despite having just introduced the optimal control problem in continuous time, we will be operating in discrete time here, in which we aim to minimize

$$J_f(\boldsymbol{x}_0) = c_f(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} c(\boldsymbol{x}_k, \boldsymbol{u}_k, k). \tag{43}$$

We will extend the methods we develop in this chapter to continuous time in the next chapter.

The principle of optimality is as follows. Figure 3 shows a trajectory from point a to c. If the cost of the trajectory, $J_{ac} = J_{ab} + J_{bc}$, is minimal, then J_{bc} is also a minimum cost trajectory connecting b and c. The proof of this principle, stated informally, is simple.

Assume there exists an alternative trajectory connecting b and c, for which we will write the cost as $\tilde{J}_{bc}$, that achieves $\tilde{J}_{bc} < J_{bc}$. Then, we have

$$\begin{aligned}\tilde{J}_{ac} &= J_{ab} + \tilde{J}_{bc} && (44)\\ &< J_{ab} + J_{bc} && (45)\\ &= J_{ac}, && (46)\end{aligned}$$

and thus J_{ac} isn't minimal. More formally,

Theorem 2.1 (Discrete-time Principle of Optimality: Deterministic Case)**.** *Let $\pi^* = (\pi_0^*, \dots, \pi_{N-1}^*)$ be an optimal policy. Assume state $\boldsymbol{x}_k$ is reachable. Consider the subproblem whereby we are at $\boldsymbol{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $(\pi_k^*, \dots, \pi_{N-1}^*)$ is optimal for the subproblem.*

Dynamic programming, intuitively, proceeds backwards in time, first solving simpler shorter horizon problems. If we have found the optimal policy for times $k+1$ to $N-1$, along with the associated cost-to-go for each state, choosing the optimal policy for time k is a one step optimization problem. More concretely, we will assume we have dynamics of the form $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k)$ with $\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)$, and the cost given by (43). Then, dynamic programming iterates backward in time, from $N-1$ to 0, with

$$J_N(\boldsymbol{x}_N) = c_T(\boldsymbol{x}_N) \tag{47}$$

$$J_k(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)} \left\{ c_k(\boldsymbol{x}_k, \boldsymbol{u}_k, k) + J_{k+1}(f(\boldsymbol{x}_k, \boldsymbol{u}_k, k)) \right\}. \tag{48}$$

Note that here we have considered only deterministic dynamical systems (there is no stochastic disturbance). Equation (48) is one form of the *Bellman equation*, one of the most important relations in optimal control.

Dynamic programming raises many practical issues if one were to attempt to apply it directly. To perform the recursion, J_{k+1} must be known for all $\boldsymbol{x}_{k+1}$ (or more precisely, all $\boldsymbol{x}_{k+1}$ that are reachable from $\boldsymbol{x}_k$). If the state space is discrete (and relatively small), this is tractable as the cost-to-go may just be maintained in tabular form. In the next subsection, we will discuss an extremely important case in continuous space in which the cost-to-go can be computed exactly for all states. However, for general systems, we can not expect to be able to compute the cost-to-go for all states. Possible approaches to make the DP approach tractable are discretizing the state space, approximating the cost-to-go (i.e. restricting the family of functions that J_{k+1} may be in), or interpolating between cost-to-go computed for a finite set of states.

2.3 Discrete LQR

An important instance in which dynamic programming can be solved analytically for continuous state-action systems is the *linear quadratic regulator* problem. We will fix the dynamics of the system to be (possibly time-varying) linear,

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k \tag{49}$$

and the cost function as quadratic

$$c(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}(\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k) \tag{50}$$

$$c_N(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_N \boldsymbol{x}_k \tag{51}$$

where $Q_k \in \mathbb{R}^{n\times n}$ is positive semi-definite and $R_k \in \mathbb{R}^{m\times m}$ is positive definite for all $k = 0,\ldots,N$. Importantly, we assume $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are unconstrained for all k. To perform DP recursion, we initialize

$$J_N^*(\boldsymbol{x}_N) = \frac{1}{2}\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N := \frac{1}{2}\boldsymbol{x}_N^T V_N \boldsymbol{x}_N. \tag{52}$$

Then, applying (48), we have

$$J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \left\{\boldsymbol{x}_{N-1}^T Q_{N-1}\boldsymbol{x}_{N-1} + \boldsymbol{u}_{N-1}^T R_{N-1}\boldsymbol{u}_{N-1} + \boldsymbol{x}_N^T V_N \boldsymbol{x}_N\right\} \tag{53}$$

which, applying the dynamics,

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \{&\boldsymbol{x}_{N-1}^T Q_{N-1}\boldsymbol{x}_{N-1} + \boldsymbol{u}_{N-1}^T R_{N-1}\boldsymbol{u}_{N-1} \\ &+ (A_{N-1}\boldsymbol{x}_{N-1} + B_{N-1}\boldsymbol{u}_{N-1})^T V_N (A_{N-1}\boldsymbol{x}_{N-1} + B_{N-1}\boldsymbol{u}_{N-1})\}. \end{aligned} \tag{54}$$

Rearranging, we have

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \{&\boldsymbol{x}_{N-1}^T (Q_{N-1} + A_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \\ &+ \boldsymbol{u}_{N-1}^T (R_{N-1} + B_{N-1}^T V_N B_{N-1})\boldsymbol{u}_{N-1} \\ &+ 2\boldsymbol{u}_{N-1}^T (B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1}\}. \end{aligned} \tag{55}$$

Note that this optimization problem is convex in $\boldsymbol{u}_{N-1}$ as $R_{N-1} + B_{N-1}^T V_N B_{N-1} > 0$. Therefore, any local minima is a global minima, and therefore we can simply apply the first order optimality conditions. Differentiating,

$$\frac{\partial J_{N-1}^*}{\partial \boldsymbol{u}_{N-1}}(\boldsymbol{x}_{N-1}) = (R_{N-1} + B_{N-1}^T V_N B_{N-1})\boldsymbol{u}_{N-1} + (B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \tag{56}$$

and setting this to zero yields

$$\boldsymbol{u}_{N-1}^* = -(R_{N-1} + B_{N-1}^T V_N B_{N-1})^{-1}(B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \tag{57}$$

which we write

$$\boldsymbol{u}_{N-1}^* = L_{N-1}\boldsymbol{x}_{N-1} \tag{58}$$

which is a time-varying linear feedback policy. Plugging this feedback policy into (54),

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) =&\boldsymbol{x}_{N-1}^T (Q_{N-1} + L_{N-1}^T R_{N-1} L_{N-1} \\ &+ (A_{N-1} + B_{N-1}L_{N-1})^T V_N (A_{N-1} + B_{N-1}L_{N-1}))\boldsymbol{x}_{N-1}. \end{aligned} \tag{59}$$

Critically, this implies that the cost-to-go is always a positive semi-definite quadratic function of the state. Because the optimal policy is always linear, and the optimal cost-to-go is always quadratic, the DP recursion may be recursively performed backward in time and the minimization may be performed analytically.

Following the same procedure, we can write the DP recursion for the discrete-time LQR controller:

1. $V_N = Q_N$
2. $L_k = -(R_k + B_k^T V_{k+1} B_k)^{-1}(B_k^T V_{k+1} A_k)$
3. $V_k = Q_k + L_k^T R_k L_k + (A_k + B_k L_k)^T V_{k+1}(A_k + B_k L_k)$
4. $\boldsymbol{u}_k^* = L_k \boldsymbol{x}_k$
5. $J_k^*(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k$

There are several implications of this recurrence relation. First, even if A, B, Q, R are all constant (not time-varying), the policy is still time-varying. Why is this the case? Control effort invested early in the problem will yield dividends over the remaining length of the horizon, in terms of lower state cost for all future time steps. However, as the remaining length of the episode becomes shorter, this tradeoff is increasingly imbalanced, and the control effort will decrease. However, for a linear time-invariant system, if (A, B) is controllable, the feedback gain L_k approach a constant as the episode length approaches infinity. This time-invariant policy is practical for long horizon control problems, and may be approximately computed by running the DP recurrence relation until approximate convergence.

2.3.1 LQR with (Bi)linear Cost and Affine Dynamics

In the previous subsection, we have derived the common formulation of the LQR controller. In this subsection, we will derive the discrete time LQR controller for a more general system with bilinear/linear terms in the cost and affine terms in the dynamics. This derivation will be the basis of algorithms we will build up in the following subsections. More concretely, we consider systems with stage-wise cost

$$c(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k + \boldsymbol{u}_k^T H_k \boldsymbol{x}_k + \boldsymbol{q}_k^T \boldsymbol{x}_k + \boldsymbol{r}_k^T \boldsymbol{u}_k + q_k, \tag{60}$$

terminal cost

$$c_N(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_N \boldsymbol{x}_k + \boldsymbol{q}_N^T \boldsymbol{x}_k + q_N, \tag{61}$$

and dynamics

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + d_k. \tag{62}$$

The cost-to-go will take the form

$$J_k(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k + \boldsymbol{v}_k^T \boldsymbol{x}_k + v_k. \tag{63}$$

Repeating our approach from the last subsection, we have

$$\begin{aligned} J_k^*(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k \in \mathbb{R}^m} \{ & \frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k + \boldsymbol{u}_k^T H_k \boldsymbol{x}_k + \boldsymbol{q}_k^T \boldsymbol{x}_k + \boldsymbol{r}_k^T \boldsymbol{u}_k + q_k \\ & + \frac{1}{2}(A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{d}_k)^T V_{k+1} (A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{d}_k) \\ & + \boldsymbol{v}_{k+1}^T (A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{d}_k) + v_{k+1} \}. \end{aligned} \tag{64}$$

Rearranging, we have

$$\begin{aligned} J_k^*(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k \in \mathbb{R}^m} \{ & \frac{1}{2}\boldsymbol{x}_k^T (Q_k + A_k^T V_{k+1} A_k) \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T (R_k + B_k^T V_{k+1} B_k) \boldsymbol{u}_k \\ & + \boldsymbol{u}_k^T (H_k + B_k^T V_{k+1} A_k)^T \boldsymbol{x}_k + (\boldsymbol{q}_k + A_k^T V_{K+1} \boldsymbol{d}_k + A_k^T \boldsymbol{v}_{k+1})^T \boldsymbol{x}_k \\ & + (\boldsymbol{r}_k + B_k^T V_{k+1} \boldsymbol{d}_k + B_k^T \boldsymbol{v}_{k+1}) \boldsymbol{u}_k + (v_{k+1} + \frac{1}{2}\boldsymbol{d}_k^T V_{k+1} \boldsymbol{d}_k + \boldsymbol{v}_{k+1}^T \boldsymbol{d}_k) \}. \end{aligned} \tag{65}$$

Solving this minimization problem, we see that our optimal controller takes the form

$$\boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \boldsymbol{x}_k. \tag{66}$$

We will define the following useful terms which will be used throughout the remainder of this section

$$S_{\boldsymbol{u},k} = \boldsymbol{r}_k + \boldsymbol{v}_{k+1}^T B_k + \boldsymbol{d}_k^T V_{k+1} B_k \tag{67}$$

$$S_{\boldsymbol{uu},k} = R_k + B_k^T V_{k+1} B_k \tag{68}$$

$$S_{\boldsymbol{ux},k} = H_k + B_k^T V_{k+1} A_k. \tag{69}$$

Given this notation, all necessary terms can be computed via the following relations

1. $V_N = Q_N$; $\boldsymbol{v}_N = \boldsymbol{q}_N$; $v_N = q_N$

2.
$$L_k = -S_{\boldsymbol{uu},k}^{-1} S_{\boldsymbol{ux},k} \tag{70}$$
$$\boldsymbol{l}_k = -S_{\boldsymbol{uu},k}^{-1} S_{\boldsymbol{u},k} \tag{71}$$

3.
$$V_k = Q_k + A_k^T V_{k+1} A_k - L_k^T S_{\boldsymbol{uu},k} L_k \tag{72}$$
$$\boldsymbol{v}_k = \boldsymbol{q}_k + A_k^T (\boldsymbol{v}_{k+1} + V_{k+1} \boldsymbol{d}_k) + S_{\boldsymbol{ux},k}^T \boldsymbol{l}_k \tag{73}$$
$$v_k = v_{k+1} + q_k + \boldsymbol{d}_k^T \boldsymbol{v}_{k+1} + \frac{1}{2}\boldsymbol{d}_k^T V_{k+1} \boldsymbol{d}_k + \frac{1}{2}\boldsymbol{l}_k^T S_{\boldsymbol{u},k} \tag{74}$$

4. $\boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \boldsymbol{x}_k$

5. $J_k(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k + \boldsymbol{v}_k^T \boldsymbol{x}_k + v_k$.

Note that in the following subsections (specifically in our discussion of differential dynamic programming) we will introduce more convenient (and compact) notation.

2.3.2 LQR Tracking around a Linear Trajectory

In the previous subsections we have considered the generic linear quadratic control problem, in which we want to regulate to a fixed point, and deviations from this point are penalized. In this section, we will address the case in which we want to track a pre-specified trajectory. Let us assume (for now) that we have been given a nominal trajectory of the form $(\bar{\boldsymbol{x}}_0, \dots, \bar{\boldsymbol{x}}_N)$ and $(\bar{\boldsymbol{u}}_0, \dots, \bar{\boldsymbol{u}}_{N-1})$. We will also assume that this trajectory satisfies our given dynamics, such that

$$\bar{\boldsymbol{x}}_{k+1} = A_k \bar{\boldsymbol{x}}_k + B_k \bar{\boldsymbol{u}}_k + \boldsymbol{d}_k, \ \forall k = 0, \dots, N-1. \tag{75}$$

Then, we can rewrite our dynamics in terms of deviations from the nominal trajectory,

$$\delta \boldsymbol{x}_k = \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k \tag{76}$$

$$\delta \boldsymbol{u}_k = \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k. \tag{77}$$

Rewriting, we have

$$\delta \boldsymbol{x}_{k+1} = A_k \delta \boldsymbol{x}_k + B_k \delta \boldsymbol{u}_k. \tag{78}$$

Thus, tracking the nominal trajectory reduces to driving the state deviation, $\delta \boldsymbol{x}_k$, to zero. Note that solving this problem requires rewriting the original cost function in terms of the deviations $\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k$.

