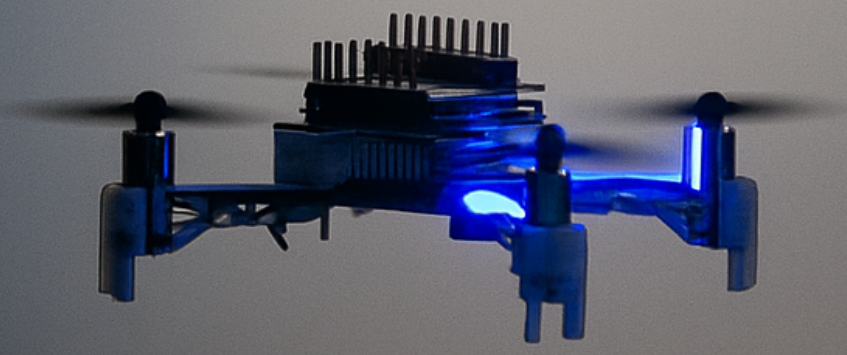
CONTROL FOR ROBOTICS

From Optimal Control to Reinforcement Learning



The chapters of the book are being released sequentially
over the course of Summer 2026.

Control for Robotics

From Optimal Control to Reinforcement Learning

Angela P. Schoellig and Siqi Zhou

with contributions from

Lukas Brunke
Martin Schuck
Adam W. Hall
Haocheng Zhao
Ralf Römer
Oliver Hausdörfer

Release Version: 2026 Summer V0.1

Contents

Preface **iii**

Acronyms **v**

Notation **vii**

1 Introduction to Optimal Control **1**
1.1 Problem Statement 1
1.2 The Dynamic Programming Algorithm 5
1.3 Exercises 19

2 Linear Quadratic Optimal Control **23**
2.1 Finite-Horizon Discrete-Time linear quadratic regulator (LQR) 23
2.2 Infinite-Horizon Problems 27
2.3 Infinite-Horizon Discrete-Time LQR 31
2.4 Continuous-Time Counterparts 35
2.5 Formulating a Linear Quadratic Optimal Control Problem 38
2.6 Verifying Stabilizability and Detectability Conditions 41
2.7 Exercises 46

3 Optimization Fundamentals **53**
3.1 Why is Optimization Relevant? 53
3.2 Problem Definition 55
3.3 Unconstrained Optimization 56
3.4 Constrained Optimization 61
3.5 Exercises 80

4 Iterative Optimal Control Algorithms **85**
4.1 Iterative Approximate Optimal Control Solutions: The Core Idea . . 86
4.2 Iterative Linear Quadratic Regulator (ILQR) 87
4.3 Connection to Discrete-Time LQR 96
4.4 Exercises 99

5 Model Predictive Control **103**
5.1 Motivation from Real-World Applications 103
5.2 Formulation of a Model Predictive Controller 109
5.3 Stability and Feasibility of Model Predictive Control 118

5.4 Model Predictive Control in Practice . . . 125
5.5 Robust Model Predictive Control . . . 130
5.6 Exercises . . . 136

6 Model Learning and Learning-Based Control 149
6.1 Model Learning in Robotics . . . 149
6.2 Deterministic Models . . . 153
6.3 Probabilistic Models . . . 164
6.4 Practical Notes . . . 168
6.5 Learning-Based Model Predictive Control . . . 169
6.6 Exercises . . . 181

7 Introduction to Reinforcement Learning 187
7.1 Problem Formulation . . . 188
7.2 The Bellman Equation . . . 191
7.3 Model-Based RL . . . 194
7.4 Model-Free RL . . . 203
7.5 Exercises . . . 211

8 Deep Reinforcement Learning 223
8.1 Deep Q-Learning . . . 223
8.2 Policy Gradient Methods . . . 233
8.3 Trust Region Algorithms . . . 239
8.4 Exercises . . . 251

9 Quadrotor Case Study 253
9.1 Quadrotor Stabilization in 2D Space . . . 253
9.2 Quadrotor Trajectory Tracking in 3D Space . . . 266
9.3 Programming Exercises: State-of-the-Art Algorithms . . . 269
9.4 Programming Exercises: A Drone Racing Challenge with Sim-to-Real Experiments . . . 271

Epilogue: The Roads Ahead Toward Embodied Intelligence 273

A Continuous-Time Optimal Control 275
A.1 Continuous-Time Optimal Control . . . 275
A.2 Hamilton-Jacobi-Bellman (HJB) Equation . . . 276
A.3 The Pontryagin Minimum Principle . . . 278

B Quadrotor Dynamics and Control 285
B.1 Overview . . . 285
B.2 Quadrotor Dynamics . . . 285
B.3 Quadrotor Control . . . 290
B.4 Smooth Trajectory Generation . . . 292
B.5 Optimal Control Problem Beyond Tracking . . . 297
B.6 Further Reading . . . 303

Preface

This book was initially developed as part of the graduate-level course, "Control for Robotics" offered at the University of Toronto (Canada) in 2020, within a suite of four courses that respectively cover perception, estimation, planning, and control. The material was later adapted and expanded for the "Optimal Control and Decision-Making" course at the Technical University of Munich (Germany), with further advanced topics incorporated into two followup courses, "Advanced Topics in Robot Learning and Decision-Making" and "Autonomous Drone Racing," offered at the same institution. Together, this sequence of three courses progresses from the foundations of robot control supported by interactive examples, to state-of-the-art approaches in robot learning and decision-making accompanied by hands-on programming exercises, and finally to pushing the limits of current methods with real-world deployment. Over several years of teaching, we refined and consolidated the set of course materials into a coherent journey from theory to practice, making it suitable for students and practitioners interested in pursuing professional careers in robotics or research in related fields.

Control or decision-making is a foundational component of any robotic software stack. Most robots operating today are *feedback control systems*. The field of control theory provides the basis for understanding the properties of these "feedback loops" and offers a set of mathematical tools for ensuring desired properties such as stability, robustness and performance are satisfied. A key branch within control is optimal decision-making, which has evolved in parallel within both the controls and reinforcement learning communities. Recent advances have begun to unify ideas from these two domains, offering new perspectives and algorithms for learning-based control in robotics [1].

This book provides an overview of the fundamentals of these methods, with particular emphasis on both theory and practical insights relevant to robotics applications. The chapters begin with the principle of optimality and proceed to derive practical algorithms widely used in robotics, such as the linear quadratic regulator and model predictive control. We then trace the progression from classical model-based control methods to modern learning-based approaches that address dynamic uncertainties and all the way to model-free reinforcement learning approaches that optimize policies based on interaction with the environment. Along the way, we highlight the connections between model-based control and model-free reinforcement learning, illustrating how ideas from

both communities can inform the design of effective decision-making systems for robots.

With this foundation, readers will be equipped with the tools and methods needed to develop controllers in both structured settings where system dynamics are known, and uncertain scenarios where full system knowledge is not available. To better illustrate the methods presented in the book and to help readers understand their respective advantages and limitations, we provide a collection of interactive Jupyter notebook examples (referenced in blue boxes) showcasing the application of the algorithms in robotics. For readers interested in further exploration, we include a set of advanced topics in the book, which are marked with an asterisk $(^*)$ after the corresponding section titles. The final chapter of the book presents a quadrotor case study that demonstrates the application of the core methods on a practical robotic platform, together with a set of programming exercises (referenced in orange boxes) and a hands-on autonomous drone racing project (referenced in light beige boxes). A summary of pertinent resources, along with our ongoing updates, is available on our website: https://learnsyslab.github.io/cfr/. For readers interested in related applications from our research, please visit www.learnsyslab.org.

This book and the courses built upon its content would not have been possible without the collective efforts of several outstanding graduate students. We specifically acknowledge Lukas Brunke, who contributed to the model predictive control (MPC) material in Chapter 5, Adam Hall, who supported the learning-based MPC section in Chapter 6, and Martin Schuck, who contributed to the deep reinforcement learning content in Chapter 8. The accompanying interactive Jupyter notebook examples are the result of dedicated efforts from Haocheng Zhao and Lukas Brunke, while the exercises and solutions at the end of each main chapter involved support from Ralf Römer, Luca Worbis, and Barry Yeh. Additionally, we thank Oliver Hausdörfer for his valuable feedback on the advanced topics covered in the book. The accompanying codebases for the advanced programming exercises and the autonomous drone racing project introduced in Chapter 9 were developed under the technical lead of Oliver Hausdörfer and Martin Schuck, respectively. These codebases also benefited from the support of many additional members over several iterations of the courses, including Marcel Rath, Yufei Hua, Niklas Schlüter, Jiaming Zhang, Luca Worbis, Timo Class, and Yi Lu.

Acronyms

BFGS Broyden-Fletcher-Goldfarb-Shanno

BNN Bayesian Neural Network

CARE Continuous-Time Algebraic Riccati Equation

CG Conjugate Gradient

COM Center of Mass

DARE Discrete-Time Algebraic Riccati Equation

DDPG Deep Deterministic Policy Gradient

DPA Dynamic Programming Algorithm

DQN Deep Q-Network

DRL Deep Reinforcement Learning

GAE Generalized Advantage Estimation

GP Gaussian Process

HJB Hamilton–Jacobi–Bellman

i.i.d. Independent and Identically Distributed

ILQR Iterative Linear Quadratic Regulator

IMU Inertial Measurement Unit

KKT Karush–Kuhn–Tucker

KL Kullback-Leibler

LP Linear Program

LQR Linear Quadratic Regulator

MDP Markov Decision Process

MPC Model Predictive Control

NN Neural Network

ODE Ordinary Differential Equation

PBH Popov-Belevitch-Hautus

PD Proportional-Derivative

PDE Partial Differential Equation

PMP Pontryagin Minimum Principle

PPO Proximal Policy Optimization

PWM Pulse-Width Modulation

QP Quadratic Program

RK4 Runge-Kutta Fourth-Order Method

RL Reinforcement Learning

RMPC Robust Model Predictive Control

RPM Revolutions Per Minute

SAC Soft Actor-Critic

SE Squared Exponential

SQP Sequential Quadratic Programming

SSP Stochastic Shortest Path

TRPO Trust Region Policy Optimization

ZOH Zero-Order-Hold Discretization

Notation

The following lists summarize the key notation used in the book. We adhere to notational conventions whenever possible; however, it is difficult to avoid occasional overloading of symbols. For specific cases, please refer to the notations defined within the respective chapters.