2.3.3 LQR Tracking around a Nonlinear Trajectory

Despite LQR being an extremely powerful approach to optimal control, it suffers from a handful of limitations. First and foremost, it assumes the dynamics are (possibly time-varying) linear, and the cost function is quadratic. While most systems are in fact nonlinear, a typical approach to designing feedback controllers is to linearize around some operating point. This is an effective method for designing regulators, which aim to control the system to some particular state. If, in contrast, we wish to track a trajectory, we must instead linearize around this trajectory. We will assume we are given a nominal trajectory which satisfies the nonlinear dynamics, such that

$$\bar{\boldsymbol{x}}_{k+1} = f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k), \ \forall k = 0, \dots, N-1. \tag{79}$$

Given this, we can linearize our system at each timestep by Taylor expanding,

$$\boldsymbol{x}_{k+1} \approx f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + \underbrace{\frac{\partial f}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)}_{A_k}(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + \underbrace{\frac{\partial f}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)}_{B_k}(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k) \tag{80}$$

which allows us to again rewrite the system in terms of deviations, to get

$$\delta \boldsymbol{x}_{k+1} = A_k \delta \boldsymbol{x}_k + B_k \delta \boldsymbol{u}_k \tag{81}$$

which is linear in $\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k$. Note that design of systems of this type often require careful design and analysis, as deviating from the nominal trajectory results in the loss of accuracy of the local model linearization.

In designing this tracking system, a second question now occurs: how do we choose our cost function? One possible option is arbitrary choice of Q and R by the system designer. This has the advantage of being easily customizable to change system behavior, and we can guarantee the necessary conditions on these matrices. A second option, if we are given some arbitrary (possibly non-quadratic) cost function c, is to locally quadratize the cost function. Writing

$$c_k := c(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) \tag{82}$$

$$c_{i,k} := \frac{\partial c}{\partial i}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) \tag{83}$$

$$c_{ij,k} := \frac{\partial^2 c}{\partial i \partial j}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) \tag{84}$$

we can second order Taylor expand our cost function around our nominal trajectory

$$c(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \end{bmatrix}^T \begin{bmatrix} 2c_k & c_{\boldsymbol{x},k}^T & c_{\boldsymbol{u},k}^T \\ c_{\boldsymbol{x},k} & c_{\boldsymbol{xx},k} & c_{\boldsymbol{ux},k}^T \\ c_{\boldsymbol{u},k} & c_{\boldsymbol{ux},k} & c_{\boldsymbol{uu},k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \end{bmatrix}. \tag{85}$$

Here $c_{\boldsymbol{xx},k}$ and $c_{\boldsymbol{uu},k}$ replace Q_k and R_k from the previous section, respectively. There are two primary concerns with this approach to choosing the cost function. First, we require the quadratic form in (85) to be positive semi-definite and $c_{\boldsymbol{uu},k}$ to be positive definite, for all k. Second, we have an implicit cost that we would like to stay close to the nominal trajectory to ensure our linearized model does not become inaccurate. As a result of this implicit cost, we may wish to tune the cost terms to yield tracking that is better suited to the nonlinear model that we are tracking.

2.4 Iterative LQR and Differential Dynamic Programming

2.4.1 Iterative LQR

We have addressed the case in which we wish to track a given trajectory with LQR. A natural question, now, is whether we can use LQR to improve on this nominal trajectory? Iterative LQR augments tracking LQR with a forward pass in which the nominal trajectory is updated. As a consequence, it can be used to improve trajectories and in most cases, can be used as a practical trajectory generation and control algorithm for nonlinear systems. We will define the following useful terms

$$Q_k = c_k + v_{k+1} \tag{86}$$

$$Q_{\boldsymbol{x},k} = c_{\boldsymbol{x},k} + f_{\boldsymbol{x},k}^T \boldsymbol{v}_{k+1} \tag{87}$$

$$Q_{\boldsymbol{u},k} = c_{\boldsymbol{u},k} + f_{\boldsymbol{u},k}^T \boldsymbol{v}_{k+1} \tag{88}$$

$$Q_{\boldsymbol{xx},k} = c_{\boldsymbol{xx},k} + f_{\boldsymbol{x},k}^T V_{k+1} f_{\boldsymbol{x},k} \tag{89}$$

$$Q_{\boldsymbol{uu},k} = c_{\boldsymbol{uu},k} + f_{\boldsymbol{u},k}^T V_{k+1} f_{\boldsymbol{u},k} \tag{90}$$

$$Q_{\boldsymbol{ux},k} = c_{\boldsymbol{ux},k} + f_{\boldsymbol{u},k}^T V_{k+1} f_{\boldsymbol{x},k} \tag{91}$$

Algorithm 1 iLQR

```
Require: Nominal control sequence, $(\bar{\boldsymbol{u}}_0, \dots, \bar{\boldsymbol{u}}_{N-1})$
1: $\delta \boldsymbol{u}_k = 0$ for all $k$
2: while not converged do
     Forward pass:
3:       Compute nominal trajectory $\bar{\boldsymbol{x}}_{k+1} = f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k)$ and set $\bar{\boldsymbol{u}}_k \leftarrow \bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k$
     Backward pass:
4:       Compute $Q$ terms around $(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)$ for all $k$ via (86 – 91)
5:       Update feedback law via (93 – 94)
6:       Update value approximation via (95 – 97)
7: end while
8: Compute control law $\pi_k(\boldsymbol{x}_k) = \bar{\boldsymbol{u}}_k + \boldsymbol{l}_k + L_k(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k)$
9: return $\{\pi_k\}_{k=0}^{N-1}$
```

where $f_{\boldsymbol{x},k} = A_k$ and $f_{\boldsymbol{u},k} = B_k$. In this form, the optimal control perturbation is

$$\delta \boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \delta \boldsymbol{x}_k \tag{92}$$

where

$$\boldsymbol{l}_k = -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \tag{93}$$

$$L_k = -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{ux},k}. \tag{94}$$

Finally, the local backward recursion can be completed by updating the value function terms via

$$v_k = Q_k - \frac{1}{2} \boldsymbol{l}_k^T Q_{\boldsymbol{uu},k} \boldsymbol{l}_k \tag{95}$$

$$\boldsymbol{v}_k = Q_{\boldsymbol{x},k} - L_k^T Q_{\boldsymbol{uu},k} \boldsymbol{l}_k \tag{96}$$

$$V_k = Q_{\boldsymbol{xx},k} - L_k^T Q_{\boldsymbol{uu},k} L_k. \tag{97}$$

So far, we have simply derived an alternative method for performing a quadratic approximation of the DP recursion around some nominal trajectory. The iterative LQR (iLQR) algorithm differs by introducing a forward pass that updates the trajectory that is being tracked. The algorithm alternates between forward passes, in which the control policy is applied to the nonlinear dynamics, and backward passes in which the cost function and dynamics are linearized around the new nominal trajectory, and the quadratic approximation of the value, as well as the new control law, is computed. The iterative LQR algorithm is outlined in Algorithm 1. Critically, note that this algorithm returns both a nominal trajectory, in terms of the $\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k$, as well as a feedback policy that stabilizes around this trajectory.

2.4.2 Differential Dynamic Programming

Iterative LQR performs trajectory optimization by first linearizing the dynamics and quadratizing the cost function, and then performing the dynamic programming recursion to compute optimal controls. While this linearization/quadratization approach is sufficient for approximating the Bellman equation such that it may be solved analytically, an alternative approach is to directly approximate the Bellman equation. *Differential dynamic programming* (DDP) directly builds a quadratic approximation of the right hand side of the Bellman equation (as opposed to first approximating the dynamics and the cost function), which may then be solved analytically. We will first define the change in the value of J_k under a perturbation $\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k$,

$$Q(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k) := c(\bar{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k, \bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k) + J_{k+1}(f(\bar{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k, \bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k)). \tag{98}$$

Note that Q here is different from the Q matrix in Section 2.3. Using the same notation as in (82), we can write the quadratic expansion of (98) as

$$Q(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \end{bmatrix}^T \begin{bmatrix} 2Q_k & Q_{\boldsymbol{x},k}^T & Q_{\boldsymbol{u},k}^T \\ Q_{\boldsymbol{x},k} & Q_{\boldsymbol{xx},k} & Q_{\boldsymbol{ux},k}^T \\ Q_{\boldsymbol{u},k} & Q_{\boldsymbol{ux},k} & Q_{\boldsymbol{uu},k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \end{bmatrix} \tag{99}$$

where

$$Q_k = c_k + v_{k+1} \tag{100}$$
$$Q_{\boldsymbol{x},k} = c_{\boldsymbol{x},k} + f_{\boldsymbol{x},k}^T \boldsymbol{v}_{k+1} \tag{101}$$
$$Q_{\boldsymbol{u},k} = c_{\boldsymbol{u},k} + f_{\boldsymbol{u},k}^T \boldsymbol{v}_{k+1} \tag{102}$$
$$Q_{\boldsymbol{xx},k} = c_{\boldsymbol{xx},k} + f_{\boldsymbol{x},k}^T V_{k+1} f_{\boldsymbol{x},k} + \boldsymbol{v}_{k+1} \cdot f_{\boldsymbol{xx},k} \tag{103}$$
$$Q_{\boldsymbol{uu},k} = c_{\boldsymbol{uu},k} + f_{\boldsymbol{u},k}^T V_{k+1} f_{\boldsymbol{u},k} + \boldsymbol{v}_{k+1} \cdot f_{\boldsymbol{uu},k} \tag{104}$$
$$Q_{\boldsymbol{ux},k} = c_{\boldsymbol{ux},k} + f_{\boldsymbol{u},k}^T V_{k+1} f_{\boldsymbol{x},k} + \boldsymbol{v}_{k+1} \cdot f_{\boldsymbol{ux},k}. \tag{105}$$

Note that these terms differ only from iLQR via the last term in (103 – 105), which are second order approximation of the dynamics. Note that the dot notation denotes tensor contraction.

Given this, we can partially minimize this quadratic form over the control deviation,

$$\delta \boldsymbol{u}_k^* = \text{argmin}_{\delta \boldsymbol{u}} Q(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}) = \boldsymbol{l}_k + L_k \delta \boldsymbol{x}_k \tag{106}$$

where

$$\boldsymbol{l}_k = -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{u},k} \tag{107}$$
$$L_k = -Q_{\boldsymbol{uu},k}^{-1} Q_{\boldsymbol{ux},k}. \tag{108}$$

The DDP algorithm is identical to Algorithm 1, just with the alternative definitions for $Q_{\boldsymbol{xx},k}, Q_{\boldsymbol{uu},k}$ and $Q_{\boldsymbol{ux},k}$. The main philosophical difference between iLQR and DDP is that iLQR first approximates the dynamics and cost, and then solves the Bellman equation directly, whereas DDP directly approximates the Bellman equation. While DDP yields a more accurate approximation, computing the second order dynamics terms is expensive in practice. Practically, iLQR is sufficient for most applications.

2.4.3 Algorithmic Details for iLQR and DDP

Algorithm 1 leaves out several details that would be critical for implementing the algorithm. First, what convergence criteria should we use? In [TL05], the authors stop when the update to the nominal control action sequence is sufficiently small. In [LK14], the authors iterate until the cost of the trajectory (with some additional penalty terms) increases. Finally, a variety of convergence criteria are based on expected trajectory improvement, computed via line search [JM70, TET12]. In the forward pass, standard iLQR computes an updated nominal control sequence via $\bar{\boldsymbol{u}}_k \leftarrow \bar{\boldsymbol{u}}_k + \boldsymbol{l_k} + L_k \delta \boldsymbol{x}_k$. Instead we can weight $\boldsymbol{l_k}$ with a scalar $\alpha \in [0,1]$ for which we perform line search. This results in increased stability (as with standard line search for step size determination in nonlinear optimization) and possibly faster convergence. When α is close to zero, or alternative conditions (such as expected improvement being small) are met, we terminate. For a further discussion of this approach, we refer the reader to [TET12], which also features a discussion of step size determination in the DDP literature.

Iterative LQR and DDP rely on minimizing a second order approximation of the cost-to-go perturbation. However, we do not have any guarantees on the convexity of $Q(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k)$ for arbitrary cost functions. Note that DDP is performing a Newton step [LS92] (iLQR is performing a Newton step with an approximation of the Hessian) via decomposing the optimization problem over controls into N smaller optimization problems. As such, standard approaches from Newton methods for regularization have been applied, such as replacing $Q_{\boldsymbol{uu},k}$ with $Q_{\boldsymbol{uu},k} + \mu I$, which is convex for sufficiently large μ. Alternative approaches have been explored in [TET12, TMT14], based on regularizing the quadratic term in the approximate cost-to-go.

Both iLQR and DDP are local methods. Full dynamic programming approaches yield globally optimal feedback policies. In contrast, iLQR and DDP yield nominal trajectories and local stabilizing controllers. However, these local controllers are often sufficient for tracking the trajectory. As they are local method, choice of initial control sequence is important, and poor choice may result in poor convergence. Additionally, we have not considered constraints on either state or action in the derivation of iLQR or DDP. This is currently an active area of research [XLH17, TMT14, GB17].

2.5 Stochastic Optimal Control

We began this chapter by discussing the generic optimal control problem that we consider in this course, and took a small detour to discuss the special case of discrete-time LQR. We will now generalize the previously stated optimal control problem to consider the stochastic case, in which random noise is added to the state transitions and/or observations. In the remainder of this chapter we will discuss the discrete time case.

2.5.1 The Stochastic Optimal Control Problem

We consider systems of the form

$$\boldsymbol{x}_{k+1} = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k),\ \ k = 0, \ldots, N-1 \tag{109}$$

where $\boldsymbol{\omega}_k \sim p(\cdot \mid \boldsymbol{x}_k, \boldsymbol{u}_k)$ is the disturbance or noise. We write the expected cost under policy $\pi = \{\pi_0, \ldots, \pi_{N-1}\}$ as

$$J_\pi(\boldsymbol{x}_0) = \mathbb{E}_{\boldsymbol{\omega}_{0:N-1}} \left[c_N(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} c_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{\omega}_k) \right]. \tag{110}$$

Then, the stochastic control problem we wish to solve is to find

$$J^*(\boldsymbol{x}_0) = \min_\pi J_\pi(\boldsymbol{x}_0). \tag{111}$$

In contrast to the deterministic optimal control problem, we are specifically interested in finding the optimal closed-loop *policy* in the stochastic case. Closed-loop policies can achieve lower cost than open-loop action sequences, as they take advantage of current state information in their action selection. We have also, importantly, assumed the cost is strictly additive. Finally, we have assumed we are operating in a *risk-neutral* setting. This risk neutrality corresponds to the expectation in (110), which could be replaced by any *risk metric*, which maps a distribution over outcomes to a scalar.

2.5.2 The Principle of Optimality: Stochastic Case

We can now state the principle of optimality for the stochastic optimal control problem. Note that this is a strict generalization of the deterministic case.

Theorem 2.2 (Discrete-time Principle of Optimality: Stochastic Case)**.** *Let* $\pi^* = (\pi_0^*, \ldots, \pi_{N-1}^*)$ *be an optimal policy. Assume state* $\boldsymbol{x}_k$ *is reachable. Consider the tail subproblem*

$$\mathbb{E}_{w_{i:N-1}} \left[c_T(\boldsymbol{x}_N) + \sum_{k=i}^{N-1} c_k(\boldsymbol{x}_k, \pi_k(\boldsymbol{x}_k), \boldsymbol{\omega}_k) \right]. \tag{112}$$

Then the truncated policy $(\pi_i^*, \ldots, \pi_{N-1}^*)$ *is optimal for the subproblem.*

The intuition behind the stochastic principle of optimality is effectively the same as for the deterministic, and the proof is also based on decomposition of the total cost into two cost terms. This is possible due to the linearity of expectation. Stated simply, if a better policy existed for the tail problem, this would imply π^* is suboptimal.

The stochastic version of the principle of optimality leads to a concomitant dynamic programming algorithm, which takes the form

$$J_N(\boldsymbol{x}_N) = c_T(\boldsymbol{x}_N) \tag{113}$$

$$J_k(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)} \mathbb{E}_{w_k} \left[c_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k) + J_{k+1}(f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k)) \right] \tag{114}$$

and the optimal policy is

$$\pi_k^*(\boldsymbol{x}_k) = \operatorname{argmin}_{\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)} \mathbb{E}_{w_k} \left[c_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k) + J_{k+1}(f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\omega}_k)) \right]. \tag{115}$$

2.6 Stochastic LQR and LQG

2.6.1 LQR with Additive Noise

We will first address the stochastic LQR problem. The system dynamics are

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k \tag{116}$$

where $\boldsymbol{\omega}_k \sim \mathcal{N}(0, \Sigma_\omega)$, and the stage-wise cost is

$$c_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}(\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k). \tag{117}$$

with terminal cost $\frac{1}{2}\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N$. We wish to minimize the expected cost. The cost-to-go, as in the deterministic case, will be quadratic. Thus, plugging into the Bellman equation, we have

$$\begin{aligned} J_k^*(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k \in \mathbb{R}^m} \mathbb{E}[\frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k \\ &\quad + \frac{1}{2}(A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k)^T V_{k+1} (A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k)] \qquad (118) \\ &= \min_{\boldsymbol{u}_k \in \mathbb{R}^m} \{\frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k \\ &\quad + \mathbb{E}[\frac{1}{2}(A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k)^T V_{k+1} (A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k)]\}. \qquad (119) \end{aligned}$$

Following the same minimization procedure as for LQR, we see that the policy is identical to that in Section 2.3. The Riccati equation, however, is

$$\begin{aligned} V_k &= Q_k + L_k^T R_k L_k + \mathbb{E}[(A_k + B_k L_k + \boldsymbol{\omega}_k)^T V_{k+1} (A_k + B_k L_k + \boldsymbol{\omega}_k)] \qquad (120) \\ &= Q_k + L_k^T R_k L_k + (A_k + B_k L_k)^T V_{k+1} (A_k + B_k L_k) + \operatorname{tr}(\Sigma_\omega V_{k+1}) \qquad (121) \end{aligned}$$

where $\operatorname{tr}(\cdot)$ denotes the trace. The equality between (120) and (121) holds as

$$\mathbb{E}[(A_k + B_k L_k)^T V_{k+1} \boldsymbol{\omega}_k] = 0 \tag{122}$$

for zero-mean $\boldsymbol{\omega}_k$, and $\mathbb{E}[\boldsymbol{\omega}_k^T V_{k+1} \boldsymbol{\omega}_k] = \operatorname{tr}(\Sigma_\omega V_{k+1})$. Note that this is identical to the deterministic case, other than the additive trace term at the end.

2.6.2 Problems with Imperfect State Information

We now consider the case in which direct, perfect state information isn't available. Instead, we have a noise-corrupted measurements

$$\boldsymbol{z}_0 = h_0(\boldsymbol{x}_0, \boldsymbol{\nu}_0) \tag{123}$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{\nu}_k), \ k = 0, \dots, N-1. \tag{124}$$

The observation disturbance is characterized by distribution

$$p(\cdot \mid \boldsymbol{x}_k, \ldots, \boldsymbol{x}_0, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_0, \boldsymbol{\omega}_{k-1}, \ldots, \boldsymbol{\omega}_0, \boldsymbol{\nu}_{k-1}, \ldots, \boldsymbol{\nu}_0) \tag{125}$$

and the initial state $\boldsymbol{x}_0$ is distributed according to $p(\boldsymbol{x}_0)$. We will define the information vector as

$$\boldsymbol{i}_k = [\boldsymbol{z}_0^T, \ldots, \boldsymbol{z}_k^T, \boldsymbol{u}_0^T, \ldots, \boldsymbol{u}_{k-1}^T]^T. \tag{126}$$

Armed with this, we will consider *admissible* policies $\pi(\boldsymbol{i}_k) \in \mathcal{U}_k$, which implies they are *causal* — they do not rely on information only available in the future. The goal of the control problem, then is to minimize

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\omega}_{0:N-1}, \boldsymbol{\nu}_{0:N-1}} \left[c_T(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} c(\boldsymbol{x}_k, \pi(\boldsymbol{i}_k), \boldsymbol{\omega}_k) \right]. \tag{127}$$

Treating this problem as a perfect state information problem and attempting to solve directly results in several problems. In addition to the standard difficulties associated with applying DP to generic problems, $\boldsymbol{i}_k$ has expanding dimension over the length of the problem. Alternatively, we may reason in terms of sufficient statistics: quantities that summarize all of the informational content of $\boldsymbol{i}_k$. For example, if we can construct a conditional distribution over state, $p(\boldsymbol{x}_k \mid \boldsymbol{i}_k)$, we can design a policy of the form $\pi_k(p(\boldsymbol{x}_k \mid \boldsymbol{i}_k))$.

2.6.3 LQG and the Separation Principle

Given the generic imperfect state information control problem, we will address an extremely important special case. We will again consider quadratic cost of the form

$$\frac{1}{2}\mathbb{E}\left[\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N + \sum_{k=0}^{N-1} \boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k \right] \tag{128}$$

subject to dynamics

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k \tag{129}$$

and measurements

$$\boldsymbol{z}_k = C_k \boldsymbol{x}_k + \boldsymbol{\nu}_k. \tag{130}$$

The initial state $\boldsymbol{x}_0$, and process and measurements noise $\boldsymbol{\omega}_{0:N-1}, \boldsymbol{\nu}_{0:N-1}$ independent, zero-mean Gaussians. We will write the covariance of $\boldsymbol{\omega}_k$ and $\boldsymbol{\nu}_k$ as $\Sigma_{\boldsymbol{\omega},k}$ and $\Sigma_{\boldsymbol{\nu},k}$, respectively. We will write $\Sigma_{\boldsymbol{x},0}$ for the covariance of $\boldsymbol{x}_0$.

We will skip the lengthy derivation, but the optimal control policy takes the form

$$\boldsymbol{u}_k^* = L_k \hat{\boldsymbol{x}}_k \tag{131}$$

where $\hat{\boldsymbol{x}}_k$ is the state estimate from the Kalman filter, and L_k is the standard gain from LQR. Note that the Kalman filter maintains a Gaussian estimate of the state, and so the sufficient statistics are the mean and variance. However, the policy depends only on the mean. The policy can be designed as if access to perfect state information is available, while the estimator can be designed without considering the controller. This is known as the *separation principle.*

2.7 Further Reading

A comprehensive coverage of linear quadratic methods for optimal control is Anderson and Moore [AM07]. LQG is covered in discrete time in [Ber12]. The original, comprehensive reference on DDP is [JM70], but a large body of literature on the method has been produced since then. The original papers on iLQR are [TL05, LT04].

3 The HJB and HJI Equations

In this section, we will extend the ideas of dynamic programming to the continuous time setting. Restating the continuous time optimal control problem, we assume dynamics

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{132}$$

and cost

$$J(\boldsymbol{x}(0)) = c_f(\boldsymbol{x}(t_f), t_f) + \int_0^{t_f} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau. \tag{133}$$

where t_f is fixed.

3.1 The Principle of Optimality in Continuous Time

3.1.1 Hamilton-Jacobi-Bellman

As in the discrete time principle of optimality, consider the tail problem

$$J(\boldsymbol{x}(t), \{\boldsymbol{u}(\tau)\}_{\tau=t}^{t_f}, t) = c_f(\boldsymbol{x}(t_f), t_f) + \int_t^{t_f} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau \tag{134}$$

where $t \leq t_f$ and $\boldsymbol{x}(t)$ is an admissible state value. The optimal solution to this tail problem comes from the functional minimization

$$J^*(\boldsymbol{x}(t), t) = \min_{\{\boldsymbol{u}(\tau)\}_{\tau=t}^{t_f}} \left\{ c_f(\boldsymbol{x}(t_f), t_f) + \int_t^{t_f} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau \right\}. \tag{135}$$

Note, then, that due to the additivity of cost we can split the problem up over time,

$$J^*(\boldsymbol{x}(t), t) = \min_{\{\boldsymbol{u}(\tau)\}_{\tau=t}^{t_f}} \left\{ \int_t^{t+\Delta t} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau + c_f(\boldsymbol{x}(t_f), t_f) + \int_{t+\Delta t}^{t_f} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau \right\} \tag{136}$$

which by applying the principle of optimality to the tail cost,

$$J^*(\boldsymbol{x}(t), t) = \min_{\{\boldsymbol{u}(\tau)\}_{\tau=t}^{t+\Delta t}} \left\{ \int_t^{t+\Delta t} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau) d\tau + J^*(\boldsymbol{x}(t+\Delta t), t+\Delta t) \right\}. \tag{137}$$

Let $J_t^*(\boldsymbol{x}(t),t) = \nabla_t J^*(\boldsymbol{x}(t),t)$ and $J_{\boldsymbol{x}}^*(\boldsymbol{x}(t),t) = \nabla_{\boldsymbol{x}} J^*(\boldsymbol{x}(t),t)$. Taylor expanding, we have

$$\begin{aligned} J^*(\boldsymbol{x}(t),t) = \min_{\{\boldsymbol{u}(\tau)\}_{\tau=t}^{t+\Delta t}} \{&c(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\Delta t + J^*(\boldsymbol{x}(t),t) + (J_t^*(\boldsymbol{x}(t),t))\Delta t \\ &+ (J_{\boldsymbol{x}}^*(\boldsymbol{x}(t),t))^T(\boldsymbol{x}(t+\Delta t) - \boldsymbol{x}(t)) + o(\Delta t)\} \end{aligned} \tag{138}$$

for small Δt. The first term is a result of Taylor expanding the integral and applying the fundamental theorem of calculus. Note that we can pull $J^*(\boldsymbol{x}(t),t)$ out of the minimization over cost, as this quantity will not vary under different choices of future actions. Dividing through by Δt and taking the limit $\Delta t \to 0$, we obtain the *Hamilton-Jacobi-Bellman* equation

$$0 = J_t^*(\boldsymbol{x}(t),t) + \min_{\boldsymbol{u}(t)} \left\{c(\boldsymbol{x}(t),\boldsymbol{u}(t),t) + (J_{\boldsymbol{x}}^*(\boldsymbol{x}(t),t))^T f(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\right\} \tag{139}$$

with terminal condition

$$J^*(\boldsymbol{x}(t_f),t_f) = c_f(\boldsymbol{x}(t_f),t_f). \tag{140}$$

For convenience, we will define the Hamiltonian

$$\mathcal{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),J_{\boldsymbol{x}}^*,t) := c(\boldsymbol{x}(t),\boldsymbol{u}(t),t) + (J_{\boldsymbol{x}}^*(\boldsymbol{x}(t),t))^T f(\boldsymbol{x}(t),\boldsymbol{u}(t),t) \tag{141}$$

which allow us to compactly write the HJB equation as

$$0 = J_t^*(\boldsymbol{x}(t),t) + \min_{\boldsymbol{u}(t)} \{\mathcal{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),J_{\boldsymbol{x}}^*,t)\}. \tag{142}$$

The HJB equation is a partial differential equation that, for cost-to-go $J^*(\boldsymbol{x}(t),t)$, will satisfy all time-state pairs $(\boldsymbol{x}(t),t)$. The previous informal derivation assumed differentiability of $J^*(\boldsymbol{x}(t),t)$, which we do not know a priori. This assumption is rectified by the following theorem on solutions to the HJB equation.