General Notation

$\|\cdot\|$	Vector or matrix norm
$\|\cdot\|$	Absolute value
min, max	Minimum and maximum operators
inf, sup	Infimum and supremum operators
arg min, arg max	Argument of the minimum and maximum operators
$\dot{(\cdot)}$	Time derivative
$\nabla_{\cdot}$	Partial derivative operator
$\nabla_{\cdot}^2$	Second partial derivative operator
$\nabla_{\cdot}^T$	Partial derivative transpose operator
$\mathbf{diag}(\cdot)$	Diagonal matrix with corresponding diagonal elements
$\mathbf{sgn}(\cdot)$	Sign function
$\mathbf{rank}(\cdot)$	Rank operator
$\lambda(\cdot)$	Eigenvalue
$\sigma(\cdot)$	Spectrum operator
$\mathbf{Re}(\cdot)$, $\mathbf{Im}(\cdot)$	Real and imaginary part operators
$\exp(\cdot)$	Exponential function
$\log(\cdot)$	Logarithm function

$\mathbb{E}$	Expected value operator
subject to	Subject to
$\cdot \mid \cdot$	Conditional operator
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integers
$\mathbf{1}$	Vector of ones
$\mathbf{I}$	Identity matrix
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\delta\cdot$	Perturbation or change in a variable or function
∞	Infinity
$\mathcal{O}(\cdot)$	Big O notation
$(\cdot)^*$	Optimal solution

Optimal Control and Decision-Making

k	Discrete-time index
t	Continuous-time variable
N	Time horizon
x	System state
u	Control input or action
y	Output or observation
w	Process noise or disturbance
v	Measurement noise
$\mathcal{X}$	Set of Admissible states
$\mathcal{U}$	Set of Admissible inputs
$\mathcal{D}$	Noise or disturbance set
Π	Set of admissible policies
$\mathcal{X}_\mathrm{f}$	Terminal set

$f(\cdot,\cdot,\cdot)$	System dynamics
A, B, C	Linear system matrices
$\ell_k(\cdot,\cdot), \ell_N(\cdot)$	Stage and terminal cost functions
Q, R, P	Quadratic cost matrices
$\pi = \{\mu_0, \mu_1, ..., \mu_{N-1}\}$	Control policy
$J(\cdot)$	Accumulated cost or cost-to-go
γ	Discount factor
$\bar{s}$, s, S	Cost-to-go quadratic approximation coefficients
K	Feedback gain
Q_c	Controllability matrix
Q_o	Observability matrix
$p(x' \mid x, u)$	Transition probability
$r(x, u, x')$, $r(x, u)$	Expected reward
$\mathbb{B}$	Bellman operator
$Pr(\cdot)$	Probability
$\mathcal{N}(\cdot,\cdot)$	Normal distribution
μ	Mean
σ	Standard deviation
Σ	Covariance matrix
$\oplus$, $\ominus$	Minkowski sum and difference operators
$\Omega_{\mathbf{tube}}$	Error tube
$V(\cdot)$, $Q(\cdot,\cdot)$	Value function and state-action value function
$R(\cdot,\cdot)$	Cumulative reward function
τ	Rollout trajectory
$A_\cdot(\cdot,\cdot)$	Advantage function
$D_{\mathbf{KL}}(\cdot\|\|\cdot)$	Kullback-Leibler divergence
ρ	Polyak averaging parameter
ψ	Target parameters

Optimization

$f(\cdot)$	Objective function
$h_i(\cdot)$	Equality constraint function
$g_j(\cdot)$	Inequality constraint function
N, M	Number of equality and inequality constraints
x	Optimization variable
$\mathcal{X}$	Feasible set
H	Hessian matrix
λ_i, μ_j	Lagrange multipliers
$\Lambda(\cdot, \cdot)$	Lagrangian function

Model Learning

$\mathcal{D}_f$	Dataset
$\mathcal{I}_f, \mathcal{O}_f$	Paired input and output datasets
D	Dataset size
d	Data index
θ, Θ	Model parameters
ξ	Model input
γ	Model output
$\phi(\cdot)$	Basis functions
Ξ, Γ	Input and output data matrices
Φ	Data matrix associated with the basis functions
$h_i(\cdot), l_i(\cdot)$	Hidden and linear layers of a neural network
$\eta(\cdot)$	Activation function
W_i	Weight matrix of the i-th layer of a neural network
a_i, z_i	Input and output of the i-th layer of a neural network
α	Learning rate
λ	Regularization coefficient
$\mathcal{GP}(\cdot, \cdot)$	Gaussian process
$m(\cdot), \kappa(\cdot, \cdot)$	Mean and kernel functions

Chapter 1

Introduction to Optimal Control

In this chapter, we introduce the key ingredients of a standard optimal control problem. Upon formulating the optimal control problem, we then present the principle of optimality and the dynamic programming algorithm (DPA), which together form the foundation for deriving optimal control policies. We conclude the chapter with examples illustrating how these concepts can be used to solve robot decision-making problems. The foundational concepts introduced here closely follow the standard framework established in [2]; the recommended additional reading material for this chapter is Sections 1.1–1.4 of [2].

1.1 Problem Statement

Problem of Interest: How to make multiple successive decisions, N stages, to minimize a cost which captures undesirable outcomes?

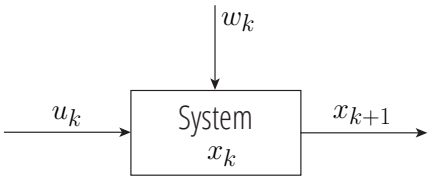
Key Ingredients

1) **Dynamics:** The underlying dynamic of the system is modelled as

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad \forall k \in \{0, 1, ..., N-1\}, \tag{1.1}$$

where k is the discrete-time index or stage, $x_k \in \mathcal{X}_k$ is the state of the system with $\mathcal{X}_k$ denoting the state space, $u_k \in \mathcal{U}_k(x_k)$ is the control or decision variable with $\mathcal{U}_k(x_k)$ denoting a set of state-dependent input constraints (e.g., the reducing set of inputs as a robot approaches a wall), $w_k \in \mathcal{D}_k$ is a random parameter representing disturbance or noise and follows a

probability distribution $Pr_k(\cdot \mid x_k, u_k)$, N is a given time horizon, and f_k is the dynamics model capturing the evolution of the system state over time.



Remark 1.1 (System State)**.** The state of a system is intuitively the set of quantities summarizing past information that is relevant for future prediction or optimization.

2) **Cost:** The cost that specifies the task has the following *scalar-valued additive* form:

$$\underbrace{\ell_N(x_N)}_{\text{terminal cost}} + \underbrace{\sum_{k=0}^{N-1} \underbrace{\ell_k(x_k, u_k, w_k)}_{\text{stage cost}}}_{\text{accumulation}}, \tag{1.2}$$

where ℓ_N is the terminal cost incurred at the end of the time horizon and ℓ_k is the stage cost incurred at each time step k.

Remark 1.2 (Minimizing Cost via Choices of Input)**.** The cost is directly or indirectly a function of the control input u_k, which is what we have access to for minimizing the cost.

Remark 1.3 (Expected Cost)**.** Since w_k is a random variable, we often consider the *expected cost*:

$$\mathbb{E}_{w_k} \left\{ \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k) \right\}, \tag{1.3}$$

where $\mathbb{E}_{w_k}$ denotes the expectation over the probability distribution of w_k. Note that minimizing the expected cost can be a big assumption in practice as this ignores the full distribution of the cost and may result in performance with a high variance.

3) **Policy:** The inputs u_k are generated by an *admissible policy*

$$\pi = \{\mu_0(x_0), \mu_1(x_1), ..., \mu_{N-1}(x_{N-1})\} \tag{1.4}$$

with $u_k = \mu_k(x_k) \in \mathcal{U}_k(x_k)$ for all $x_k \in \mathcal{X}_k$. Note that, a policy is said to be *admissible* if the input u_k is an element of the allowable set $\mathcal{U}_k(x_k)$ for all k.

For convenience, we denote Π as the set containing all possible admissible policies.

Given the above ingredients, our goal is to find an optimal policy and the associated optimal cost under stochastic uncertainty. Suppose the system starts at an initial state $x_0 \in \mathcal{X}_0$. The expected closed-loop cost associated with a policy $\pi \in \Pi$ is

$$J_\pi(x_0) = \mathbb{E}_{w_k}\left\{\ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k)\right\} \quad \text{with} \quad u_k = \mu_k(x_k). \tag{1.5}$$

Definition 1.1 (Optimal Policy and Optimal Cost)**.** Given an initial state $x_0 \in \mathcal{X}_0$, an optimal policy $\pi^* \in \Pi$ is one that satisfies the following condition:

$$J_{\pi^*}(x_0) \leq J_\pi(x_0), \quad \forall \pi \in \Pi. \tag{1.6}$$

The optimal cost of the problem for the given $x_0 \in \mathcal{X}_0$ is $J^*(x_0) = J_{\pi^*}(x_0)$. In general, $J^*(x_0) = \inf_{\pi\in\Pi} J_\pi(x_0)$, where inf denotes the infimum (or, the greatest lower bound) of the cost function.

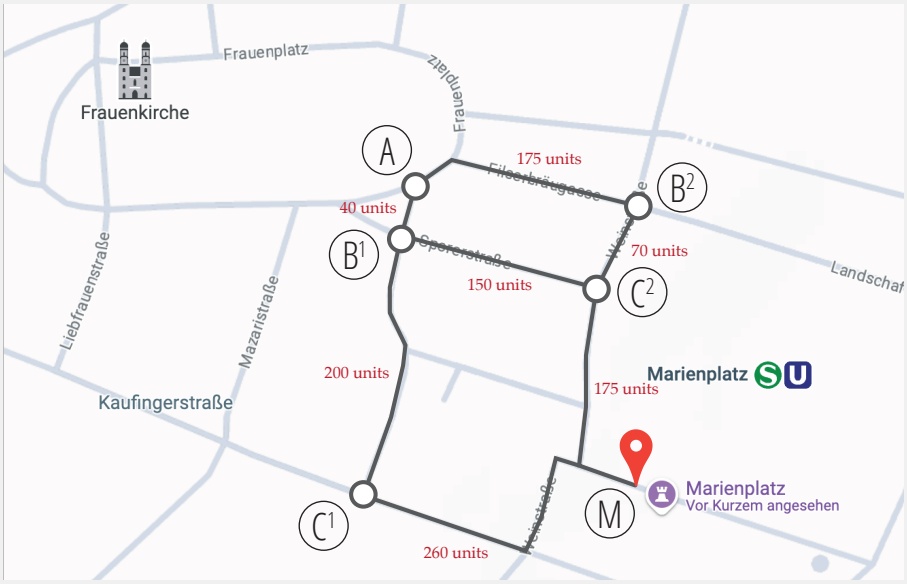
Open-Loop vs. Closed-Loop Control: We distinguish two general types of control methodologies:

- **Open-loop:** We determine a sequence of control inputs $\{u_0, u_1, ..., u_{N-1}\}$ at once before $k = 0$.
- **Closed-loop:** Our goal is to design a *control law* or *policy* $\pi = \{\mu_0, \mu_1, ..., \mu_{N-1}\}$, and the sequence of control inputs are calculated as $u_k = \mu_k(x_k)$. In contrast to open-loop control, closed-loop methods determine the sequence of control inputs in a "just-in-time fashion" (i.e., wait until time k to act) based on the measured state x_k, assuming we can measure x_k.

Remark 1.4 (Performance of Open-Loop and Closed-Loop Control)**.** The performance of closed-loop methodologies is equal to or better than open-loop methodologies, especially if there is uncertainty in the system. This is due to the fact that we are making decisions based on more information (i.e., the feedback of x_k).

Example 1.1 (City Navigation Problem)**.** We are currently at location A in the downtown area (see the map below) and would like to go to Marienplatz (red pin).

To get there, we need to make two decisions in sequence: first, choose between intersections B^1 and B^2; then, choose between C^1 and C^2. The available paths connecting these intersections are shown as solid grey lines, with the corresponding distances labeled in red. Our goal is to find the shortest path from A to the destination M.

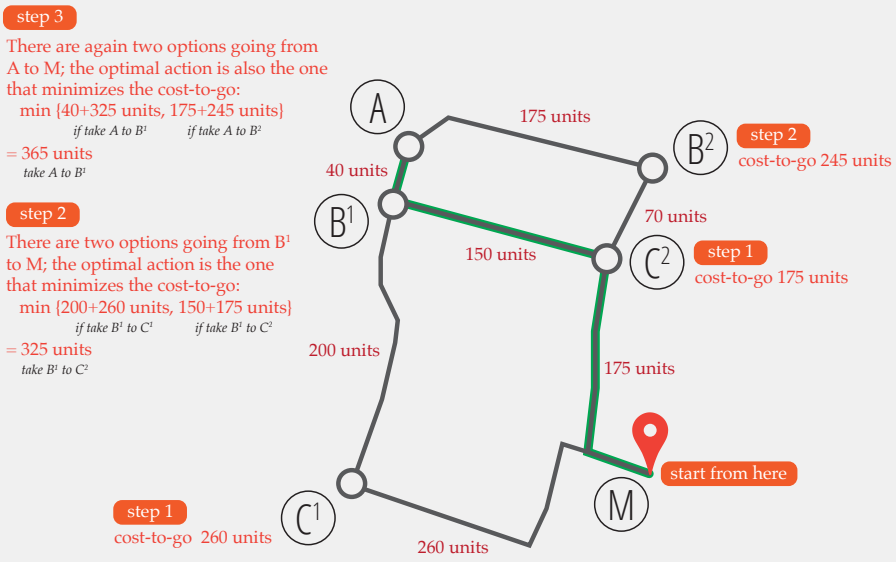


The problem can be formulated as a discrete-time optimal control problem over three stages $k \in \{0, 1, 2\}$. The state space, input space, system dynamics, and stage cost at each stage are summarized in the table below. The terminal cost is defined as $\ell_3(M) = 0$, where M denotes the destination, Marienplatz.

Stage k	**State Space** $\mathcal{X}_k$	**Input Space** $\mathcal{U}_k(x_k)$	**Dynamics** $x_{k+1} = f_k(x_k, u_k)$	**Stage Cost** $\ell_k(x_k, u_k)$
0	A	$A \to B^1$	B^1	40 units
		$A \to B^2$	B^2	175 units
1	B^1	$B^1 \to C^1$	C^1	200 units
		$B^1 \to C^2$	C^2	150 units
	B^2	$B^2 \to C^2$	C^2	70 units
2	C^1	$C^1 \to M$	M	260 units
	C^2	$C^2 \to M$	M	175 units

As we more formally introduce later in this chapter, one possible approach to solving such an optimal control problem is dynamic programming. Dynamic

programming starts from the goal and propagates backward to find the optimal solution at each stage. Intuitively, if the target is known, it is straightforward to determine the immediate optimal feedback policy leading to the target, then the previous step, and so on, working backward until the start is reached. The annotated figure below sketches the solution following the dynamic programming strategy. The optimal path is marked in green. Note that any truncated portion of the optimal path also corresponds to an optimal solution to the associated subproblem. For example, if we are at B^1, then the optimal path to the destination must be B^1C^2M as opposed to B^1C^1M; otherwise, we would have preferred this alternative when starting from A.



In this problem, we considered a deterministic setting. We can also consider more interesting cases where the stage costs and/or dynamics are stochastic (e.g., due to road closures or unplanned detours). In these stochastic settings, we need to account for the distribution of the total cost. One common approach is to minimize the expected cost as formulated in (1.5). In practice, one may also wish to account for extreme scenarios, for example, by using a risk measure [3].

1.2 The Dynamic Programming Algorithm

As hinted at in the city navigation problem above, we can use dynamic programming to solve an optimal control problem. We will formalize this idea

in this section. We will begin by introducing the *principle of optimality*, which is the basis of the dynamic programming algorithm (DPA).

Key Concept 1.1 (Principle of Optimality [2])**.** Suppose we have an optimal policy $\pi^* = \{\mu_0^*, \mu_1^*, ..., \mu_{N-1}^*\}$ that minimizes the expected cost $\mathbb{E}_{w_k}\left\{\ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k)\right\}$ subject to the system dynamics and constraints. If, while following this policy π^*, we reach a specific state x_i at time i, then the remaining part of the policy $\{\mu_i^*, \mu_{i+1}^*, ..., \mu_{N-1}^*\}$ must be optimal for the subproblem with the truncated cost $\mathbb{E}_{w_k}\left\{\ell_N(x_N) + \sum_{k=i}^{N-1} \ell_k(x_k, u_k, w_k)\right\}$.

Intuitively, the principle of optimality says that if a path ABCDE is optimal, then any truncated path from an intermediate position to E (i.e., BCDE, CDE, and DE) is optimal (Figure 1.1). The principle of optimality can be shown by contradiction. As a proof sketch, one can argue that if the truncated policy $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ were not optimal for the subproblem, then one could choose an alternative policy to further reduce the cost for the original optimal control problem. This leads to a contradiction that the original solution is optimal.

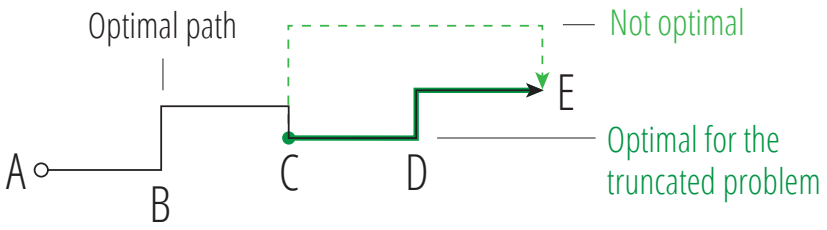


Figure 1.1: Illustration of the principle of optimality.

The principle of optimality motivates us to find an optimal policy as a backward search (i.e., start with the end and determine the optimal input recursively from $N-1$ to 0). This idea is formalized in the DPA, which computes the optimal cost and the associated policy by first constructing the optimal solution at the last stage and then working backward recursively until the initial stage is reached. We present the DPA below, and a proof of the algorithm be found in [2].

Theorem 1.1 (Dynamic Programming Algorithm [2])**.** Consider an initial state $x_0 \in \mathcal{X}_0$. The optimal cost $J^*(x_0)$ and an optimal policy π^* can be computed using the DPA, which proceeds backward in time:

- **Initialization**

$$J_N(x_N) = \ell_N(x_N), \quad \forall x_N \in \mathcal{X}_N, \tag{1.7}$$

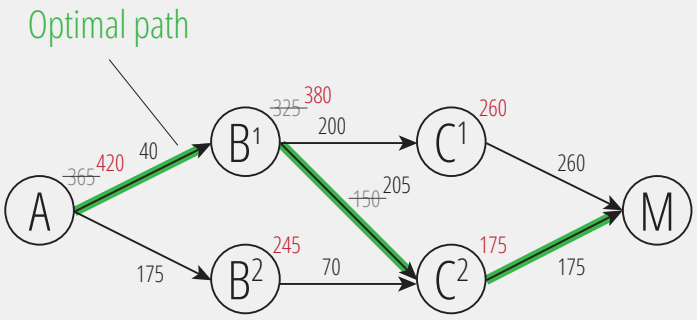
- **Recursion** (going from $N-1$ to 0)

$$J_k(x_k) = \min_{u_k \in \mathcal{U}_k(x_k)} \mathbb{E}_{w_k} \Big\{ \underbrace{\ell_k(x_k, u_k, w_k)}_{\text{stage cost}} + \underbrace{J_{k+1}\big(\underbrace{f_k(x_k, u_k, w_k)}_{x_{k+1}}\big)}_{\text{cost-to-go}} \Big\}, \forall\, x_k \in \mathcal{X}_k. \tag{1.8}$$

The optimal cost is $J^*(x_0) = J_0(x_0)$, given by the backward recursion. The optimal policy $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ consists of control laws $u_k^* = \mu_k^*(x_k)$ that minimize the recursive equation across all states and stages.

The key implication of the principle of optimality and consequently the DPA is that we need to look all the way to the end to know what is optimal. We demonstrate this idea with a simple, shortest-path problem.

Example 1.2 (Shortest or "Cheapest" Path Problem)**.** Consider again the problem in Example 1.1. Our goal is to find the shortest or "cheapest" path from A to M. We can alternatively visualize this problem as a directed graph, where each possible state is represented as a node, and possible paths with their corresponding stage costs are denoted at the edges.



As before, to find the optimal policy for each state, we apply the DPA and compute the cost-to-go values by propagating backward from the final node M. These values are shown in red near each node. The action that achieves this

minimum is the optimal action from the current state. The resulting optimal path AB^1C^2M is marked in green.

Now, suppose the cost from B^1 to C^2 has increased from 150 to 205 due to construction on Sporerstraße. What is the optimal path from A to M after this modification? If we are at B^1 and greedily choose to transition to C^1 (which has a lower stage cost compared to transitioning to C^2), we would end up with a suboptimal solution. In this case, we are not accounting for the fact that the cost-to-go from C^1 is higher than that from C^2. Thus, to determine the optimal solution, we would need to look all the way to the end. In this case, AB^1C^2M remains to be an optimal solution.

Furthermore, it should be noted that, with this modification, there exists another possible optimal solution, AB^2C^2M, with the same optimal cost. In general, while the optimal cost is unique, optimal solutions may not always be unique.

Remark 1.5 (Connection to Forward Search)**.** The DPA allows us to systematically solve the shortest path problem by computing the cost-to-go backward for every possible state at each stage. Practically, if we have a fixed initial state, we may alternatively find the optimal path through a forward search. This is the fundamental idea behind Dijkstra's algorithm, where we iteratively expand from the initial state and compute the cost from the known initial state (i.e., the cost-to-come) of the neighbouring states until the goal is reached. This is effectively a forward version of the DPA, where we swap the roles of the goal and the initial state. Variants such as the A^* algorithm further accelerate this forward search by evaluating nodes based on the sum of the exact cost-to-come and a lower-bound heuristic estimate of the cost-to-go.

The DPA can also be applied to problems with continuous state and action spaces, and to problems with constrained or discrete action spaces. To illustrate this, we consider a simple mobile robot goal-reaching task.

Example 1.3 (Mobile Robot Goal Reaching)**.** A ground robot is moving along a straight line in a warehouse to scan item labels.

Let $x_0 \in \mathbb{R}$ be its initial position and $u_k \in \mathbb{R}$ be the linear velocity command (i.e., the displacement over one time step). The system evolves over two time steps according to the following dynamic model:

$$x_{k+1} = x_k + u_k, \quad \forall k \in \{0, 1\}. \tag{1.9}$$

Our goal is to determine the control inputs u_0 and u_1 that minimize the following cost function:

$$u_0^2 + u_1^2 + q(x_2 - x_g)^2, \tag{1.10}$$

where $x_g \in \mathbb{R}$ is the goal position, and $q \in \mathbb{R}_+$ is a parameter that trades off the objective of minimizing control effort and the objective of minimizing error in reaching the goal position.

We will solve this problem with the DPA.

- **Initialization**

$$J_2(x_2) = q(x_2 - x_g)^2 \tag{1.11}$$

- **Recursion**

▷ *Step* $k = 1$

$$J_1(x_1) = \min_{u_1} \Big(\underbrace{u_1^2}_{\text{stage cost}} + \underbrace{J_2(x_2)}_{\text{cost-to-go}} \Big) \tag{1.12}$$

$$= \min_{u_1} \left(u_1^2 + J_2(x_1 + u_1) \right) \text{ (substitute in the dynamics)} \tag{1.13}$$

$$= \min_{u_1} \left(u_1^2 + q\left(x_1 + u_1 - x_g\right)^2 \right) \tag{1.14}$$

Solve for optimal u_1 by differentiating the cost and setting it to zero:

$$u_1 = \mu_1^*(x_1) = -\frac{q}{1+q}(x_1 - x_g) \tag{1.15}$$

Substitute u_1 back to $J_1(x_1)$:

$$J_1(x_1) = \frac{q}{1+q}(x_1 - x_g)^2 \tag{1.16}$$

▷ *Step* $k = 0$

$$J_0(x_0) = \min_{u_0} \left(u_0^2 + J_1(x_1)\right) \tag{1.17}$$

$$= \min_{u_0} \left(u_0^2 + J_1(x_0 + u_0)\right) \text{ (substitute in the dynamics)} \tag{1.18}$$

$$= \min_{u_0} \left(u_0^2 + \frac{q}{1+q}(x_0 + u_0 - x_g)^2\right) \tag{1.19}$$

Similarly, solve for optimal u_0 by differentiating the cost and setting it to zero:

$$u_0 = \mu_0^*(x_0) = -\frac{q}{1+2q}(x_0 - x_g) \tag{1.20}$$

Substitute u_0 back to $J_0(x_0)$:

$$J_0(x_0) = \frac{q}{1+2q}(x_0 - x_g)^2 \tag{1.21}$$

Note that the optimal policies μ_0^* and μ_1^* are *feedback* laws—they map the current state x_k to an action u_k. In the extreme case where $q \to \infty$, we have $\mu_0(x_0)^* = -(x_0 - x_g)/2$ and $\mu_1^*(x_1) = -(x_1 - x_g)$, which drive the robot exactly to the target position in two steps. On the other hand, if $q = 0$, we do not penalize any deviation from the goal state, and the optimal policies are minimal-effort solutions with $\mu_0(x_0)^* = \mu_1^*(x_1) = 0$, which keep the robot stationary at x_0.