Theorem 3.1 (Sufficiency Theorem)**.** *Suppose $V(\boldsymbol{x},t)$ is a solution to the HJB equation, that $V(\boldsymbol{x},t)$ is C^1 in $\boldsymbol{x}$ and t, and that*

$$\begin{aligned} 0 &= V_t(\boldsymbol{x},t) + \min_{\boldsymbol{u}\in\mathcal{U}} \left\{c(\boldsymbol{x},\boldsymbol{u},t) + (V_{\boldsymbol{x}}(\boldsymbol{x},t))^T f(\boldsymbol{x},\boldsymbol{u},t)\right\} \\ V(\boldsymbol{x},t_f) &= c_f(\boldsymbol{x},t_f)\ \forall \boldsymbol{x} \end{aligned}$$

Suppose also that $\pi^*(\boldsymbol{x},t)$ attains the minimum in this equation for all t and $\boldsymbol{x}$. Let $\{\boldsymbol{x}^*(t) \mid t \in [t_0,t_f]\}$ be the state trajectory obtained from the given initial condition $\boldsymbol{x}(0)$ when the control trajectory $\boldsymbol{u}^*(t) = \pi^*(\boldsymbol{x}^*(t),t), t \in [t_0,t_f]$ is used. Then V is equal to the optimal cost-to-go function, i.e.,

$$V(\boldsymbol{x},t) = J^*(\boldsymbol{x},t)\ \forall \boldsymbol{x},t. \tag{143}$$

Furthermore, the control trajectory $\{\boldsymbol{u}^*(t) \mid t \in [t_0,t_f]\}$ is optimal..

Proof. [Ber12], Volume 1, Section 7.2. □

3.1.2 Continuous-Time LQR

As a useful result of the HJB equations, we will derive LQR in continuous time. We aim to minimize

$$J(\boldsymbol{x}(0)) = \frac{1}{2}\boldsymbol{x}^T(t_f)Q_f\boldsymbol{x}(t_f) + \frac{1}{2}\int_0^{t_f} \boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) + \boldsymbol{u}^T(t)R(t)\boldsymbol{u}(t)dt \tag{144}$$

subject to dynamics

$$\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t). \tag{145}$$

As in discrete LQR, we will assume $Q_f, Q(t)$ are positive semidefinite, and $R(t)$ is positive definite. We will also assume t_f is fixed, and the state and action are unconstrained.

We will write the Hamiltonian,

$$\mathcal{H} = \frac{1}{2}\boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{u}^T(t)R(t)\boldsymbol{u}(t) + J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^T(A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t)) \tag{146}$$

which yields necessary optimality conditions

$$0 = \nabla_{\boldsymbol{u}}\mathcal{H} = R(t)\boldsymbol{u}(t) + B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t). \tag{147}$$

Since $\nabla^2_{\boldsymbol{uu}}\mathcal{H} = R(t) > 0$, the control that satisfies the necessary conditions is the global minimizer. Rearranging, we have

$$\boldsymbol{u}^*(t) = -R^{-1}(t)B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t) \tag{148}$$

which we can plug back into the Hamiltonian to yield

$$\begin{aligned}\mathcal{H} &= \frac{1}{2}\boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) + \frac{1}{2}J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TB(t)R^{-1}(t)B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t) \qquad (149)\\ &\quad + J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TA(t)\boldsymbol{x}(t) - J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TB(t)R^{-1}(t)B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)\\ &= \frac{1}{2}\boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) - \frac{1}{2}J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TB(t)R^{-1}(t)B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t) + J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TA(t)\boldsymbol{x}(t). \qquad (150)\end{aligned}$$

This gives the HJB equation

$$\begin{aligned}0 &= J^*_t(\boldsymbol{x}(t),t) + \frac{1}{2}\boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) - \frac{1}{2}J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TB(t)R^{-1}(t)B^T(t)J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t) \qquad (151)\\ &\quad + J^*_{\boldsymbol{x}}(\boldsymbol{x}(t),t)^TA(t)\boldsymbol{x}(t)\end{aligned}$$

with boundary condition

$$J^*(\boldsymbol{x}(t_f),t_f) = \frac{1}{2}\boldsymbol{x}^T(t_f)Q_f\boldsymbol{x}(t_f). \tag{152}$$

It may appear as if we are stuck here, as this form of the HJB doesn't immediately yield $J^*(\boldsymbol{x}(t),t)$. Armed with the knowledge that the discrete time LQR problem has a quadratic cost-to-go, we will cross our fingers and guess a solution of the form

$$J^*(\boldsymbol{x}(t),t) = \frac{1}{2}\boldsymbol{x}^T(t)V(t)\boldsymbol{x}(t). \tag{153}$$

Substituting, we have

$$
\begin{aligned}
0 = &\frac{1}{2}\boldsymbol{x}^T(t)\dot{V}(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{x}^T(t)Q(t)\boldsymbol{x}(t) \\
&- \frac{1}{2}\boldsymbol{x}^T(t)V(t)B(t)R^{-1}(t)B^T(t)V(t)\boldsymbol{x}(t) + \boldsymbol{x}^T(t)V(t)A(t)\boldsymbol{x}(t)
\end{aligned} \tag{154}
$$

Note that we will decompose

$$
\boldsymbol{x}^T(t)V(t)A(t)\boldsymbol{x}(t) = \frac{1}{2}\boldsymbol{x}^T(t)V(t)A(t)\boldsymbol{x}(t) + \frac{1}{2}\boldsymbol{x}^T(t)A^T(t)V(t)\boldsymbol{x}(t) \tag{155}
$$

which yields

$$
0 = \frac{1}{2}\boldsymbol{x}^T(t)\left(\dot{V}(t) + Q(t) - V(t)B(t)R^{-1}(t)B^T(t)V(t) + V(t)A(t) + A^T(t)V(t)\right)\boldsymbol{x}(t). \tag{156}
$$

This equation must hold for all $\boldsymbol{x}(t)$, so

$$
-\dot{V}(t) = Q(t) - V(t)B(t)R^{-1}(t)B^T(t)V(t) + V(t)A(t) + A^T(t)V(t) \tag{157}
$$

with boundary condition $V(t_f) = Q_f$.

Therefore, the HJB PDE has been reduced to a set of matrix ordinary differential equations (the Riccati equation). This is integrated backwards in time to find the full control policy as a function of time. One we have found $V(t)$, the control policy is

$$
\boldsymbol{u}^*(t) = -R^{-1}(t)B^T(t)V(t)\boldsymbol{x}(t). \tag{158}
$$

Similarly to the discrete case, the feedback gains tend toward constant in the limit of the infinite horizon problem, under some technical assumptions.

3.2 Differential Games

We have so far addressed the case in which we aim to solve the optimal control problem for a single agent. We will now consider an adversarial game setting, in which there exists another player that aims to maximally harm the first agent. In particular, we will consider zero sum games in which the second agent aims to maximize the cost of the first agent. While the differential game setting is not restricted to this case — agents may have separate cost functions that partially interfere or aid each other — the zero-sum case lends itself to useful analytical tools.

3.2.1 Differential Games and Information Patterns

We consider the two player differential game with dynamics

$$
\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \tag{159}
$$

where the first player takes action $\boldsymbol{u}(t)$ at time t, and the second player takes action $\boldsymbol{d}(t)$. The state $\boldsymbol{x}(t)$ is the joint state of both players. We write the cost as

$$J(\boldsymbol{x}(t)) = c_f(\boldsymbol{x}(0)) + \int_t^0 c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{d}(\tau))d\tau \tag{160}$$

which the first agent aims to maximize, and the second agent aims to minimize.

To fully specify the differential game, we must specify what each agent knows, and when. This is referred to as the *information pattern* of the game. In addition to capturing the knowledge of the state available to each agent, the information pattern also captures the knowledge of each other agents' strategies available to each agent.

3.2.2 Hamilton-Jacobi-Isaacs

The key idea in building the multi-agent equivalent of the HJB equation will again be to apply the principle of optimality. We consider the information pattern in which the adversary has access to the instantaneous control action of the first agent, so the cost takes the form

$$J(\boldsymbol{x}(t), t) = \min_{\Gamma(\boldsymbol{u})(\cdot)} \max_{\boldsymbol{u}(\cdot)} \left\{ \int_t^0 c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{d}(\tau))d\tau + c_f(\boldsymbol{x}(0)) \right\}. \tag{161}$$

Applying the dynamic programming principle, we have

$$J(\boldsymbol{x}(t), t) = \min_{\Gamma(\boldsymbol{u})(\cdot)} \max_{\boldsymbol{u}(\cdot)} \left\{ \int_t^{t+\Delta t} c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{d}(\tau))d\tau + J(\boldsymbol{x}(t+\Delta t), t+\Delta t) \right\}. \tag{162}$$

We can take the same strategy as with the informal derivation of the HJB equation, and Taylor expand both terms to yield

$$\begin{aligned} J(\boldsymbol{x}(t), t) = \min_{\Gamma(\boldsymbol{u})(\cdot)} \max_{\boldsymbol{u}(\cdot)} \{& c(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{d}(\tau))\Delta t + J(\boldsymbol{x}(t), t) \\ &+ (J_{\boldsymbol{x}}(\boldsymbol{x}(t), t))^T f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t))\Delta t + J_t(\boldsymbol{x}(t), t)\Delta t\}. \end{aligned} \tag{163}$$

Note that we are optimizing over instantaneous actions, and so we optimizing over finite dimensional quantities as opposed to functions. Dividing through by Δt and removing redundant terms, we get the *Hamilton-Jacobi-Isaacs* (HJI) equation

$$0 = J_t(\boldsymbol{x}, t) + \max_{\boldsymbol{u}} \min_{\boldsymbol{d}} \left\{ c(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) + (J_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}))^T f(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{u}) \right\} \tag{164}$$

with boundary condition

$$J(\boldsymbol{x}, 0) = c_f(\boldsymbol{x}). \tag{165}$$

Note that we have switched the order of the min/max.

3.2.3 Reachability

Differential games have applications in multi-agent modeling (both in the context of autonomous systems engineering and, e.g., economics and operations research). One concrete application in engineering is reachability analysis. In this setting, an agent aims to compute the set of states in which there exists a policy that either avoids a target set or enters a target set, subject to adversarial disturbances. The former case, in which we would like to avoid a target set, is useful for safety verification. If we are able to, even in the worst case, guarantee e.g. collision avoidance, we have guarantees on safety (subject of course to our system assumptions). The latter case is useful for task satisfaction. For example, we would like a quadrotor to reach a set of safe hovering poses, even under adversarial disturbances. Finding the backward reachable set in this case would find all states such that there exists a policy that succeeds in reaching the target set.

More concretely, the first case aims to find a set

$$\mathcal{A}(t) = \{\bar{\boldsymbol{x}} : \exists \Gamma(\boldsymbol{u})(\cdot), \forall \boldsymbol{u}(\cdot), \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}), \boldsymbol{x}(t) = \bar{\boldsymbol{x}}, \boldsymbol{x}(0) \in \mathcal{T}\} \tag{166}$$

where $\mathcal{T}$ is the unsafe set which we aim to avoid. Breaking this down, $\mathcal{A}(t)$ is the set of states at time t such that there exists $\Gamma(\boldsymbol{u})$ that maps action $\boldsymbol{u}$ to a disturbance such that, following the dynamics induced by the disturbance and the action sequence, the state is in $\mathcal{T}$ at time 0 (note that we are considering $t \leq 0$).

The second case aims to find a set

$$\mathcal{R}(t) = \{\bar{\boldsymbol{x}} : \forall \Gamma(\boldsymbol{u})(\cdot), \exists \boldsymbol{u}(\cdot), \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}), \boldsymbol{x}(t) = \bar{\boldsymbol{x}}, \boldsymbol{x}(0) \in \mathcal{T}\}, \tag{167}$$

where in this case $\mathcal{T}$ is the set that we wish to reach. In this setting, we wish to find all states that, no matter what strategy the disturbance takes, there exist control actions that can steer the system to the goal state. Because the disturbance is adversarial (we reason over all adversary strategies), this is an extremely conservative form of safety analysis.

Computation of the backward reachable set results from solving a differential *game of kind* in which the outcome is Boolean (i.e. whether or not $\boldsymbol{x}(0) \in \mathcal{T}$). This boolean outcome can be encoded by removing the running cost and choosing a particular form for the final cost. In particular, we can choose a final cost where

$$\boldsymbol{x} \in \mathcal{T} \iff c_f(\boldsymbol{x}) \leq 0. \tag{168}$$

As a result, the agent should aim to maximize c_f to avoid $\mathcal{T}$, whereas the disturbance should aim to minimize it. The two settings then take the following forms:

- Set avoidance: $J(\boldsymbol{x}, t) = \min_{\Gamma(\boldsymbol{u})} \max_{\boldsymbol{u}} c_f(\boldsymbol{x}(0))$
- Set reaching: $J(\boldsymbol{x}, t) = \max_{\Gamma(\boldsymbol{u})} \min_{\boldsymbol{u}} c_f(\boldsymbol{x}(0))$

Sets vs. Tubes. We have so far considered avoidance or reachability problems for which we care about set membership at time $t = 0$. However, for something like collision avoidance, we would like to stay collision free at every time as opposed to a particular time. *Backward reachable sets* capture the case in which only the final time set membership matters, and states for times $t < 0$ do not matter. *Backward reachable tubes* capture the entire time duration of the problem. Any state that passes through the target at any time in the problem duration is included. This yields a modified value function of the form

$$J(\boldsymbol{x}, t) = \min_{\Gamma(\boldsymbol{u})} \max_{\boldsymbol{u}} \min_{\tau \in [t,0]} c_f(\boldsymbol{x}(\tau)). \tag{169}$$

If the target set membership holds at any time τ', then $\min_{\tau \in [t,0]} c_f(\boldsymbol{x}(\tau)) \leq c_f(\boldsymbol{x}(\tau')) \leq 0$.

3.3 Further Reading

Our coverage of reachability analysis is based on the [MBT05], which is an important early work in the field, in addition to being a relatively comprehensive coverage of the method. For a review of differential games with a (slight) emphasis on economics and management science, we refer the reader to [Bre10]. For a review of HJB and continuous time LQR, we refer the reader to [Ber12] and [Kir12].

4 Indirect Methods

4.1 Calculus of Variations

We will begin by restating the optimal control problem. We will to find an admissible control sequence $\boldsymbol{u}^*$ which causes the system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{170}$$

to follow an *admissible* trajectory $\boldsymbol{x}^*$ that minimizes the functional

$$J = c_f(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt. \tag{171}$$

To find the minima of functions of a finite number of real numbers, we rely on the first order optimality conditions to find candidate minima, and use higher order derivatives to determine whether a point is a local minimum. Because we are minimizing a function that maps from some n dimensional space to a scalar, candidate points have zero gradient in each of these dimensions. However, in the optimal control problem, we have a cost *functional*, which maps functions to scalars. This is immediately problematic for our first order conditions — we are required to check the necessary condition at infinite points. The necessary notion of optimality conditions for functionals is provided by calculus of variations.

Concretely, we define a functional J as a rule of correspondence assigning each function $\boldsymbol{x}$ in a class Ω (the domain) to a unique real number. The functional J is linear if and only if

$$J(\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2) = \alpha_1 J(\boldsymbol{x}_1) + \alpha_2 J(\boldsymbol{x}_2) \tag{172}$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2, \alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2$ in Ω. We must now define a notion of "closeness" for functions. Intuitively, two points being close together has an immediate geometric interpretation. We first define the norm of a function. The norm of a function is a rule of correspondence that assigns each $\boldsymbol{x} \in \Omega$, defined over $t \in [t_0, t_f]$, a real number. The norm of $\boldsymbol{x}$, which we denote $\|\boldsymbol{x}\|$, satisfies:

1. $\|\boldsymbol{x}\| \geq 0$, and $\|\boldsymbol{x}\| = 0$ iff $\boldsymbol{x}(t) = 0$ for all $t \in [t_0, t_f]$
2. $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\|$ for all real numbers α
3. $\|\boldsymbol{x}_1 + \boldsymbol{x}_2\| \leq \|\boldsymbol{x}_1\| + \|\boldsymbol{x}_2\|$.

To compare the closeness of two functions $\boldsymbol{y}, \boldsymbol{z}$, we let $\boldsymbol{x}(t) = \boldsymbol{y}(t) - \boldsymbol{z}(t)$. Thus, for two identical functions, $\|\boldsymbol{x}\|$ is zero. Generally, a norm will be small for "close" functions, and large for "far apart" functions. However, there exist many possible definitions of norms that satisfy the above conditions.

4.1.1 Extrema for Functionals

A functional J with domain Ω has a local minimum at $\boldsymbol{x}^* \in \Omega$ if there exists an $\epsilon > 0$ such that $J(\boldsymbol{x}) \geq J(\boldsymbol{x}^*)$ for all $\boldsymbol{x} \in \Omega$ such that $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \epsilon$. Maxima are defined similarly, just with $J(\boldsymbol{x}) \leq J(\boldsymbol{x}^*)$.

Analogously to optimization of functions, we define the variation of the functional as

$$\Delta J(\boldsymbol{x}, \delta \boldsymbol{x}) := J(\boldsymbol{x} + \delta \boldsymbol{x}) - J(\boldsymbol{x}) \tag{173}$$

where $\delta \boldsymbol{x}(t)$ is the *variation* of $\boldsymbol{x}(t)$. The increment of a functional can be written as

$$\Delta J(\boldsymbol{x}, \delta \boldsymbol{x}) = \delta J(\boldsymbol{x}, \delta \boldsymbol{x}) + g(\boldsymbol{x}, \delta \boldsymbol{x}) \|\delta \boldsymbol{x}\| \tag{174}$$

where δJ is linear in $\delta \boldsymbol{x}$. If

$$\lim_{\|\delta \boldsymbol{x}\| \to 0} \{g(\boldsymbol{x}, \delta \boldsymbol{x})\} = 0 \tag{175}$$

then J is said to be differentiable on $\boldsymbol{x}$ and δJ is the variation of J at $\boldsymbol{x}$. We can now state the *fundamental theorem of the calculus of variations.*

Theorem 4.1 (Fundamental Theorem of CoV)**.** *Let $\boldsymbol{x}(t)$ be a vector function of t in the class Ω, and $J(\boldsymbol{x})$ be a differentiable functional of $\boldsymbol{x}$. Assume that the functions in Ω are not constrained by any boundaries. If $\boldsymbol{x}^*$ is an extremal, the variation of J must vanish at $\boldsymbol{x}^*$, that is $\delta J(\boldsymbol{x}^*, \delta \boldsymbol{x}) = 0$ for all admissible $\delta \boldsymbol{x}$ (i.e. such that $\boldsymbol{x} + \delta \boldsymbol{x} \in \Omega$).*

Proof. [Kir12], Section 4.1. □

We will now look at how calculus of variations may be leveraged to approach practical problems. Let x be a scalar continuous function in C^1. We would like to find a function x^* for which the functional

$$J(s) = \int_{t_0}^{t_f} g(x(t), x(t), t)dt \tag{176}$$

has a relative extremum. We will assume $g \in C^2$, that t_0, t_f are fixed, and x_0, x_f are fixed. Let $\boldsymbol{x}$ be any curve in Ω, and we will write the variation δJ from the increment

$$\begin{aligned}\Delta J(x, \delta x) &= J(x + \delta x) - J(x) &(177)\\ &= \int_{t_0}^{t_f} g(x + \delta x, \dot{x} + \delta\dot{x}, t)dt - \int_{t_0}^{t_f} g(x, \dot{x}, t)dt &(178)\\ &= \int_{t_0}^{t_f} g(x + \delta x, \dot{x} + \delta\dot{x}, t) - g(x, \dot{x}, t)dt. &(179)\end{aligned}$$

Expanding via Taylor series, we get

$$\Delta J(x, \delta x) = \int_{t_0}^{t_f} g(x, \dot{x}, t) + \underbrace{\frac{\partial g}{\partial x}}_{g_x}(x, \dot{x}, t)\delta x + \underbrace{\frac{\partial g}{\partial \dot{x}}}_{g_{\dot{x}}}(x, \dot{x}, t)\delta\dot{x} + o(\delta x, \delta\dot{x}) - g(x, \dot{x}, t)dt \tag{180}$$

which yields the variation

$$\delta J = \int_{t_0}^{t_f} g_x(x, \dot{x}, t)\delta x + g_{\dot{x}}(x, \dot{x}, t)\delta\dot{x}\ dt. \tag{181}$$

Integrating by parts, we have

$$\delta J = \int_{t_0}^{t_f} \left[g_x(x, \dot{x}, t) - \frac{d}{dt}g_{\dot{x}}(x, \dot{x}, t)\right] \delta x \delta t + [g_{\dot{x}}(x, \dot{x}, t)\delta x(t)]_{t_0}^{t_f}. \tag{182}$$

We have assumed $x(t_0), x(t_f)$ given, and thus $\delta x(t_0) = 0$, $\delta x(t_f) = 0$. Considering an extremal curve, applying the CoV theorem yields

$$\int_{t_0}^{t_f} \left[g_x(x, \dot{x}, t) - \frac{d}{dt}g_{\dot{x}}(x, \dot{x}, t)\right] \delta x \delta t. \tag{183}$$

We can now state the fundamental lemma of CoV. We will state it for vector functions, although our derivation was for the scalar case.

Lemma 4.2 (Fundamental Lemma of CoV)**.** *If a function h is continuous and*

$$\int_{t_0}^{t_f} h(t)\delta\boldsymbol{x}(t)dt = 0 \tag{184}$$

for every function $\delta\boldsymbol{x}$ that is continuous in the interval $[t_0, t_f]$, then h must be zero everywhere in the interval $[t_0, t_f]$.

Proof. [Kir12], Section 4.2. □

Applying the fundamental lemma, we find that a necessary condition for $\boldsymbol{x}^*$ being an extremal is

$$g_{\boldsymbol{x}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{d}{dt} g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = 0 \tag{185}$$

for all $t \in [t_0, t_f]$, which is the *Euler equation.* This is a nonlinear, time-varying second-order ordinary differential equation with split boundary conditions (at $\boldsymbol{x}(t_0)$ and $\boldsymbol{x}(t_f)$).

4.1.2 Generalized Boundary Conditions

In the previous subsection, we assumed that $t_0, t_f, \boldsymbol{x}(t_0), \boldsymbol{x}(t_f)$ were all given. We will now relax that assumption. In particular, t_f may be fixed or free, and each component of $\boldsymbol{x}(t_f)$ may be fixed or free.

We begin by writing the variation around $\boldsymbol{x}^*$

$$\begin{aligned}\delta J = &\left[g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f)\right]^T \delta\boldsymbol{x}(t_f) + \left[g(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f)\right]^T \delta t_f \\ &+ \int_{t_0}^{t_f} \left[g_{\boldsymbol{x}}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, t) - \frac{d}{dt} g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, t)\right]^T \delta\boldsymbol{x}\delta t\end{aligned} \tag{186}$$

by using the same integration by parts approach as before. Note that for fixed t_f and $\boldsymbol{x}(t_f)$, the variations δt_f and $\delta\boldsymbol{x}(t_f)$ vanish, and so we are left with (183). Because δt_f and $\delta\boldsymbol{x}(t_f)$ do not vanish in this case, we are left with additional boundary conditions that must be satisfied. Note that

$$\delta\boldsymbol{x}_f = \delta\boldsymbol{x}(t_f) + \dot{\boldsymbol{x}}^*(t_f)\delta t_f \tag{187}$$

and substituting this, we have

$$\begin{aligned}\delta J = &\left[g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f)\right]^T \delta\boldsymbol{x}_f + \left[g(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f) - g_{\dot{\boldsymbol{x}}}^T(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f)\dot{\boldsymbol{x}}^*(t_f)\right] \delta t_f \\ &+ \int_{t_0}^{t_f} \left[g_{\boldsymbol{x}}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, t) - \frac{d}{dt} g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, t)\right] \delta\boldsymbol{x}\delta t.\end{aligned} \tag{188}$$

Stationarity of this variation thus requires

$$g_{\dot{\boldsymbol{x}}}(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f) = 0 \tag{189}$$

if $\boldsymbol{x}_f$ is free, and

$$g(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f) - g_{\dot{\boldsymbol{x}}}^T(\boldsymbol{x}^*(t_f), \dot{\boldsymbol{x}}^*(t_f), t_f)\dot{\boldsymbol{x}}^*(t_f) = 0 \tag{190}$$

if t_f is free, in addition to the Euler equation being satisfied. For a complete reference on the boundary conditions associated with a variety of problem specifications, we refer the reader to Section 4.3 of [Kir12].

4.1.3 Constrained Extrema

Previously, we have not considered constraints in the variational problem. However, constraints (and in particular, dynamics constraints) are central to most optimal control problems. Let $\boldsymbol{w} \in \mathbb{R}^{n+m}$ be a vector function in C^1. As previously, we would like to find a function $\boldsymbol{w}^*$ for which the functional

$$J(\boldsymbol{w}) = \int_{t_0}^{t_f} g(\boldsymbol{w}(t), \dot{\boldsymbol{w}}(t), t) dt \tag{191}$$

has a relative extremum, although we additionally introduce the constraints

$$f_i(\boldsymbol{w}(t), \dot{\boldsymbol{w}}(t), t) = 0, \quad i = 1, \dots, n. \tag{192}$$

We will again assume $g \in C^2$ and that $t_0, \boldsymbol{w}(t_0)$ are fixed. Note that as a result of these n constraints, only m of the $n+m$ components of $\boldsymbol{w}$ are independent.

One approach to solving this constrained problem is re-writing the n dependent components of $\boldsymbol{w}$ in terms of the m independent components. However, the nonlinearity of the constraints typically makes this infeasible. Instead, we will turn to Lagrange multipliers. We will write our *augmented functional* as

$$\hat{g}(\boldsymbol{w}(t), \dot{\boldsymbol{w}}(t), \boldsymbol{p}(t), t) := g(\boldsymbol{w}(t), \dot{\boldsymbol{w}}(t), t) + \boldsymbol{p}^T(t) \boldsymbol{f}(\boldsymbol{w}(t), \dot{\boldsymbol{w}}(t), t) \tag{193}$$

where $\boldsymbol{p}(t)$ are Lagrange multipliers that are functions of time. Based on this, a necessary condition for optimality is

$$\hat{g}_{\boldsymbol{w}}(\boldsymbol{w}^*(t), \dot{\boldsymbol{w}}^*(t), \boldsymbol{p}^*(t), t) - \frac{d}{dt} \hat{g}_{\dot{\boldsymbol{w}}}(\boldsymbol{w}^*(t), \dot{\boldsymbol{w}}^*(t), \boldsymbol{p}^*(t), t) = 0 \tag{194}$$

with

$$\boldsymbol{f}(\boldsymbol{w}^*(t), \dot{\boldsymbol{w}}^*(t), t) = 0. \tag{195}$$

4.2 Indirect Methods for Optimal Control

Having built the foundations of functional optimization via calculus of variations, we will now derive the necessary conditions for optimal control under the assumption that the admissible controls are not bounded. The problem, as previously stated, is to find an *admissible control* $\boldsymbol{u}^*$ which causes the system

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{196}$$

to follow an *admissible trajectory* $\boldsymbol{x}^*$ that minimizes the functional

$$J(\boldsymbol{u}) = c_f(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \tag{197}$$

under the assumptions that $c_f \in C^2$, the state and control are unconstrained, and $t_0, \boldsymbol{x}(t_0)$ are fixed. We define the *Hamiltonian* as

$$\mathcal{H}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}(t), t) := c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{p}^T(t) f(\boldsymbol{x}(t), \boldsymbol{u}(t), t). \tag{198}$$

Then, the necessary conditions are

$$\dot{\boldsymbol{x}}^*(t) = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \tag{199}$$

$$\dot{\boldsymbol{p}}^*(t) = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \tag{200}$$

$$0 = \frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \tag{201}$$

which must hold for all $t \in [t_0, t_f]$. Additionally, the boundary conditions

$$\begin{aligned}&[\frac{\partial c_f}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t_f), t_f) - \boldsymbol{p}^*(t_f)]^T \delta \boldsymbol{x}_f \\ &\qquad + [\mathcal{H}(\boldsymbol{x}^*(t_f), \boldsymbol{u}^*(t_f), \boldsymbol{p}^*(t_f), t_f) + \frac{\partial c_f}{\partial t}(\boldsymbol{x}^*(t_f), t_f)]\delta t_f = 0\end{aligned} \tag{202}$$

must be satisfied. Note that as in the previous section, they are automatically satisfied if the terminal state and time are fixed. Based on these necessary conditions, we have a set of $2n$ *first-order* differential equations (for the state and co-state), and a set of m algebraic equations (control equations). The solution to the state and co-state equations will contain $2n$ constants of integration. To solve for these constants, we use the initial conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ (of which there are n), and an additional n (or $n+1$) equations from the boundary conditions. We are left with a two-point boundary value problem, which are considerably more difficult to solve than initial value problems which can just be integrated forward. For a full review of boundary conditions, we again refer the reader to [Kir12].