Remark 1.6 (Certainty Equivalence)**.** In the example above, we considered the dynamics of the system to be deterministic. If we add zero-mean random noise to the dynamics, the optimal policy would be unaffected. To see this, let the dynamics of the system be

$$x_{k+1} = x_k + u_k + w_k, \quad \forall k \in \{0, 1\}, \tag{1.22}$$

where w_1 and w_2 are random variables with zero mean and finite variance. We can

rewrite the cost at stage 1 as

$$J(x_1) = \min_{u_1} \mathbb{E}_{w_1} \left\{ u_1^2 + q\big(x_1 + u_1 + w_1 - x_g\big)^2 \right\} \tag{1.23}$$

$$= \min_{u_1} \Big(u_1^2 + q\big(x_1 + u_1 - x_g\big)^2 + 2q\cancel{\mathbb{E}_{w_1}\{w_1\}}\big(x_1 + u_1 - x_g\big) + \underbrace{q\mathbb{E}_{w_1}\{w_1^2\}}_{\text{constant w.r.t. } u_1} \Big). \tag{1.24}$$

The minimization problem is equivalent to

$$J(x_1) = \min_{u_1} \Big(u_1^2 + q\big(x_1 + u_1 - x_g\big)^2 \Big), \tag{1.25}$$

which is identical to the case without additive noise. This property is called *certainty equivalence* and holds for several types of linear quadratic problems [2].

Definition 1.2 (Certainty Equivalence)**.** Certainty equivalence is said to hold if the optimal policy for a stochastic problem is identical to the policy obtained when the random variables are replaced by their expected values:

$$\begin{aligned} &\arg\min_{\pi \in \Pi} \quad \mathbb{E}_{w_k} \left\{ \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k) \right\} \\ &= \arg\min_{\pi \in \Pi} \quad \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, \mathbb{E}_{w_k}\{w_k\}). \end{aligned} \tag{1.26}$$

Example 1.4 (Mobile Robot Goal Reaching with Bounded Inputs)**.** Suppose now that, in the mobile robot example, the input is bounded within the set $\mathcal{U}_k(x_k) = \mathcal{U} = \{u_k \in \mathbb{R} \mid -1 \leq u_k \leq 1\}$ for $k \in \{0, 1\}$ and $x \in \mathbb{R}$. Find the optimal policy for the first two time steps.

We will again solve the problem with the DPA.

- **Initialization**

$$J_2(x_2) = q(x_2 - x_g)^2 \tag{1.27}$$

- **Recursion**

▷ *Step* $k = 1$

$$J_1(x_1) = \min_{-1 \leq u_1 \leq 1} \quad (1+q)u_1^2 + 2q(\underbrace{x_1 - x_g}_{\delta x_1})u_1 + q(\underbrace{x_1 - x_g}_{\delta x_1})^2, \tag{1.28}$$

Given the input constraints, we have three possible cases for $J_1(x_1)$ depending on the value of the state error $\delta x_1 = x_1 - x_g$:

$$u_1 = \mu^*(x_1) = \begin{cases} 1 & \text{if} \quad \delta x_1 < -\dfrac{1+q}{q} \\ -\dfrac{q}{1+q}\delta x_1 & \text{if} \quad -\dfrac{1+q}{q} \le \delta x_1 \le \dfrac{1+q}{q} \\ -1 & \text{otherwise.} \end{cases} \tag{1.29}$$

The policy in the second case is identical to the unconstrained solution from Example 1.3, while the other two cases clamp the input to its constrained bounds. Note that the boundaries of the different cases depend on the value of q. When q is smaller, we care less about the state error and thus we less likely saturate the input; as a result, the interval of δx_1 corresponding to the second case (i.e., the range of δx_1 for which the input constraints are inactive) becomes wider. The corresponding optimal cost is

$$J_1(x_1) = \begin{cases} (1+q) + 2q\,\delta x_1 + q\,\delta x_1^2 & \text{if } \delta x_1 < -\dfrac{1+q}{q} \\ \frac{q}{1+q}\delta x_1^2 & \text{if } -\dfrac{1+q}{q} \le \delta x_1 \le \dfrac{1+q}{q} \\ (1+q) - 2q\,\delta x_1 + q\,\delta x_1^2 & \text{otherwise.} \end{cases} \tag{1.30}$$

▷ *Step* $k = 0$

$$J_0(x_0) = \min_{-1 \le u_0 \le 1} \quad u_0^2 + J_1(x_0 + u_0), \tag{1.31}$$

where J_1 is defined in (1.30). For each of the J_1 cases in (1.30), we have three cases depending on the value of the state error $\delta x_0 = x_0 - x_g$, though not all possible cases will arise. After some calculations (see Table 1.1), one can show that we arrive at the following feasible cases:

$$u_0 = \mu^*(x_0) = \begin{cases} 1 & \text{if } \delta x_0 < -\frac{1+2q}{q} \\ -\frac{q}{1+2q}\delta x_0 & \text{if } -\frac{1+2q}{q} \le \delta x_0 \le \frac{1+2q}{q} \\ -1 & \text{otherwise.} \end{cases} \tag{1.32}$$

The corresponding $J_0(x_0)$ is

$$J_0(x_0) = \begin{cases} q\,\delta x_0^2 + 4q\,\delta x_0 + 4q + 2 & \text{if } \delta x_0 < -\dfrac{1+2q}{q} \\ \dfrac{q}{1+2q}\delta x_0^2 & \text{if } -\dfrac{1+2q}{q} \leq \delta x_0 \leq \dfrac{1+2q}{q} \\ q\,\delta x_0^2 - 4q\,\delta x_0 + 4q + 2 & \text{otherwise.} \end{cases} \tag{1.33}$$

Example 1.5 (Mobile Robot Goal Reaching with Discrete Inputs)**.** Consider again the problem in Example 1.3. We now restrict the inputs to two possible values $\mathcal{U}_k(x_k) = \mathcal{U} = \{-1, 1\}$ for $k \in \{0, 1\}$ and $x \in \mathbb{R}$. Our goal is to find the optimal policy for the first two time steps.

We will solve this problem again with the DPA but subject to the given input constraints.

- **Initialization**

$$J_2(x_2) = q(x_2 - x_g)^2 \tag{1.34}$$

- **Recursion**

▷ *Step* $k = 1$

$$J_1(x_1) = \min_{u_1 \in \mathcal{U}} \left(u_1^2 + J_2(x_2)\right) \tag{1.35}$$

$$= \min_{u_1 \in \mathcal{U}} \left(u_1^2 + J_2(x_1 + u_1)\right) \text{ (substitute in the dynamics)} \tag{1.36}$$

$$= \min_{u_1 \in \mathcal{U}} \left(u_1^2 + q\big(x_1 + u_1 - x_g\big)^2\right) \tag{1.37}$$

Since the input space is constrained to two values, we have the following two possible values for $J_1(x_1)$:

$$J_1(x_1) = \begin{cases} q(x_1 - x_g)^2 + (1+q) - 2q(x_1 - x_g) & \text{if } u_1 = -1 \\ q(x_1 - x_g)^2 + (1+q) + 2q(x_1 - x_g) & \text{if } u_1 = 1 \end{cases} \tag{1.38}$$

At each state $x_1 \in \mathbb{R}$, the optimal input u_1 is the value $u_1 \in \{-1, 1\}$ that minimizes $J_1(x_1)$. If we inspect (1.38) closely, the two cases are equal when the state error $\delta x_1 = x_1 - x_g$ is zero. It is not hard to verify that the optimal

policy has the following two cases:

$$u_1 = \mu_1^*(x_1) = \begin{cases} 1 & \text{if } \delta x_1 \leq 0 \\ -1 & \text{if } \delta x_1 > 0 \end{cases} \tag{1.39}$$

$$= -\text{sgn}(\delta x_1), \tag{1.40}$$

where $\text{sgn}(\cdot)$ is the sign function that has a value of 1 if its argument is positive, -1 if negative, and 0 if zero. Intuitively, this feedback law (1.39) commands the input velocity to act in the opposite direction of the current position error δx_1, steering the system towards the goal. In contrast to the unconstrained input case in Example 1.3, with the discrete input case, we have a policy that takes on only two values.

Substitute u_1 back to $J_1(x_1)$:

$$J_1(x_1) = \begin{cases} q\,\delta x_1^2 + (1+q) + 2q\,\delta x_1 & \text{if } \delta x_1 \leq 0 \\ q\,\delta x_1^2 + (1+q) - 2q\,\delta x_1 & \text{if } \delta x_1 > 0 \end{cases} \tag{1.41}$$

We can also rewrite it in a more compact form:

$$J_1(x_1) = q\,\delta x_1^2 + (1+q) - 2q\,\text{sgn}(\delta x_1)\,\delta x_1, \tag{1.42}$$

$$= q\,\delta x_1^2 + (1+q) - 2q\,|\delta x_1|. \tag{1.43}$$

Note that, in the above derivation, we make use of the identity $\text{sgn}(z)\,z = |z|$ for some $z \in \mathbb{R}$.

▷ *Step* $k = 0$

$$J_0(x_0) = \min_{u_0 \in \mathcal{U}} \left(u_0^2 + J_1(x_1)\right) \tag{1.44}$$

$$= \min_{u_0 \in \mathcal{U}} \left(u_0^2 + J_1(x_0 + u_0)\right) \text{ (substitute in the dynamics)} \tag{1.45}$$

$$= \min_{u_0 \in \mathcal{U}} \left(u_0^2 + q(x_0 + u_0 - x_g)^2 + (1+q) - 2q\,|x_0 + u_0 - x_g|\right) \tag{1.46}$$

Similarly, given the discrete input set $\mathcal{U}$, we have the following two possible

values for $J_0(x_0)$:

$$J_0(x_0) = \begin{cases} q(x_0 - x_g)^2 + 2(1+q) + 2q(x_0 - x_g) - 2q\,|x_0 + 1 - x_g| & \text{if } u_0 = 1 \\ q(x_0 - x_g)^2 + 2(1+q) - 2q(x_0 - x_g) - 2q\,|x_0 - 1 - x_g| & \text{if } u_0 = -1 \end{cases} \tag{1.47}$$

Note that the two expressions for $J_0(x_0)$ differ only in the last two terms. For convenience, we define

$$c_a(x_0 - x_g) = 2q(x_0 - x_g) - 2q\,|x_0 + 1 - x_g|, \text{ and} \tag{1.48}$$

$$c_b(x_0 - x_g) = -2q(x_0 - x_g) - 2q\,|x_0 - 1 - x_g|. \tag{1.49}$$

There are three possible cases based on the value of the error term $\delta x_0 = x_0 - x_g$:

$$\begin{cases} c_a\,\delta x_0 < c_b\,\delta x_0 & \text{if } \delta x_0 \le -1 \\ c_b\,\delta x_0 = c_a\,\delta x_0 & \text{if } -1 < \delta x_0 < 1 \\ c_a\,\delta x_0 > c_b\,\delta x_0 & \text{if } \delta x_0 \ge 1. \end{cases} \tag{1.50}$$

Based on this observation, we have the following optimal policy:

$$u_0 = \mu_0^*(x_0) = \begin{cases} 1 & \text{if } \delta x_0 \le -1 \\ 1 \text{ or } -1 & \text{if } -1 < \delta x_0 < 1 \\ -1 & \text{if } \delta x_0 \ge 1, \end{cases} \tag{1.51}$$

which can be written as

$$u_0 = \mu_0^*(x_0) = \begin{cases} 1 \text{ or } -1 & \text{if } -1 < \delta x_0 < 1 \\ -\text{sgn}(\delta x_0) & \text{otherwise.} \end{cases} \tag{1.52}$$

Note that, for the case where $-1 < \delta x_0 < 1$, we have $c_a\,\delta x_0 = c_b\,\delta x_0$, which implies that choosing either input will lead to the same cost.