4.2.1 Proof of the Necessary Conditions

We will now prove the necessary conditions, (199 – 201), along with the boundary conditions (211). For simplicity, assume that the terminal cost is zero, and that $t_f, \boldsymbol{x}(t_f)$ are fixed and given. Consider the augmented cost function

$$\hat{c}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{u}(t), \boldsymbol{p}(t), t) := c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{p}^T(t)[f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) - \dot{\boldsymbol{x}}(t)]. \tag{203}$$

When the constraint holds, this augmented cost function is exactly equal to the original cost function. The augmented total cost is then

$$\hat{J}(\boldsymbol{u}) = \int_{t_0}^{t_f} \hat{c}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{u}(t), \boldsymbol{p}(t), t)dt. \tag{204}$$

Applying the fundamental theorem of CoV on an extremal, we have

$$0 = \delta\hat{J}(\boldsymbol{u}) = \int_{t_0}^{t_f} \left[\overbrace{\frac{\partial \hat{c}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t)}^{\frac{\partial c}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),t)+\frac{\partial f}{\partial \boldsymbol{x}}^T(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),t)\boldsymbol{p}^*(t)} - \overbrace{\frac{d}{dt}\frac{\partial \hat{c}}{\partial \dot{\boldsymbol{x}}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t)}^{-\frac{d}{dt}(-\boldsymbol{p}^*(t))} \right]^T \delta\boldsymbol{x}(t) \tag{205}$$

$$+ \left[\frac{\partial \hat{c}}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t)\right]^T \delta\boldsymbol{u}(t) + \underbrace{\left[\frac{\partial \hat{c}}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t), \dot{\boldsymbol{x}}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t)\right]^T}_{f(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),t)-\dot{\boldsymbol{x}}^*(t)} \delta\boldsymbol{p}(t)dt.$$

Considering each term in sequence, we have:

- $f(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t) - \dot{\boldsymbol{x}}^*(t) = 0$ on an extremal.
- The Lagrange multipliers are arbitrary, so we can select them to make the coefficients of $\delta\boldsymbol{x}(t)$ equal to zero, giving $\dot{\boldsymbol{p}}(t) = -\frac{\partial c}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t) - \frac{\partial f}{\partial \boldsymbol{x}}^T(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t)\boldsymbol{p}^*(t)$.
- The remaining variation $\delta\boldsymbol{u}(t)$ is independent, so its coefficient must be zero, thus $\frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t) + \frac{\partial f}{\partial \boldsymbol{u}}^T(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), t)\boldsymbol{p}^*(t) = 0$.

These conditions exactly give the necessary conditions as previously stated, when recast with the Hamiltonian formalism.

4.3 Pontryagin's Minimum Principle

So far, we have assumed that the admissible controls and states are unconstrained. This assumption is frequently violated for real systems—physical actuators have limits on their realizable outputs, and state constraints may occur due to safety considerations. The control $\boldsymbol{u}^*$ causes the functional J to have a relative minimum if

$$J(\boldsymbol{u}) - J(\boldsymbol{u}^*) = \Delta J \geq 0 \tag{206}$$

for all admissible controls "close" to $\boldsymbol{u}^*$. Letting $\boldsymbol{u} = \boldsymbol{u}^* + \delta\boldsymbol{u}$, the increment can be expressed as

$$\Delta J(\boldsymbol{u}^*, \delta\boldsymbol{u}) = \delta J(\boldsymbol{u}^*, \delta\boldsymbol{u}) + \text{higher order terms.} \tag{207}$$

The variation $\delta\boldsymbol{u}$ is arbitrary only if the extremal control is strictly within the boundary for all time in the interval $[t_0, t_f]$. In general, however, an extremal control lies on a boundary during at least subinterval in the interval $[t_0, t_f]$. As a consequence, admissible control variations $\delta\boldsymbol{u}$ exist whose negatives are not admissible. This implies that a necessary condition for $\boldsymbol{u}^*$ to minimize J is $\delta J(\boldsymbol{u}^*, \delta\boldsymbol{u}) \geq 0$ for all admissible variations with $\|\delta\boldsymbol{u}\|$ small enough. The reason why the equality in the fundamental theorem of CoV (in which we explicitly assumed no constraints) is replaced with an inequality is the presence of the control constraints. This result has an analogue in calculus, where the necessary condition for a scalar function f to have a relative minimum at the end point is that the differential $df \geq 0$.

Assuming bounded controls $\boldsymbol{u} \in \mathcal{U}$, the necessary optimality conditions are

$$\dot{\boldsymbol{x}}^*(t) = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \tag{208}$$

$$\dot{\boldsymbol{p}}^*(t) = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \tag{209}$$

$$\mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t) \leq \mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{p}^*(t), t) \ \forall \boldsymbol{u} \in \mathcal{U} \tag{210}$$

along with the boundary conditions

$$[\frac{\partial c_f}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t_f), t_f) - \boldsymbol{p}^*(t_f)]^T \delta \boldsymbol{x}_f + [\mathcal{H}(\boldsymbol{x}^*(t_f), \boldsymbol{u}^*(t_f), \boldsymbol{p}^*(t_f), t_f) + \frac{\partial c_f}{\partial t}(\boldsymbol{x}^*(t_f), t_f)]\delta t_f = 0. \tag{211}$$

The control $\boldsymbol{u}^*(t)$ causes $\mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t), t)$ to assume its global minimum. This is a harder condition, in general, to analyze. Finally, we have additional necessary conditions. If the final time is fixed and the Hamiltonian does not explicitly depend on time,

$$\mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t)) = c \ \forall t \in [t_0, t_f] \tag{212}$$

and if the final time is free and the Hamiltonian does not depend explicitly on time,

$$\mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*(t)) = 0 \ \forall t \in [t_0, t_f]. \tag{213}$$

Note that in general, uniqueness and existence are not guaranteed in the constrained setting.

4.4 Numerical Aspects of Indirect Optimal Control

4.5 Further Reading

For a practical treatment of indirect methods, we refer the reader to [BH75]. For a more theoretical treatment, we refer the reader to [LM67].

5 Direct Methods for Optimal Control

In the previous section we considered indirect methods to optimal control, in which the necessary conditions for optimality were first applied, yielding a two-point boundary value problem that was solved numerically. We will now consider the class of direct methods, in which the optimal control problem is first discretized, and then the resulting discrete optimization problem is solved numerically.

5.1 Direct Methods

We will write our original continuous optimal control problem,

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \int_0^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t), t \in [0, t_f] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ & \boldsymbol{x}(t_f) \in \mathcal{M}_f \\ & \boldsymbol{u}(t) \in \mathcal{U}, t \in [0, t_f] \end{aligned} \tag{214}$$

where $\mathcal{M}_f = \{\boldsymbol{x} \in \mathbb{R}^n : F(\boldsymbol{x}) = 0\}$ and where we have, for simplicity, assumed zero terminal cost and $t_0 = 0$. We will use forward Euler discretization of the dynamics. We select a discretization $0 = t_0 < t_1 < \ldots < t_N = t_f$ for the interval $[0, t_f]$, and we will write $\boldsymbol{x}_{i+1} \approx \boldsymbol{x}(t), \boldsymbol{u}_i \approx \boldsymbol{u}(t)$ for $t \in [t_i, t_{i+1}]$, and $\boldsymbol{x}_0 \approx \boldsymbol{x}(0)$. Denoting $h_i = t_{i+1} - t_i$, the continuous time optimal control problem is transcibed into the nonlinear constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{u}} \quad & \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \\ \text{s.t.} \quad & \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), i = 0, \ldots, N-1 \\ & \boldsymbol{x}_N \in \mathcal{M}_f \\ & \boldsymbol{u}_i \in \mathcal{U}, i = 0, \ldots, N-1 \end{aligned} \tag{215}$$

5.1.1 Consistency of Time Discretization

Having performed this discretization, a reasonable (and important) sanity check on the validity of the direct approach is whether we recover the original problem in the limit of $h_i \to 0$. For simplicity, we will drop the time-dependence of the cost and dynamics. We will write the Lagrangian for (215) as

$$\mathcal{L} = \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}_i, \boldsymbol{u}_i) + \sum_{i=0}^{N-1} \boldsymbol{\lambda}_i^T (\boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i) - \boldsymbol{x}_{i+1}). \tag{216}$$

Then, the KKT conditions are

$$0 = h_i \frac{\partial c}{\partial \boldsymbol{x}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + \boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1} + h_i \frac{\partial f}{\partial \boldsymbol{x}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i \tag{217}$$

$$0 = h_i \frac{\partial c}{\partial \boldsymbol{u}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) + h_i \frac{\partial f}{\partial \boldsymbol{u}_i}^T (\boldsymbol{x}_i, \boldsymbol{u}_i) \boldsymbol{\lambda}_i \tag{218}$$

Rearranging, we have

$$\frac{\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1}}{h_i} = -\frac{\partial f}{\partial \boldsymbol{x}_i}^T(\boldsymbol{x}_i, \boldsymbol{u}_i)\boldsymbol{\lambda}_i - \frac{\partial c}{\partial \boldsymbol{x}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{219}$$

$$0 = \frac{\partial f}{\partial \boldsymbol{u}_i}^T(\boldsymbol{x}_i, \boldsymbol{u}_i)\boldsymbol{\lambda}_i + \frac{\partial c}{\partial \boldsymbol{u}_i}(\boldsymbol{x}_i, \boldsymbol{u}_i). \tag{220}$$

Let $\boldsymbol{p}(t) = \boldsymbol{\lambda}_i$ for $t \in [t_i, t_{i+1}], i = 0, \dots, N-1$ and $p(0) = \lambda_0$. Then, the above are direct discretizations of the necessary conditions for (253),

$$\dot{\boldsymbol{p}}(t) = -\frac{\partial f}{\partial \boldsymbol{x}}^T(\boldsymbol{x}(t), \boldsymbol{u}(t))\boldsymbol{p}_i - \frac{\partial c}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{221}$$

$$0 = \frac{\partial f}{\partial \boldsymbol{u}}^T(\boldsymbol{x}(t), \boldsymbol{u}(t))\boldsymbol{p}(t) + \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{222}$$

5.2 Transcription Methods

A fundamental choice in the design of numerical algorithms for direct optimization of the discretized optimal control problem is whether to optimize over the state and action variables (a method known as collocation or simultaneous optimization) or strictly over the action variables (known as shooting).

5.2.1 Collocation Methods

Collocation methods optimize both the state variables and the control input at a fixed, finite number of times, $t_0, \dots, t_i, \dots, t_N$. Moreover, the dynamics constraints are enforced at these points. As such, it is necessary to choose a finite-dimensional representation of the trajectory between these points. This rough outline leaves unspecified a large number of algorithmic design choices.

First, how are the dynamics constraints enforced? Both derivative and integral constraints exist. The derivative approach enforces that the derivative of the state with respect to time of the parameterized trajectory is equal to the given system dynamics. The integral approach relies on integrating the given dynamics and enforcing agreement between this and the trajectory parameterization. In these notes, we will focus on the derivative approach.

Second, a choice of trajectory parameterization is required. We will primarily discuss Hermite-Simpson methods in herein, which parameterize each subinterval of the trajectory (in $[t_i, t_{i+1}]$) with a cubic polynomial. Note that the choice of a polynomial results in integral and derivative constraints being relatively simple to evaluate. However, a wide variety of parameterizations exist. For example, pseudospectral methods represent the entire trajectory as a single high-order polynomial.

We will now outline the Hermite-Simpson method as one example of direct collocation. Having selected a discretization $0 = t_0 < t_1 < \ldots < t_N = t_f$, we denote $h_i = t_{i+1} - t_i$. In every subinterval $[t_i, t_{i+1}]$, we approximate $\boldsymbol{x}(t)$ with a cubic polynomial

$$\boldsymbol{x}(t) = \boldsymbol{c}_0^i + \boldsymbol{c}_1^i(t - t_i) + \boldsymbol{c}_2^i(t - t_i)^2 + \boldsymbol{c}_3^i(t - t_i)^3 \tag{223}$$

which yields derivative

$$\boldsymbol{x}(t) = \boldsymbol{c}_1^i + 2\boldsymbol{c}_2^i(t - t_i) + 3\boldsymbol{c}_3^i(t - t_i)^2. \tag{224}$$

Writing $\boldsymbol{x}_i = \boldsymbol{x}(t_i), \boldsymbol{x}_{i+1} = \boldsymbol{x}(t_{i+1}), \dot{\boldsymbol{x}}_i = \dot{\boldsymbol{x}}(t_i), \dot{\boldsymbol{x}}_{i+1} = \dot{\boldsymbol{x}}(t_{i+1})$, we may write

$$\begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ I & h_i I & h_i^2 I & h_i^3 I \\ 0 & I & 2h_i I & 3h_i^2 I \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_0^i \\ \boldsymbol{c}_1^i \\ \boldsymbol{c}_2^i \\ \boldsymbol{c}_3^i \end{bmatrix} \tag{225}$$

which in turn results in

$$\begin{bmatrix} \boldsymbol{c}_0^i \\ \boldsymbol{c}_1^i \\ \boldsymbol{c}_2^i \\ \boldsymbol{c}_3^i \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ -\frac{3}{h_i^2} I & -\frac{2}{h_i} I & \frac{3}{h_i^2} I & -\frac{1}{h_i} I \\ \frac{2}{h_i^2} I & \frac{1}{h_i^2} I & -\frac{2}{h_i^3} I & \frac{1}{h_i^2} I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \dot{\boldsymbol{x}}_i \\ \boldsymbol{x}_{i+1} \\ \dot{\boldsymbol{x}}_{i+1} \end{bmatrix}. \tag{226}$$

Choosing intermediate times $t_i^c = t_i + \frac{h_i}{2}$ (collocation points), we can define interpolated controls $\boldsymbol{u}_i^c = \frac{\boldsymbol{u}_i + \boldsymbol{u}_{i+1}}{2}$. From the above, we have

$$\boldsymbol{x}_i^c := \boldsymbol{x}(t_i + \frac{h_i}{2}) = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{h_i}{8}(f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) - f(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}, t_{i+1})) \tag{227}$$

$$\dot{\boldsymbol{x}}_i^c := \dot{\boldsymbol{x}}(t_i + \frac{h_i}{2}) = -\frac{3}{2h_i}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) - \frac{1}{4}(f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) + f(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}, t_{i+1})). \tag{228}$$

Thus, we can write our discretized problem as

$$\begin{aligned} \min_{\boldsymbol{u}_{0:N-1}, \boldsymbol{x}_{0:N}} \quad & \sum_{i=0}^{N-1} h_i c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}_i^c - f(\boldsymbol{x}_i^c, \boldsymbol{u}_i^c, t_i^c) = 0, i = 0, \dots, N-1 \\ & F(\boldsymbol{x}_N) = 0 \\ & \boldsymbol{u}(t) \in \mathcal{U}, i = 0, \dots, N-1 \end{aligned} \tag{229}$$

5.2.2 Shooting Methods

Shooting methods solve the discrete optimization problem via optimizing only over the control inputs, and integrating the dynamics forward given these controls. A simple approach to the forward integration is the approach we have discussed above, in which forward Euler integration is used. Single-shooting methods directly optimize the controls for the entire problem. These approaches are fairly efficient for low dimension, short horizon problems, but typically struggle to scale to larger problems. Multiple shooting methods, on the other hand, optimize via shooting over subcomponents of the problem, and enforce agreement between the trajectory segments generated via shooting within each subproblem. These methods are therefore a combination of shooting methods and collocation methods. Generally, numerical solvers for shooting problems will, given an initial action sequence, linearize the trajectory and optimize the objective function with respect to those linearized dynamics to obtain new control inputs.

5.2.3 Sequential Convex Programming

Direct optimization of the discretized nonlinear control problem typically results in a non-convex optimization problem, for which finding a good solution may be difficult or impossible. The source of this non-convexity is typically the dynamics (and sometimes the cost function). The key idea of sequential convex programming (SCP) is to iterative re-linearize the dynamics (and construct a convex approximation of the cost function, if it is non-convex) around a nominal trajectory.

First, we will assume for this outline that the cost c is convex. Let $(\boldsymbol{x}_0(\cdot), \boldsymbol{u}_0(\cdot))$ be a nominal tuple of trajectory and control (which is not necessarily feasible). We linearize the dynamics around this trajectory:

$$f_1(\boldsymbol{x}, \boldsymbol{u}, t) = f(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t) + \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t)(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \frac{\partial f}{\partial \boldsymbol{u}}(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), t)(\boldsymbol{u} - \boldsymbol{u}_0(t)). \tag{230}$$

We can then solve the linear optimal control problem (with $k = 0$, initially),

$$\begin{aligned} \min \quad & \int_0^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f_{k+1}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), t \in [0, t_f] \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \qquad\qquad\qquad\qquad \boldsymbol{x}(t_f) = \boldsymbol{x}_f \\ & \boldsymbol{u}(t) \in \mathcal{U}, t \in [0, t_f] \end{aligned} \tag{231}$$

where the dynamics are linear and the cost function is quadratic. Discretizing this continuous control problem yields a tractable convex optimization problem with dynamics $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i f(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i), i = 0, \ldots, N-1$. We then iterate this procedure until convergence is achieved with the new trajectory.

5.3 Further Reading

A broad introduction to direct methods for trajectory optimization is presented in [Kel17b]. This tutorial also features a discussion of trajectory optimization for hybrid systems, which we have not discussed in this section, as well as numerical solver features. For a more comprehensive review of direct methods for trajectory optimization by the same author with an emphasis on collocation methods, see [Kel17a].

6 Model Predictive Control

Both direct and indirect methods for open-loop control result in trajectories that must be tracked with an auxiliary controller, if there is any mismatch between the systems model and the true system. This often results in a decoupling of the auxiliary controller from the original optimal control problem, which may result in performance degradation. Alternatively, the auxiliary controller may not be able to take into account other problem considerations such as state or control constraints. In this section, we introduce model predictive control, which applies the ideas from direct methods for trajectory generation online to iteratively replan, and thus results in a closed-loop controller.

6.1 Overview of MPC

Model predictive control entails solving finite-time optimal control problems in a receding horizon fashion (and thus is also frequently referred to as *receding horizon control*). The rough structure of model predictive control algorithms is

- At each sampling time t, solve an *open-loop* optimal control problem over a finite horizon
- Apply the generated optimal input signal during the subsequent sampling interval $[t, t+1)$
- At the next time step $t+1$, solve the new optimal control problem based on new measurements of the state over a shifted horizon

Consider the problem of regulating to the origin the discrete-time linear time-invariant system

$$\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \tag{232}$$

for $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$, subject to constraints $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}, t \geq 0$, where $\mathcal{X}$,$\mathcal{U}$ are polyhedra. We will assume the full state measurement is available at time t. Given this, we can state the finite-time optimal control problem solved at each stage, t, as

$$\begin{aligned}
\min_{\boldsymbol{u}_{t|t},\dots,\boldsymbol{u}_{t+N-1|t}} \quad & c_f(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} c(\boldsymbol{x}_{t+k|t}, \boldsymbol{u}_{t+k|t}) \\
\text{s.t.} \quad & \boldsymbol{x}_{t+k+1|t} = A\boldsymbol{x}_{t+k|t} + B\boldsymbol{u}_{t+k|t}, \quad k = 0, \dots, N-1 \\
& \boldsymbol{x}_{t+k|t} \in \mathcal{X}, \quad k = 0, \dots, N-1 \\
& \boldsymbol{u}_{t+k|t} \in \mathcal{U}, \quad k = 0, \dots, N-1 \\
& \boldsymbol{x}_{t+N|t} \in \mathcal{X}_f, \\
& \boldsymbol{x}_{t|t} = \boldsymbol{x}(t)
\end{aligned} \tag{233}$$

for which we write the solution as $J_t^*(\boldsymbol{x}(t))$. In this problem, $\boldsymbol{x}_{t+k|t}$ and $\boldsymbol{u}_{t+k|t}$ are the state and action predicted at time $t+k$ from time t. Letting $U^*_{t\to t+N|t} := \{\boldsymbol{u}^*_{t|t}, \dots, \boldsymbol{u}^*_{t+N-1|t}\}$

denote the optimal solution, we take $\boldsymbol{u}(t) = \boldsymbol{u}^*_{t|t}(\boldsymbol{x}(t))$. This optimization problem is then repeated at time $t+1$, based on the new state $\boldsymbol{x}_{t+1|t+1} = \boldsymbol{x}(t+1)$. Defining the closed-loop control policy as $\pi_t(\boldsymbol{x}(t)) := \boldsymbol{u}^*_{t|t}(\boldsymbol{x}(t))$, we have the closed-loop dynamics

$$\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\pi_t(\boldsymbol{x}(t)). \tag{234}$$

Thus, the central question of this formulation becomes characterizing the behavior of the closed-loop system defined by this iterative re-optimization. As the problem is time-invariant, we can rewrite the closed-loop dynamics as

$$\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\pi(\boldsymbol{x}(t)). \tag{235}$$

The rough structure of the online model predictive control framework is then as follows:

1. Measure the state $\boldsymbol{x}(t)$ at every time t
2. Obtain $U_0^*(\boldsymbol{x}(t)$ by solving finite-time optimal control problem
3. If $U_0^*(\boldsymbol{x}(t) = \emptyset$ then 'problem infeasible', stop
4. Apply the first element $\boldsymbol{u}_0^*$ of $U_0^*(\boldsymbol{x}(t)$ to the system
5. Wait for the new sampling time $t+1$

This framework leads to two main implementation issues. First, the controller may lead us into a situation where after a few steps the finite-time optimal control problem is infeasible, which we refer to as the *persistent feasibility issue*. Even if the feasibility problem does not occur, the generated control inputs may not lead to trajectories that converge to the origin, which we refer to as the *stability issue*. The key question in the analysis of MPC algorithms is how we may guarantee that our "short-sighted" control strategy leads to effective long-term behavior. While one possible approach is directly analyzing the closed-loop dynamics, this is in practice very difficult. Our approach will instead be to derive conditions on the terminal function c_f and terminal constraint set $\mathcal{X}_f$ so that the persistent feasibility and closed-loop stability are guaranteed.

6.2 Feasibility

Model predictive control simplifies the online control optimization problem by solving a shorter horizon problem, as opposed to solving the full optimal control problem online at each timestep. This myopic optimization leads to the possibility that after several steps, the problem may no longer be feasible. As such, in this section we will discuss approaches to impose constraints on so-called *recursive feasibility* to avoid this problem.

Let

$$\begin{aligned}\mathcal{X}_0 := \{\boldsymbol{x} \in \mathcal{X} \mid \exists(\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}) \text{ s.t. } &\boldsymbol{x}_k \in \mathcal{X}, \boldsymbol{u}_k \in \mathcal{U}, k = 0, \dots, N-1, \\ &\boldsymbol{x}_N \in \mathcal{X}_f, \text{ where } \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k, k = 0, \dots, N-1\}\end{aligned} \tag{236}$$

be the set of feasible initial states. Simply, this set is the set of initial states for which a sequence of control inputs exist that cause the final state to satisfy the terminal constraint. For the autonomous system $\boldsymbol{x}(t+1) = \phi(\boldsymbol{x}(t))$ with constraints $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}$, the one-step controllable set to set $\mathcal{S}$ is defined as

$$\text{Pre}(\mathcal{S}) := \{\boldsymbol{x} \in \mathbb{R}^n : \phi(\boldsymbol{x}) \in \mathcal{S}\}. \tag{237}$$

For the system $\boldsymbol{x}(t+1) = \phi(\boldsymbol{x}(t), \boldsymbol{u}(t))$ with constraints $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}$, the one-step controllable set to set $\mathcal{S}$ is defined as

$$\text{Pre}(\mathcal{S}) := \{\boldsymbol{x} \in \mathbb{R}^n : \exists \boldsymbol{u} \in \mathcal{U} \text{ s.t. } \phi(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{S}\}. \tag{238}$$

A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a control invariant set for the system $\boldsymbol{x}(t+1) = \phi(\boldsymbol{x}(t), \boldsymbol{u}(t))$ with constraints $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}$, if

$$\boldsymbol{x}(t) \in \mathcal{C} \implies \exists \boldsymbol{u} \in \mathcal{U} \text{ s.t. } \phi(\boldsymbol{x}(t), \boldsymbol{u}(t)) \in \mathcal{C}, \forall t. \tag{239}$$

The set $\mathcal{C}_\infty \subset \mathcal{X}$ is said to the maximal control invariant set for the system $\boldsymbol{x}(t+1) = \phi(\boldsymbol{x}(t), \boldsymbol{u}(t))$ with constraints $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}$, if it control invariant all control invariant sets contained in $\mathcal{X}$[1].

We will now proceed to derive critical results on recursive feasibility for linear dynamical systems. We will define the "truncated" feasibility set

$$\begin{aligned} \mathcal{X}_1 := \{\boldsymbol{x} \in \mathcal{X} \mid \exists(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}) \text{ s.t. } & \boldsymbol{x}_k \in \mathcal{X}, \boldsymbol{u}_k \in \mathcal{U}, k = 1, \ldots, N-1, \\ & \boldsymbol{x}_N \in \mathcal{X}_f, \text{ where } \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k, k = 1, \ldots, N-1\}. \end{aligned} \tag{240}$$

Then, we may state the following result on feasibility.

Lemma 6.1 (Persistent Feasibility)**.** *If set $\mathcal{X}_1$ is a control invariant set for system $\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t)$, $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}$, then the MPC law is persistently feasible.*

Proof. Note that

$$\text{Pre}(\mathcal{X}_1) := \{\boldsymbol{x} \in \mathbb{R}^n : \exists \boldsymbol{u} \in \mathcal{U} \text{ s.t. } A\boldsymbol{x} + B\boldsymbol{u} \in \mathcal{X}_1\}. \tag{241}$$

Since $\mathcal{X}_1$ is control invariant, there exists $\boldsymbol{u} \in \mathcal{U}$ such that $A\boldsymbol{x} + B\boldsymbol{u} \in \mathcal{X}_1$ for all $\boldsymbol{x} \in \mathcal{X}_1$. Thus, $\mathcal{X}_1 \subseteq \mathcal{X}_1 \cap \mathcal{X}$. One may write

$$\mathcal{X}_0 = \{\boldsymbol{x}_0 \in \mathcal{X} \mid \exists \boldsymbol{u}_0 \in \mathcal{U} \text{ s.t. } A\boldsymbol{x}_0 + B\boldsymbol{u}_0 \in \mathcal{X}_1\} = \text{Pre}(\mathcal{X}_1) \cap \mathcal{X}. \tag{242}$$

This then implies $\mathcal{X}_1 \subseteq \mathcal{X}_0$. Choose some $\boldsymbol{x}_0 \in \mathcal{X}_0$. Let U_0^* be the solution to the finite-time optimization problem, and $\boldsymbol{u}_0^*$ be the first control. Let $\boldsymbol{x}_1 = A\boldsymbol{x}_0 + B\boldsymbol{u}_0^*$. Since U_0^* is feasible, one has $\boldsymbol{x}_1 \in \mathcal{X}_1$. Since $\mathcal{X}_1 \subseteq \mathcal{X}_0$, $\boldsymbol{x}_1 \in \mathcal{X}_0$, and hence the next optimization problem is feasible. □

[1]Control invariant sets can be computed using the MPT toolbox: `www.mpt3.org`

For $N = 1$, we may set $\mathcal{X}_f = \mathcal{X}1$. If the terminal set is chosen to be control invariant, then MPC problem will be persistently feasible *independent* of chosen control objectives and parameters. The system designer may then choose the parameters to affect the system performance. The logical question, then, is how to extent this result to $N > 1$, for which we have the following result.