Substitute u_0 back to $J_0(x_0)$:

$$J_0(x_0) = \begin{cases} q\,\delta x_0^2 + 2(1+q) + 2q\,\delta x_0 - 2q\,|\delta x_0 + 1| & \text{if } \delta x_0 \le -1 \\ q\,\delta x_0^2 + 2(1+q) - 2q & \text{if } -1 < \delta x_0 < 1 \\ q\,\delta x_0^2 + 2(1+q) - 2q\,\delta x_0 - 2q\,|\delta x_0 - 1| & \text{if } \delta x_0 \ge 1. \end{cases} \tag{1.53}$$

The expression of $J_0(x_0)$ can be further simplified to

$$J_0(x_0) = \begin{cases} q\,\delta x_0^2 + 2(1+q) - 2q & \text{if } -1 < \delta x_0 < 1 \\ q\,\delta x_0^2 + 2(1+q) - 2q|\delta x_0| - 2q\,|\delta x_0 - 1| & \text{otherwise.} \end{cases} \tag{1.54}$$

As shown in the simple mobile robot examples above, DPA can be applied to optimal control problems with input constraints; however, in more general settings, solving a finite-horizon optimal control problem with input constraints is not necessarily trivial. To efficiently address such problems in practice, a receding-horizon approach (i.e., model predictive control) is often used. We will cover this topic in a later chapter.

</> See Jupyter notebook 1.1 for example implementation of the DPA.

Chapter Summary

Basic problem:

- Dynamics

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad \forall k \in \{0, 1, ..., N-1\} \tag{1.55}$$

$$x_k \in \mathcal{X}_k, \quad u_k \in \mathcal{U}(x_k) \subset \mathcal{C}_k, \quad w_k \in \mathcal{D}_k, \tag{1.56}$$

$$w_k \sim Pr_k(\cdot \mid x_k, u_k) \tag{1.57}$$

- Cost

$$J_\pi(x_0) = \mathbb{E}_{w_k}\left\{\ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k)\right\} \tag{1.58}$$

Principle of optimality: Consider an optimal policy $\pi^* = \{\mu_0^*(x_0), \mu_1^*(x_1), ..., \mu_{N-1}^*(x_{N-1})\}$. The truncated optimal policy $\{\mu_i^*(x_i), \mu_{i+1}^*(x_{i+1}), ..., \mu_{N-1}^*(x_{N-1})\}$ is optimal for any resulting state x_i at time i.

DPA: We solve the problem in a backward manner (proof in [2, Sec. 1.3]). For any initial state, $x_0 \in \mathcal{X}_0$, the optimal cost $J^*(x_0)$ is equal to $J_0(x_0)$ given by a

recursive algorithm:

- Initialization $J_N(x_N) = \ell_N(x_N)$ for all $x_N \in \mathcal{X}_N$
- Recursion $J_k(x_k) = \min_{u_k \in \mathcal{U}(x_k)} \mathbb{E}_{w_k} \left\{ \ell_k(x_k, u_k, w_k) + J_{k+1}\big(f_k(x_k, u_k, w_k)\big) \right\}$
- Find $u_k^* = \mu_k^*(x_k)$

Table 1.1: A summary of the nine possible cases for stage $k = 0$. The first column corresponds to the three possible regions of the cost-to-go function J_1 defined in (1.30). The second and third columns show the bounds on x_0 derived from the corresponding interval of x_1 and from the input constraints on u_0, respectively. The fourth column indicates whether the two bounds define a feasible (i.e., non-empty) range. The final column provides the optimal input u_0 for each case.

Region of J_1	**Bound on x_0 from x_1 Region (with u_0)**	**Bound x_0 from Input Constraint $u_0 \in [-1, 1]$**	**Feasible Case?**	**Expression of u_0**
Upper	$x_0 - x_g + (-1) > \frac{1+q}{q} \Rightarrow x_0 - x_g > \frac{1+2q}{q}$	$x_0 - x_g > \frac{1+2q}{q}$	Yes	$u_0 = -1$
Upper	$x_0 - x_g - \frac{q}{1+q}(x_0 - x_g - 1) > \frac{1+q}{q} \Rightarrow x_0 - x_g > \frac{1+2q}{q}$	$x_0 - x_g \in \left[-\frac{1}{q}, \frac{1+2q}{q}\right]$	No	$u_0 = -\frac{q}{1+q}(x_0 - x_g - 1)$
Upper	$x_0 - x_g + 1 > \frac{1+q}{q} \Rightarrow x_0 - x_g > \frac{1}{q}$	$x_0 - x_g < -\frac{1}{q}$	No	$u_0 = 1$
Inner	$x_0 - x_g + (-1) \in \left[-\frac{1+q}{q}, \frac{1+q}{q}\right] \Rightarrow x_0 - x_g \in \left[-\frac{1}{q}, \frac{1+2q}{q}\right]$	$x_0 - x_g > \frac{1+2q}{q}$	No	$u_0 = -1$
Inner	$x_0 - x_g - \frac{q}{1+2q}(x_0 - x_g) \in \left[-\frac{1+q}{q}, \frac{1+q}{q}\right] \Rightarrow x_0 - x_g \in \left[-\frac{1+2q}{q}, \frac{1+2q}{q}\right]$	$x_0 - x_g \in \left[-\frac{1+2q}{q}, \frac{1+2q}{q}\right]$	Yes	$u_0 = -\frac{q}{1+2q}(x_0 - x_g)$
Inner	$x_0 - x_g + 1 \in \left[-\frac{1+q}{q}, \frac{1+q}{q}\right] \Rightarrow x_0 - x_g \in \left[-\frac{1+2q}{q}, \frac{1}{q}\right]$	$x_0 - x_g < -\frac{1+2q}{q}$	No	$u_0 = 1$
Lower	$x_0 - x_g + (-1) < -\frac{1+q}{q} \Rightarrow x_0 - x_g < -\frac{1}{q}$	$x_0 - x_g > \frac{1}{q}$	No	$u_0 = -1$
Lower	$x_0 - x_g - \frac{q}{1+q}(x_0 - x_g + 1) < -\frac{1+q}{q} \Rightarrow x_0 - x_g < -\frac{1+2q}{q}$	$x_0 - x_g \in \left[-\frac{1+2q}{q}, \frac{1}{q}\right]$	No	$u_0 = -\frac{q}{1+q}(x_0 - x_g + 1)$
Lower	$x_0 - x_g + 1 < -\frac{1+q}{q} \Rightarrow x_0 - x_g < -\frac{1+2q}{q}$	$x_0 - x_g < -\frac{1+2q}{q}$	Yes	$u_0 = 1$

1.3 Exercises

Exercise 1.1 (Short Questions)**.** For the problems below, select the option(s) that best answer each question and justify your selection(s).

a) In the formulation of an optimal control problem, what is the purpose of the stage cost $\ell_k(x_k, u_k, w_k)$ and terminal cost $\ell_N(x_N)$?

   (A) To model the evolution of the system dynamics over time.

   (B) To capture the desirable outcomes or penalties incurred at every stage.

   (C) To specify constraints on the admissible control inputs at every stage.

   (D) To define the probability distribution of random disturbances affecting the system.

b) Suppose you are programming a robot to perform a navigation task. What do you need to know to come up with an optimal strategy for executing the task?

   (A) An expression capturing desired outcomes (a cost function).

   (B) A model of the robot behaviour (system dynamics).

   (C) The robot's past experience (a history of the robot's states).

   (D) Physical constraints of the robot (input constraints).

c) Consider a system where both open-loop and closed-loop control methodologies are being evaluated for performance. The discrete system dynamics are given by $x_{k+1} = Ax_k + Bu_k$, where x_k is the state vector and u_k is the control input. The performance of the control methodologies is assessed based on the ability to reject disturbances and track a reference signal. Select all true statements.

   (A) Open-loop control relies on the system model and does not use feedback from the state vector x_k.

   (B) Open-loop control is generally more robust to disturbances and model uncertainties compared to closed-loop control.

   (C) Closed-loop control requires a precise model of the system dynamics to function effectively.

   (D) Closed-loop control can compensate for model inaccuracies and external disturbances by using feedback from the state vector x_k.

d) According to the principle of optimality, if a path ABCDE is deemed optimal, what can be inferred about the subpath CDE for the subproblem from C to E?

(A) The subpath CDE can be suboptimal for the subproblem.

(B) The subpath CDE is optimal for the subproblem.

(C) There can be another subpath other than CDE that is optimal for the subproblem.

(D) The subpath must be independently recalculated to verify optimality.

e) Which of the following statements about the principle of optimality are correct?

(A) An optimal policy for a problem remains optimal for all possible subproblems derived from it.

(B) Let $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ be a suboptimal policy. Then, the truncated policy $\{\mu_i, \mu_{i+1}, ..., \mu_{N-1}\}$ cannot be optimal for the subproblem, where we are at x_i at time i and want to minimize $\mathbb{E}_{w_k}\left\{\ell_N(x_N) + \sum_{k=i}^{N-1} \ell_k(x_k, u_k, w_k)\right\}$.

(C) To obtain the optimal policy, we need to solve for all possible trajectories simultaneously.

(D) It guarantees that open-loop control will outperform closed-loop control.

f) Which of the following statements about the DPA are correct?

(A) It provides a feedback policy.

(B) It solves the problem backwards.

(C) It also applies to nonlinear system dynamics.

(D) It only works for finite input and state spaces.

g) Which of the following statements about the certainty equivalence principle are correct?

(A) Certainty equivalence assumes that the stochastic disturbance w_k can be replaced by its expected value when designing the control policy.

(B) Certainty equivalence is guaranteed to lead to a control policy that is optimal for the actual stochastic system.

(C) Certainty equivalence holds in a linear quadratic optimal control problem when the disturbance is additive and the random variable w_k has zero mean and finite variance.

(D) If $\mathbb{E}[w_k] = 0$, the optimal policy is the same as without the disturbances.

Exercise 1.2 (Shortest Path Problem)**.** Consider the directed graph shown in

Figure 1.2. Find the shortest path from nodes A and B to node C by using the DPA. Assume that the cost to remain at any node is zero.

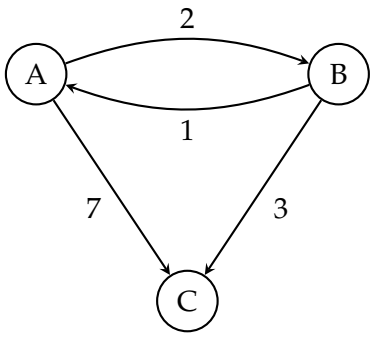


Figure 1.2: Directed graph for the shortest path problem in Exercise 1.2.

Exercise 1.3 (Shortest Path Problem)**.** Consider the directed graph shown in Figure 1.3. Find the shortest path from the start node (S) to the terminal node (T) by using the DPA. Assume that the cost to remain at any node is zero.

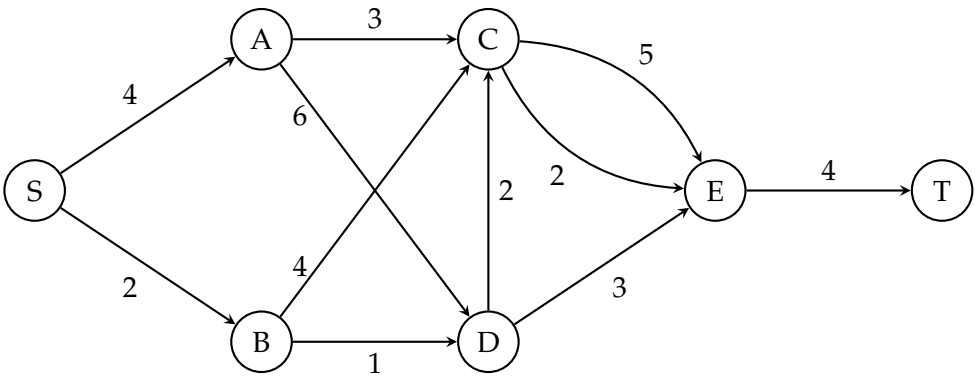


Figure 1.3: Directed graph for the shortest path problem in Exercise 1.3.

Exercise 1.4 (Mobile Robot Goal-Reaching Problem)**.** Consider again the mobile robot goal-reaching setup in Example 1.3 but with the dynamics of the robot represented as

$$x_{k+1} = x_k + u_k + w_k, \qquad k \in \{0, 1\}, \tag{1.59}$$

where k is the discrete-time index, $x_k \in \mathbb{R}$ is the position of the robot, $u_k \in \mathbb{R}$ is the normalized velocity input (i.e., the distance driven in one time step), and $w_k \in \mathbb{R}$ is a disturbance (e.g., caused by variations in the terrain). The initial position of the robot is $x_0 = -1$. Our goal is to minimize the cost function given by

$$\mathbb{E}_{w_k}\left\{x_2^2 + \sum_{k=0}^{1}(qx_k^2 + ru_k^2)\right\}, \tag{1.60}$$

where $q \geq 0$ and $r > 0$ are constant scalars. No actuation constraints are imposed on u_k.

a) Given the cost function and initial position, how do you expect the policy found using the DPA to behave? How do you expect different values of q and r to change the robot's behaviour?

b) Let $q = 5/2$ and $r = 1$ and assume that $w_k = 0$ for all k. Find the optimal policy and $J(x_0)$.

c) We keep $q = 5/2$ and $r = 1$, but now assume there is some variation in the terrain causing the disturbance to follow a distribution with mean $\mathbb{E}[w_k] = 1$ and variance $\text{Var}[w_k] = 1$ for $k = \{0, 1\}$. The disturbance could represent rocks, hills, and slippable terrain along the path that either helps or hinders the robot's motion. Find the optimal policy and $J(x_0)$. *Hint: Recall that* $\text{Var}[y] = \mathbb{E}[y^2] - \mathbb{E}[y]^2$.

d) Comment on any differences in the solutions you obtained for parts b) and c).

Exercise 1.5 (Constrained Mobile Robot Goal-Reaching Problem)**.** Consider again the mobile robot goal-reaching setup in Exercise 1.4. We now apply the DPA to find the optimal control policy with input and state constraints.

Recall that the dynamics of the robot are represented as

$$x_{k+1} = x_k + u_k + w_k, \qquad k \in \{0, 1\}, \tag{1.61}$$

where k is the discrete-time index, $x_k \in \mathbb{R}$ is the position of the robot, $u_k \in \mathbb{R}$ is the normalized velocity input, and $w_k \in \mathbb{R}$ is a disturbance. The initial position of the robot is $x_0 = -1$. For simplicity, for this problem, we assume $w_k = 0$ for all k.

Our goal is to minimize the cost function given by

$$\mathbb{E}_{w_k}\left\{ x_2^2 + \sum_{k=0}^{1} (q x_k^2 + r u_k^2) \right\}, \tag{1.62}$$

where $q = 5/2$ and $r = 1$ are constant scalars.

a) Apply the DPA to find the optimal control policy, but now assume the input u_k can only take the two values 1 and -1.

b) Assume the input u_k is again continuous and unconstrained, but the robot is operating near an obstacle such that its state is restricted to $x_k \leq -0.5$ for $k \in \{1, 2\}$. Find the new optimal policy and $J(x_0)$.

Chapter 2

Linear Quadratic Optimal Control

In the previous chapter, we introduced the basic problem of decision-making under stochastic uncertainty over a finite time horizon and the DPA that allows us to find an optimal closed-loop policy with respect to the expected cost.

In this chapter, we focus on a special class of optimal control problems with linear quadratic structure—that is, problems where the system dynamics are linear and the cost function is quadratic. In the context of stabilization problems, these are commonly referred to as linear quadratic regulator (LQR) problems. The linear quadratic structure of such problems allows us to apply the DPA more efficiently. In the infinite-horizon cases, if the necessary assumptions are satisfied, finding the optimal policy boils down to solving a matrix quadratic equation.

We will begin the chapter by deriving the optimal policy for the LQR problem in both finite- and infinite-horizon discrete-time systems, then introduce the continuous-time counterpart. We will conclude by discussing how we can obtain a linear, and possibly discrete-time, system in practice, enabling us to leverage the results presented in this chapter. The recommended additional reading materials are Section 1.7 of [4], Section 5.1 of [2], and Sections 12.3-12.4 of [5].

2.1 Finite-Horizon Discrete-Time LQR

In this section, we consider a special case where the system has linear dynamics and the cost has a quadratic form. These structural properties significantly simplify the derivation of the optimal policy. As we will elaborate, when dealing with a linear quadratic optimal control problem, the optimal policy and the cost-to-go function

can be expressed in *closed form*.

Consider a linear discrete-time system

$$x_{k+1} = A_k x_k + B_k u_k, \quad \forall k \in \{0, 1, ..., N-1\} \tag{2.1}$$

and a quadratic cost

$$J(x_0) = x_N^T Q_N x_N + \sum_{k=0}^{N-1} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right), \tag{2.2}$$

where $x_k \in \mathbb{R}^n$ is the system state, $u_k \in \mathbb{R}^m$ is the system input (without any constraints), (A_k, B_k) are the matrices parametrizing the system dynamics, Q_k and Q_N are symmetric positive semidefinite matrices penalizing the state, and R_k is a symmetric positive definite matrix penalizing the input. Our goal is to derive the optimal policy using the DPA.

To derive the recursive update, we assume that the cost-to-go at each timestep k is of the quadratic form:

$$J_k(x_k) = x_k^T S_k x_k, \quad \forall k \in \{0, 1, ..., N\} \tag{2.3}$$

with $S_k = S_k^T$ without loss of generality[1]. The matrix S_k is often referred to as the Riccati matrix.

- **Initialization**
  The terminal cost is given by

  $$J_N(x_N) = x_N^T Q_N x_N, \tag{2.4}$$

  which means

  $$S_N = Q_N. \tag{2.5}$$

- **Recursion** (going from $N-1$ to 0)
  At timestep k, we have

  $$J_k(x_k) = \min_{u_k} \left(x_k^T Q_k x_k + u_k^T R_k u_k + J_{k+1}(x_{k+1}) \right) \tag{2.6}$$

[1] A generic square matrix $M \in \mathbb{R}^{q \times q}$ can be decomposed into a symmetric component M_{sym} and a skew-symmetric component M_{skw}: $M = M_{\text{sym}} + M_{\text{skw}}$ with $M_{\text{sym}} = (M + M^T)/2$ and $M_{\text{skw}} = (M - M^T)/2$. The quadratic term $v^T M v$ with $v \in \mathbb{R}^q$ is only dependent on the symmetric component of the matrix: $v^T M v = v^T M_{\text{sym}} v$. In particular, since $v^T M v$ is a scalar, we have $v^T M v = (v^T M v)^T = v^T M^T v$. This implies that $v^T (M - M^T) v = 0$, and thus $v^T M_{\text{skw}} v = 0$. The quadratic term $v^T M v$ is then $v^T M v = v^T (M_{\text{sym}} + M_{\text{skw}}) v = v^T M_{\text{sym}} v$.

$$= \min_{u_k} \Big(x_k^T Q_k x_k + u_k^T R_k u_k + (A_k x_k + B_k u_k)^T S_{k+1} (A_k x_k + B_k u_k) \Big) \quad (2.7)$$

$$= \min_{u_k} \Big(x_k^T (Q_k + A_k^T S_{k+1} A_k) x_k + u_k^T (R_k + B_k^T S_{k+1} B_k) u_k + 2 u_k^T B_k^T S_{k+1} A_k x_k \Big). \quad (2.8)$$

Note that the optimization problem in (2.8) is quadratic in the optimization variable u_k. As before, the optimal solution can be found by taking the derivative with respect to u_k and setting the derivative to zero. The optimal solution is

$$u_k = \mu_k^*(x_k) = \underbrace{-(R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k}_{=K_k} x_k = K_k x_k, \quad (2.9)$$

where $K_k \in \mathbb{R}^{m \times n}$ is the feedback control gain. The corresponding optimal cost-to-go is

$$J_k(x_k) = x_k^T \Big(Q_k + A_k^T S_{k+1} A_k + K_k^T (R_k + B_k^T S_{k+1} B_k) K_k + 2 (A_k x_k)^T S_{k+1} B_k K_k \Big) x_k \quad (2.10)$$

$$= x_k^T \Big(\underbrace{Q_k + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k (R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k}_{=S_k} \Big) x_k. \quad (2.11)$$

The backward recursive update for the cost-to-go function can be expressed in terms of the backward recursive update of S_k as follows:

$$S_k = Q_k + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k (R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k. \quad (2.12)$$

The recursive update for S_k in (2.12) is known as the discrete-time algebraic Riccati equation (DARE) for the finite-horizon linear quadratic optimal control problems. The term S_k is calculated backwards from timestep $(N-1)$ to timestep 0, and the optimal policy for each timestep is given by

$$u_k = \mu_k^*(x_k) = \underbrace{-(R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k}_{=K_k} x_k = K_k x_k, \quad (2.13)$$

which is a function of the system matrices (A_k, B_k), the cost parameter R_k, and the variable S_{k+1}. Note that the sequence of S_k can be calculated ahead of time, and the

optimal policy is a *linear* feedback policy with gain K_k.

</> See Jupyter notebook 2.1 example 1.1 for example implementation of the finite horizon LQR.

Example 2.6 (Mobile Robot Goal Reaching with Finite-Horizon LQR)**.** Consider the mobile robot goal-reaching problem from the previous chapter. Reformulate the problem and apply the DARE to obtain the optimal feedback policy $\mu_k^*(x_k) = K_k x_k$.

Recall that the dynamics of the system are given by

$$x_{k+1} = x_k + u_k, \quad \forall k \in \{0, 1\}, \tag{2.14}$$

and the cost to be minimized is

$$u_0^2 + u_1^2 + q(x_2 - x_g)^2, \tag{2.15}$$

where $x_k \in \mathbb{R}$ is the robot position, $u_k \in \mathbb{R}$ is the velocity input, $x_g \in \mathbb{R}$ is the goal position, and $q \in \mathbb{R}_+$ is a parameter trading off the relative significance of the control effort penalization and state error minimization terms.