Theorem 6.2 (Persistent Feasibility)**.** *If $\mathcal{X}_f$ is a control invariant set for the the system $\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t)$, $\boldsymbol{x}(t) \in \mathcal{X}, \boldsymbol{u}(t) \in \mathcal{U}, t \geq 0$, then the MPC law is persistently feasible.*

Proof. We will begin by defining the "truncated" feasibility set at step $N-1$,

$$\begin{aligned}\mathcal{X}_{N-1} := \{\boldsymbol{x}_{N-1} \in \mathcal{X} \mid \exists \boldsymbol{u}_{N-1} \text{ s.t. } \boldsymbol{x}_{N-1} \in \mathcal{X}, \boldsymbol{u}_{N-1} \in \mathcal{U} \\ \boldsymbol{x}_N \in \mathcal{X}_f, \text{ where } \boldsymbol{x}_N = A\boldsymbol{x}_{N-1} + B\boldsymbol{u}_{N-1}\}.\end{aligned} \tag{243}$$

Due to the terminal constraints, have $A\boldsymbol{x}_{N-1} + B\boldsymbol{u}_{N-1} = \boldsymbol{x}_N \in \mathcal{X}_f$. Since $\mathcal{X}_f$ is a control invariant set, there exists a $\boldsymbol{u} \in \mathcal{U}$ such that $\boldsymbol{x}^+ = A\boldsymbol{x}_N + B\boldsymbol{u} \in \mathcal{X}_f$. This is the requirement that $\boldsymbol{x}_N \in \mathcal{X}_{N-1}$. Thus, $\mathcal{X}_{N-1}$ is control invariant. Repeating this argument, one can recursively show that $\mathcal{X}_{N-2}, \ldots, \mathcal{X}_1$ are control invariant, and the persistent feasibility lemma then applies. □

Practically, we introduce the terminal set $\mathcal{X}_f$ artificially for the purpose of leading to a sufficient condition for persistent feasibility. We would like to choose it to be large, so that is avoids compromising closed-loop performance.

6.3 Stability

Persistent feasibility does not guarantee that the closed-loop trajectories converge toward the desired equilibrium point. One of the most popular approaches to guarantee persistent feasibility and stability of the MPC law makes use of a control invariant terminal set $\mathcal{X}_f$ for feasibility, and a terminal function $c_f(\cdot)$ for stability. To prove stability, we leverage Lyapunov stability theory.

Theorem 6.3 (Lyapunov Stability)**.** *Consider the equilibrium point $\boldsymbol{x} = 0$ for the autonomous system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$ (with $f(0) = 0$). Let $\Omega \subset \mathbb{R}^n$ be a closed and bounded set containing the origin. Let $V : \mathbb{R}^n \to \mathbb{R}$ be a function, continuous at the origin, such that*

$$V(0) = 0 \textit{ and } V(\boldsymbol{x}) > 0, \ \forall \boldsymbol{x} \in \Omega \setminus \{0\} \tag{244}$$

$$V(\boldsymbol{x}_{k+1}) - V(\boldsymbol{x}_k) < 0, \ \forall \boldsymbol{x} \in \Omega \setminus \{0\}. \tag{245}$$

Then $\boldsymbol{x} = 0$ is asymptotically stable in Ω.

We will utilize this result to show that with appropriate choices of $\mathcal{X}_f$ and $c_f(\cdot)$, J_0^* is a Lyapunov function for the closed-loop system.

Theorem 6.4 (MPC Stability (for Quadratic Cost))**.** *Assume*

1. $Q = Q^T > 0, R = R^T > 0, Q_f > 0$

2. Sets $\mathcal{X}, \mathcal{X}_f$*, and* $\mathcal{U}$ *contain the origin in their interior and are closed*

3. $\mathcal{X}_f \subseteq \mathcal{X}$ *is control invariant*

4. $\min_{\boldsymbol{v}\in\mathcal{U}, A\boldsymbol{x}+B\boldsymbol{v}\in\mathcal{X}_f} \{-c_f(\boldsymbol{x}) + c(\boldsymbol{x},\boldsymbol{v}) + c_f(A\boldsymbol{x}+B\boldsymbol{v})\} \le 0, \forall \boldsymbol{x} \in \mathcal{X}_f$.

Then, the origin of the closed-loop system is asymptotically stable with domain of attraction $\mathcal{X}_f$*.*

Proof. Note that via assumption 3, persistent feasibility is guaranteed for any Q_f, Q, R. We want to show that J_0^* is a Lyapunov function for the closed-loop system $\boldsymbol{x}(t+1) = f_{cl}(\boldsymbol{x}(t)) = A\boldsymbol{x}(t) + B\pi(\boldsymbol{x}(t))$, with respect to the equilibrium $f_{cl}(0) = 0$ (the origin is indeed an equilibrium as $0 \in \mathcal{X}, 0 \in \mathcal{U}$, and the cost is positive for any non-zero control sequence. Note also that $\mathcal{X}_0$ is closed and bounded, and $J_0^*(0) = 0$, both by assumption. Note also that $J_0^*(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathcal{X}_0 \setminus \{0\}$.

We will now show the decay property. Since the setup is time-invariant, we can study the decay property between $t = 0$ and $t = 1$. Let $\boldsymbol{x}(0) \in \mathcal{X}_0$, let $U_0^{[0]} = [\boldsymbol{u}_0^{[0]}, \dots, \boldsymbol{u}_{N-1}^{[0]}]$ be the optimal control sequence, and let $[\boldsymbol{x}(0), \dots, \boldsymbol{x}_N^{[0]}]$ be the corresponding trajectory. After applying $\boldsymbol{u}_0^{[0]}$, one obtains $\boldsymbol{x}(1) = A\boldsymbol{x}(0) + B\boldsymbol{u}_0^{[0]}$. Now, consider the sequence of control $[\boldsymbol{u}_1^{[0]}, \dots, \boldsymbol{u}_{N-1}^{[0]}, \boldsymbol{v}]$, where $\boldsymbol{v} \in \mathcal{U}$ and the corresponding state trajectory is $[\boldsymbol{x}(1), \dots, \boldsymbol{x}_N^{[0]}, A\boldsymbol{x}_N^{[0]} + B\boldsymbol{V}]$. Since $\boldsymbol{x}_N^{[0]} \in \mathcal{X}_f$ (by the terminal constraint), and since $\mathcal{X}_f$ is control invariant,

$$\exists \bar{\boldsymbol{v}} \in \mathcal{U} \mid A\boldsymbol{x}_N^{[0]} + B\bar{\boldsymbol{v}} \in \mathcal{X}_f. \tag{246}$$

With such a choice of $\bar{\boldsymbol{v}}$, the sequence $[\boldsymbol{u}_1^{[0]}, \dots, \boldsymbol{u}_{N-1}^{[0]}, \boldsymbol{v}]$ is feasible for the MPC optimization problem at time $t = 1$. Subce tgus sequence is not necessarily optimal,

$$J_0^*(\boldsymbol{x}(1)) \le c_f(A\boldsymbol{x}_N^{[0]} + B\bar{\boldsymbol{v}}) + \sum_{k=1}^{N-1} c(\boldsymbol{x}_k^{[0]}, \boldsymbol{u}_k^{[0]}) + c(\boldsymbol{x}_N^{[0]}, \boldsymbol{v}). \tag{247}$$

Equivalently,

$$J_0^*(\boldsymbol{x}(1)) \le c_f(A\boldsymbol{x}_N^{[0]} + B\bar{\boldsymbol{v}}) + J_0^*(\boldsymbol{x}(0)) - c_f(\boldsymbol{x}_N^{[0]}) - c(\boldsymbol{x}(0), \boldsymbol{u}_0^{[0]}) + c(\boldsymbol{x}_N^{[0]}, \bar{\boldsymbol{v}}) \tag{248}$$

Since $\boldsymbol{x}_N^{[0]} \in \mathcal{X}_f$ by assumption, we can select $\bar{\boldsymbol{v}}$ such that

$$J_0^*(\boldsymbol{x}(1)) \le J_0^*(\boldsymbol{x}(0)) - c(\boldsymbol{x}(0), \boldsymbol{u}_0^{[0]}). \tag{249}$$

Since $c(\boldsymbol{x}(0), \boldsymbol{u}_0^{[0]}) > 0$ for all $\boldsymbol{x}(0) \in \mathcal{X}_0 \setminus \{0\}$,

$$J_0^*(\boldsymbol{x}(1)) - J_0^*(\boldsymbol{x}(0)) < 0. \tag{250}$$

The last step is to prove continuity, for which we omit the details and refer the reader to [BBM17]. □

6.3.1 Choosing $\mathcal{X}_f$ and Q_f

We will look at two cases. First, we will assume that A is asymptotically stable. Then, we set $\mathcal{X}_f$ as the maximally positive invariant set $\mathcal{O}_\infty$ for the system $\boldsymbol{x}(t+1) = A\boldsymbol{x}(t), \boldsymbol{x}(t) \in \mathcal{X}$. The set $\mathcal{X}_f$ is a control invariant set for the system $\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t)$ as $\boldsymbol{u} = 0$ is a feasible control. As for stability, $\boldsymbol{u} = 0$ is feasible and $A\boldsymbol{x} \in \mathcal{X}_f$ if $\boldsymbol{x} \in \mathcal{X}_f$, thus assumption 4 of Theorem 6.3 becomes

$$-\boldsymbol{x}^T Q_f \boldsymbol{x} + \boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{x}^T A^T Q_f A \boldsymbol{x} \le 0, \forall \boldsymbol{x} \in \mathcal{X}_f \tag{251}$$

which is true since, due to the fact that A is asymptotically stable,

$$\exists Q_f > 0 \mid -Q_f + Q + A^T Q_f A = 0 \tag{252}$$

Next, we will look at the general case. Let L_∞ be the optimal gain for the infinite-horizon LQR controller. Set $\mathcal{X}_f$ as the maximal positive invariant set for system $\boldsymbol{x}(t+1) = (A + BL_\infty)\boldsymbol{x}(t)$ (with constraints $\boldsymbol{x}(t) \in \mathcal{X}, F_\infty \boldsymbol{x}(t) \in \mathcal{U}$). Then, set Q_f as the solution Q_∞ to the discrete-time Riccati equation.

6.3.2 Explicit MPC

In some cases, the MPC law can be pre-computed, which removes the need for online optimization. An important case of this is that of constrained LQR, in which we wish to solve the optimal control problem

$$\begin{array}{ll}
\min\limits_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1}} & \boldsymbol{x}_N^T Q_f \boldsymbol{x}_N + \sum\limits_{k=0}^{N-1} \boldsymbol{x}_k^T Q \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k \\
\text{s.t.} & \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k, \quad k = 0, \ldots, N-1 \\
& \boldsymbol{x}_k \in \mathcal{X}, \quad k = 0, \ldots, N-1 \\
& \boldsymbol{u}_k \in \mathcal{U}, \quad k = 0, \ldots, N-1 \\
& \boldsymbol{x}_N \in \mathcal{X}_f, \\
& \boldsymbol{x}_0 = \boldsymbol{x}
\end{array} \tag{253}$$

The solution to the constrained LQR problem is a control $\boldsymbol{u}^*$ which is a continuous piecewise affine function on polyhedral partition of the state space $\mathcal{X}$, that is $\boldsymbol{u}^* = \pi(\boldsymbol{x})$, where

$$\pi(\boldsymbol{x}) = L^j \boldsymbol{x} + \boldsymbol{l}^j \text{ if } H^j \boldsymbol{x} \le K^j,\ j = 1, \ldots, N^r. \tag{254}$$

Thus, online, one has to locate in which cell of the polyhedral partition the state $\boldsymbol{x}$ lies, and then one obtains the optimal control via a look-up table query.

6.4 Further Reading

We refer the reader to [BBM17] and [RMD17] for two broad and comprehensive treatments of the topic.

7 Adaptive Optimal Control

7.1 Problem Statement

7.2 System Identification

7.3 Adaptive Control

7.4 Probing, Planning for Information Gain, and Dual Control

7.5 Further Reading

8 Reinforcement Learning

8.1 Model-Based Approaches: Linear Systems

8.1.1 Adaptive LQR

8.1.2 Learning-Based MPC

8.1.3 Time-Varying Linear Models

8.2 Model-Based Approaches: Nonlinear Systems

8.3 Model-Free Approaches

8.3.1 Temporal Difference Learning

8.3.2 Policy Gradient

8.4 Connections Between Model-Based and Model-Free Methods

8.4.1 Model-Based Acceleration for Model-Free RL

8.4.2 Model-Free RL for Terminal Costs

8.4.3 LQR via Q-Learning

8.5 Exploration

8.6 Safety

8.6.1 Risk-Sensitive RL

8.6.2 Stability Analysis

8.6.3 Safe Exploration

8.7 Further Reading

References

[AM07] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods.* Courier Corporation, 2007.

[BBM17] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[Ber12] Dimitri P Bertsekas. *Dynamic programming and optimal control*. Number 1. 4 edition, 2012.

[Ber16] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 2016.

[BH75] Arthur Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975.

[Bre10] Alberto Bressan. Noncooperative differential games. a tutorial. 2010.

[BT97] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[GB17] Markus Giftthaler and Jonas Buchli. A projection approach to equality constrained iterative linear quadratic optimal control. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 2017.

[JM70] David Jacobson and David Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

[Kel17a] Matthew Kelly. An introduction to trajectory optimization: how to do your own direct collocation. *SIAM Review*, 2017.

[Kel17b] Matthew Kelly. Transcription methods for trajectory optimization: a beginners tutorial. *arXiv:1707.00284*, 2017.

[Kir12] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[Kol08] Zico Kolter. Convex optimization overview. *CS 229 Lecture Notes*, 2008.

[LK14] Sergey Levine and Vladlen Koltun. Learning complex neural network policies with trajectory optimization. In *International Conference on Machine Learning (ICML)*, 2014.

[LM67] Ernest Bruce Lee and Lawrence Markus. Foundations of optimal control theory. 1967.

[LS92] Li-zhi Liao and Christine A Shoemaker. Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems. Technical report, Cornell University, 1992.

[LT04] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation, and Robotics*, 2004.

[MBT05] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 2005.

[Pow12] Warren B Powell. AI, OR and control theory: A rosetta stone for stochastic optimization. 2012.

[RMD17] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[TET12] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[TL05] Emanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference (ACC)*, 2005.

[TMT14] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[XLH17] Zhaoming Xie, C Karen Liu, and Kris Hauser. Differential dynamic programming with nonlinear constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.