To solve the problem using the DARE, we first introduce the error dynamics of the system:

$$\tilde{x}_{k+1} = \tilde{x}_k + u_k, \tag{2.16}$$

where $\tilde{x}_k = x_k - x_g$. The cost function can then be expressed as

$$u_0^2 + u_1^2 + q\tilde{x}_2^2. \tag{2.17}$$

We solve for S_k and K_k by setting $A_0 = A_1 = 1$, $B_0 = B_1 = 1$, $Q_0 = Q_1 = 0$, $Q_2 = q$, and $R_0 = R_1 = 1$ in (2.12) and (2.13).

- **Initialization**

$$S_2 = Q_2 = q \tag{2.18}$$

- **Recursion**

▷ *Step* $k = 1$ The Riccati matrix and optimal gain are

$$S_1 = Q_1 + A^T S_2 A - A^T S_2 B(R_1 + B^T S_2 B)^{-1} B^T S_2 A = \frac{q}{1+q}, \text{ and} \tag{2.19}$$

$$K_1 = -(R_1 + B^T S_2 B)^{-1} B^T S_2 A = -\frac{q}{1+q}. \tag{2.20}$$

The corresponding optimal policy is

$$u_1 = \mu_1^*(\tilde{x}_1) = K_1 \tilde{x}_1 = -\frac{q}{1+q}\tilde{x}_1. \tag{2.21}$$

▷ *Step* $k = 0$ The Riccati matrix and optimal gain are

$$S_0 = Q_0 + A^T S_1 A - A^T S_1 B(R_0 + B^T S_1 B)^{-1} B^T S_1 A = \frac{q}{1+2q}, \tag{2.22}$$

$$K_0 = -\left(R_0 + B^T S_1 B\right)^{-1} B^T S_1 A = -\frac{q}{1+2q}. \tag{2.23}$$

The corresponding optimal policy is

$$u_0 = \mu_0^*(\tilde{x}_0) = K_0 \tilde{x}_0 = -\frac{q}{1+2q}\tilde{x}_0. \tag{2.24}$$

Note that the optimal policies in (2.24)-(2.21) are the same as those obtained by directly applying the DPA.

2.2 Infinite-Horizon Problems

We have so far looked at solving discrete-time, finite-horizon optimal control problems with the DPA. We now turn to problems with infinite horizon (i.e., $N \to \infty$). The discrete-time, infinite-horizon problem has the following components:

- **Dynamics**: We consider a time-invariant system represented as

$$x_{k+1} = f(x_k, u_k, w_k), \quad \forall k \in \{0, 1, ...\}, \tag{2.25}$$

where $x_k \in \mathcal{X}$ is the state, $u_k \in \mathcal{U}(x_k)$, and $w_k \sim Pr(\cdot \mid x_k, u_k)$ is a random disturbance or noise. We assume that, given x_k and u_k, the disturbance w_k is independent of all previous variables. Note that, here, both the dynamics f and the probability distribution of w_k have no explicit dependency on time.

- **Cost:** The cost function is the sum of time-invariant stage costs, which is given by

$$\sum_{k=0}^{N-1} \ell(x_k, u_k, w_k). \tag{2.26}$$

Suppose the sequence of inputs u_k is generated by an admissible policy

$$\pi = \{\mu_0(x_0), \mu_1(x_1), ..., \mu_{N-1}(x_{N-1})\} \tag{2.27}$$

with $u_k = \mu_k(x_k) \in \mathcal{U}(x_k)$ for all $x_k \in \mathcal{X}$. Our objective is to derive an optimal policy π^* with an associated optimal cost $J^*(x) = J_{\pi^*}(x)$ for all $x \in \mathcal{X}$:

$$\pi^* = \arg\min_{\pi \in \Pi} \quad \underbrace{\mathbb{E}_{w_k}\left\{\sum_{k=0}^{N-1} \ell(x_k, u_k, w_k)\right\}}_{=J_\pi(x)} \tag{2.28a}$$

$$\text{subject to} \quad \text{dynamics and constraints.} \tag{2.28b}$$

As the time horizon goes to infinity, the optimal control problem becomes simpler. In particular, since the dynamics and the stage cost are time-invariant and the time horizon is infinite, the cost-to-go and the policy also become time-invariant. This means that it does not matter anymore when we reach a state, and the cost-to-go and optimal policy going from the particular state will remain the same.

Definition 2.3 (Stationary Policy)**.** A policy π is stationary or time-invariant if the control law is not changing at each time step: $\pi = \{\mu(x_0), \mu(x_1), ...\}$.

We can rewrite the DPA algorithm for the finite-horizon problem and explore what happens when $N \to \infty$. For convenience, we first introduce the *Bellman operator*.

Definition 2.4 (Bellman Optimality Operator)**.** Given a function $J : \mathcal{X} \mapsto \mathbb{R}$, the Bellman optimality operator is defined as

$$\mathbb{B}_* J(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w\{\ell(x, u, w) + J(f(x, u, w))\}, \quad \forall x \in \mathcal{X}. \tag{2.29}$$

The composition of the map is given by

$$\mathbb{B}_*^k J(x) = \mathbb{B}_* \mathbb{B}_*^{k-1} J(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w \left\{\ell(x, u, w) + \mathbb{B}_*^{k-1} J(f(x, u, w))\right\}, \quad \forall x \in \mathcal{X} \tag{2.30}$$

with $\mathbb{B}_*^0 J(x) = J(x)$ for all $x \in \mathcal{X}$.

Definition 2.5 (Bellman Operator)**.** For a given function $J : \mathcal{X} \mapsto \mathbb{R}$, the Bellman operator for a stationary policy $\pi = \{\mu(x_0), \mu(x_1), ...\}$ is defined as

$$\mathbb{B}_\pi J(x) = \mathbb{E}_w\{\ell(x, \mu(x), w) + J(f(x, \mu(x), w))\}, \quad \forall x \in \mathcal{X}. \tag{2.31}$$

The composition of the map is given by

$$\mathbb{B}_\pi^k J(x) = \mathbb{B}_\pi \mathbb{B}_\pi^{k-1} J(x) = \mathbb{E}_w \left\{ \ell(x, \mu(x), w) + \mathbb{B}_\pi^{k-1} J(f(x, \mu(x), w)) \right\}, \quad \forall x \in \mathcal{X} \tag{2.32}$$

with $\mathbb{B}_\pi^0 J(x) = J(x)$ for all $x \in \mathcal{X}$.

The recursion in the DPA for the finite-horizon problem can be written as

$$J_k(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w \left\{ \ell(x, u, w) + J_{k+1}\big(f(x, u, w)\big) \right\}, \forall x \in \mathcal{X}, \; k = N-1, N-2, ..., 0, \tag{2.33}$$

which can alternatively be expressed using the Bellman operator as

$$\mathbb{B}_*^{k+1} J_N(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w \left\{ \ell(x, u, w) + \mathbb{B}_*^k J_N\big(f(x, u, w)\big) \right\}, \forall x \in \mathcal{X}, \; k = 0, 1, ..., N-1. \tag{2.34}$$

We are interested in the behaviour of the recursive update (2.34) in the limit of $k \to \infty$. A key result is summarized in the theorem below.

Theorem 2.2 (Bellman Equation for an Optimal Policy)**.** Under the assumption that $\mathbb{E}_w\{\ell(x, u, w)\} \geq 0$ for all $(x, u) \in \mathcal{X} \times \mathcal{U}(x)$, the optimal cost-to-go function $J^*(x)$ associated with an optimal policy satisfies

$$J^*(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w \left\{ \ell(x, u, w) + J^*\big(f(x, u, w)\big) \right\}, \quad \forall x \in \mathcal{X}, \tag{2.35}$$

which can be equivalently written as

$$J^*(x) = \mathbb{B}_* J^*(x), \quad \forall x \in \mathcal{X}. \tag{2.36}$$

Proof. We are interested in showing that in the limit of $N \to \infty$, the recursion (2.34) becomes (2.35). We cannot directly take the limit, $k \to \infty$, as the limit and the minimization operations on the right-hand side of (2.34) cannot be interchanged in general. We instead work with $J_\pi(x)$ and show that $J^*(x) \geq \mathbb{B}_* J^*(x)$ and $J^*(x) \leq \mathbb{B}_* J^*(x)$, which together implies $J^*(x) = \mathbb{B}_* J^*(x)$. Note that, in both steps below, the assumption that $\mathbb{E}_w\{\ell(x, u, w)\} \geq 0$ ensures the monotonicity of a k-stage cost-to-go function, which allows us to interchange the limit and expectation operations as we compute the infinite-horizon costs [6, Monotone Convergence Theorem].

- ▷ $J^*(x) \geq \mathbb{B}_* J^*(x)$*:* Consider an admissible policy $\pi = \{\mu_0, \mu_1, ...\}$. The corresponding cost-to-go can be lower bounded by the optimal cost-to-go

function as follows:

$$J_\pi(x) \geq \mathbb{E}_w\{\ell(x, \mu_0(x), w) + J^*(f(x, \mu_0(x), w))\} \tag{2.37}$$
$$\geq \min_{u \in \mathcal{U}(x)} \mathbb{E}_w\{\ell(x, u, w) + J^*(f(x, u, w))\}. \tag{2.38}$$

By taking the minimum over all possible admissible policies, we have

$$J^*(x) = \min_{\pi \in \Pi} J_\pi(x) \geq \min_{u \in \mathcal{U}(x)} \mathbb{E}_w\{\ell(x, u, w) + J^*(f(x, u, w))\} = \mathbb{B}_* J^*(x). \tag{2.39}$$

▷ $J^*(x) \leq \mathbb{B}_* J^*(x)$: Consider a suboptimal policy $\pi = \{\mu_0, \mu_1, ...\}$ such that $\mathbb{B}_{\mu_k} J^*(x) \leq \mathbb{B}_* J^*(x) + \varepsilon_k$ with ε_k being a small positive number capturing the extent of suboptimality of the policy at stage k. Since $J^*(x) \geq \mathbb{B}_* J^*(x)$ (as shown in the step above), we have $\mathbb{B}_{\mu_k} J^*(x) \leq \mathbb{B}_* J^*(x) + \varepsilon_k \leq J^*(x) + \varepsilon_k$. By repeating this procedure for $\{\mathbb{B}_{\pi_{k-1}}, \mathbb{B}_{\pi_{k-2}}, ..., \mathbb{B}_{\pi_0}\}$, one can show that

$$\mathbb{B}_{\mu_0} \mathbb{B}_{\mu_1} \cdots \mathbb{B}_{\mu_k} J^*(x) \leq \mathbb{B}_* J^*(x) + \sum_{i=0}^{k} \varepsilon_i. \tag{2.40}$$

As k tends to infinity, we have

$$J^*(x) \leq J_\pi(x) = \lim_{k \to \infty} \mathbb{B}_{\mu_0} \mathbb{B}_{\mu_1} \cdots \mathbb{B}_{\mu_k} J_0 \tag{2.41}$$
$$\leq \lim_{k \to \infty} \mathbb{B}_{\mu_0} \mathbb{B}_{\mu_1} \cdots \mathbb{B}_{\mu_k} J^*(x) \tag{2.42}$$
$$\leq \mathbb{B}_* J^*(x) + \sum_{i=0}^{\infty} \varepsilon_i \tag{2.43}$$

By choosing π arbitrarily close to π^*, we can make the sum of the residual $\sum_{i=0}^{\infty} \varepsilon_i$ arbitrarily small. This implies that $J^*(x) \leq \mathbb{B} J^*(x)$.

□

As compared to (2.33), in the infinite-horizon case, the cost-to-go J on both sides of (2.35) dropped their time dependency. As a result, the optimal policy is also stationary. We can similarly write a Bellman equation for a stationary policy.

Theorem 2.3 (Bellman Equation for a Stationary Policy)**.** Under the assumption that $\mathbb{E}_w\{\ell(x, u, w)\} \geq 0$ for all $(x, u) \in \mathcal{X} \times \mathcal{U}(x)$, the Bellman equation for the cost-to-go associated with a stationary policy is

$$J_\pi(x) = \mathbb{E}_w\left\{\ell(x, \mu(x), w) + J_\pi\big(f(x, \mu(x), w)\big)\right\}, \quad \forall x \in \mathcal{X}, \tag{2.44}$$

which can be equivalently written as

$$J_\pi(x) = \mathbb{B}_\pi J_\pi(x), \quad \forall x \in \mathcal{X}. \tag{2.45}$$

In general, the solutions to the Bellman equations $\tilde{J}(x) = \mathbb{B}_* \tilde{J}(x)$ and $\tilde{J}(x) = \mathbb{B}_\pi \tilde{J}(x)$ may not be unique (i.e., there maybe multiple stationary points $\tilde{J}(x)$ satisfying the Bellman equations). The optimal cost-to-go function $J^*(x)$ and the cost-to-go function associated with a given policy $J_\pi(x)$ are the smallest stationary points of their respective Bellman equations [6, Propositions 4.1.3-4.1.4]. There are special cases (e.g., discounted cost problems, which will be discussed in a later chapter) where the unique solution of $J^*(x)$ and $J_\pi(x)$ can be guaranteed based on additional assumptions.

Theorem 2.4 (Optimality Condition for Stationary Policies)**.** Under the assumption that $\mathbb{E}_w\{\ell(x,u,w)\} \geq 0$ for all $(x,u) \in \mathcal{X} \times \mathcal{U}(x)$, a stationary policy π is optimal if and only if

$$\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J^*(x), \quad \forall x \in \mathcal{X}. \tag{2.46}$$

Proof. ($\Leftarrow$) If $\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J^*(x)$, then $J^*(x) = \mathbb{B}_\pi J^*(x)$, which implies $J^*(x) \geq J_\pi(x)$ and thus π is optimal. ($\Rightarrow$) Conversely, if $J^*(x) = J_\pi(x)$, then $\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J_\pi(x) = \mathbb{B}_\pi J^*(x)$. □

Intuitively, this theorem implies that any stationary policy π satisfying the minimal cost-to-go (i.e., $\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J^*(x)$) is optimal.

2.3 Infinite-Horizon Discrete-Time LQR

We now revisit the linear system in (2.1), this time with time-invariant dynamics (i.e., A_k and B_k being constant matrices) and an infinite-horizon cost function of quadratic form:

$$J(x_0) = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right), \tag{2.47}$$

where $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix, and $R \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix. We will show that the optimal policy is linear and time-invariant:

$$u_k = \mu(x_k) = K x_k, \tag{2.48}$$

where $K \in \mathbb{R}^{m \times n}$ is a time-invariant controller gain. Our goal is to find the optimal policy that minimizes the cost function in (2.47).

To solve the problem, analogous to the finite-horizon case, we first parameterize the cost-to-go function as

$$J(x_k) = x_k^T S x_k, \tag{2.49}$$

where S is a positive semidefinite matrix. By substituting the stage cost $\ell(x_k, u_k, w_k) = x_k^T Q x_k + u_k^T R u_k$, the cost-to-go function expression in (2.49), and the dynamics in (2.1) into the Bellman equation (2.35), we obtain the following:

$$\begin{aligned} J(x_k) &= \min_u \; x_k^T Q x_k + u_k^T R u_k + J(Ax_k + Bu_k) && (2.50)\\ &= \min_u \; x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T S (Ax_k + Bu_k) && (2.51)\\ &= \min_u \; x_k^T (Q + A^T S A) x_k + u_k^T (R + B^T S B) u_k + 2 u_k^T B^T S A x_k. && (2.52) \end{aligned}$$

Note that the minimization problem in (2.52) has a quadratic form, and we can obtain the optimal policy and express it in terms of the system matrices (A, B), cost function parameters (Q, R), and the cost-to-go function parameter S:

$$u_k = \mu^*(x_k) = \underbrace{-(R + B^T S B)^{-1} B^T S A}_{=K} x_k = K x_k. \tag{2.53}$$

To find S, we substitute the optimal solution back into the Bellman equation:

$$J(x_k) = x_k^T (Q + A^T S A) x_k + 2 x_k^T A^T S B (-K x_k) + (-K x_k)^T (R + B^T S B)(-K x_k) \tag{2.54}$$

$$= x_k^T \Big(Q + A^T S A - A^T S B K + K^T (R + B^T S B) K \Big) x_k \tag{2.55}$$

$$= x_k^T \Big(\underbrace{Q + A^T S A - A^T S B (R + B^T S B)^{-1} B^T S A}_{=S} \Big) x_k, \tag{2.56}$$

which implies

$$S = Q + A^T S A - A^T S B (R + B^T S B)^{-1} B^T S A. \tag{2.57}$$

The above equation is the DARE for infinite-horizon linear quadratic optimal control problems. If the solution to the DARE is positive semidefinite (i.e., $S \succeq 0$), then the optimal policy is

$$u_k = \mu^*(x_k) = K x_k = -(R + B^T S B)^{-1} B^T S A x_k. \tag{2.58}$$

</> See Jupyter notebook 2.1 example 2.1 for example implementation of infinite horizon LQR.

Example 2.7 (Stabilizability Condition)**.** Consider a discrete-time, infinite-horizon problem with dynamics

$$x_{k+1} = 2x_k + 0u_k \tag{2.59}$$

and cost

$$\sum_{k=0}^{\infty} \left(x_k^T x_k + u_k^T u_k \right). \tag{2.60}$$

If we put the problem in the standard form, we have $A = 2$, $B = 0$, $Q = 1$, and $R = 1$. Moreover, the policy has the form $u_k = Kx_k$, where K is constant. Does the DARE always have a solution? Is the closed-loop solution stable?

In this example, the DARE is

$$S = 2(S - 0)2 + 1, \tag{2.61}$$

which has a unique solution $S = -1/3$. Note that we require the solution to DARE to be non-negative; for this particular problem, the solution does not satisfy this condition. In general, if the system is unstable and is not stabilizable (see Definition 2.6), then we cannot find a policy that solves the optimal control problem.

Definition 2.6 (Stabilizability for Discrete-Time System)**.** For a discrete-time system, the pair (A, B) is said to be stabilizable if there exists a matrix K such that $(A + BK)$ is Schur (i.e., all eigenvalues lie strictly inside the unit circle).

Example 2.8 (Detectability Condition)**.** Consider a discrete-time, infinite-horizon problem with dynamics

$$x_{k+1} = 0.5x_k + u_k \tag{2.62}$$

and cost

$$\sum_{k=0}^{\infty} u_k^T u_k. \tag{2.63}$$

In standard form, we have $A = 0.5$, $B = 1$, $Q = 0$, and $R = 1$. Note that, for

this problem, we do not have a penalty on the state of the system, and the system without input is stable. The solution to the DARE is $S = \{0, -3/4\}$, and the corresponding optimal policy is $\mu^*(x_k) = 0$ for all k. Since the system is stable, the system state x_k converges to 0 as k tends to infinity.

However, what if the system with $\mu^*(x_k) = 0$ is unstable? Suppose $A = 2$. The solution to the DARE is $S = \{0, 3\}$. In this case, we have two non-negative solutions to the DARE. The solution $S = 0$ will lead to an unstable closed-loop system, while the solution $S = 3$ leads to a stabilizing policy $\mu^*(x_k) = -3/2\, x_k$. Now we add a small penalty term $Q(\varepsilon) = \varepsilon$ with $\varepsilon \ll 1$ and redefine the cost function as

$$\sum_{k=0}^{\infty} \left(\varepsilon x_k^T x_k + u_k^T u_k \right). \tag{2.64}$$

The solution for the perturbed problem is $S = \{(c - \sqrt{c^2 + 4\varepsilon})/2,\ (c + \sqrt{c^2 + 4\varepsilon})/2\}$ with $c = 3 + \varepsilon$. The DARE has a unique non-negative solution $S = (c + \sqrt{c^2 + 4\varepsilon})/2$, and the corresponding stabilizing optimal policy is $\mu^*(x_k) = -\frac{2\left(c+\sqrt{c^2+4\varepsilon}\right)}{2+c+\sqrt{c^2+4\varepsilon}} x_k \approx -\frac{3}{2} x_k$ (for small ε).

From this example, we see that if the system is stable, then we do not need to penalize x_k to obtain a stabilizing optimal policy; however, if the system is unstable, then we need to penalize the state–even with just a small ε penalty–to have a well-posed problem. Intuitively, this means that an unstable state must be "visible" in the cost for an optimal control strategy to be able to stabilize the system. In general, in addition to stabilizability, we also require the pair $(A, \sqrt{Q})$ to be detectable (see Definition 2.7) to solve the discrete-time, infinite-horizon problem, where $Q = \sqrt{Q}^T \sqrt{Q}$ and $\sqrt{Q}$ is a positive semidefinite matrix.

Definition 2.7 (Detectability for Discrete-Time System)**.** For a discrete-time system, the pair $(A, \sqrt{Q})$ is said to be detectable if there exists a matrix L such that $(A + L\sqrt{Q})$ is Schur.

The observations above are formalized in the theorem below. A more detailed discussion can be found in [2, Chapter 3.1].

Theorem 2.5 (Discrete-Time, Infinite-Horizon, Linear Quadratic Optimal Control)**.** For a discrete-time, infinite-horizon, linear quadratic optimal control problem, if (A, B) is stabilizable and $(A, \sqrt{Q})$ is detectable, then there exists a unique solution S, in the class of positive semidefinite matrices, to the DARE in (2.57). Moreover,

the closed-loop system matrix $(A + BK)$ is Schur.

Example 2.9 (Mobile Robot Goal-Reaching with Infinite-Horizon LQR)**.** Consider again the setup in Example 2.6 but now with an infinite-horizon cost:

$$J(x_0) = \sum_{k=0}^{\infty} u_k^2 + q\tilde{x}_k^2, \tag{2.65}$$

where $\tilde{x}_k = x_k - x_g$.

To solve the problem, we use the DARE for the case with infinite-horizon cost. By substituting $A = 1$, $B = 1$, $R = 1$ and $Q = q$ into (2.57), we have

$$S^2 - qS - q = 0. \tag{2.66}$$

Since $q \geq 0$, the solution $S = \frac{q+\sqrt{q^2+4q}}{2} \geq 0$. The optimal policy for the infinite-horizon problem is then given by

$$u_k = \mu^*(x_k) = -\frac{S}{1+S}\tilde{x}_k = \underbrace{-\frac{q+\sqrt{q^2+4q}}{2+q+\sqrt{q^2+4q}}}_{=K}\tilde{x}_k = K\tilde{x}_k. \tag{2.67}$$

Remark 2.7 (Stochastic System)**.** If our system is stochastic and has the form $x_{k+1} = Ax_k + Bu_k + w_k$ with $\mathbb{E}\{w_k\} = 0$ and $\mathbb{E}\{w_k^2\} < \infty$ (i.e., finite), the optimal policy coincides with the optimal policy in the deterministic case (certainty equivalence), but the optimal cost is different.

2.4 Continuous-Time Counterparts

Our discussion will primarily focus on discrete-time optimal control problems; however, it is worth noting that continuous-time formulations of both the DPA and LQR also exist. The infinite-horizon continuous-time LQR formulation, in particular, is widely used in control. In this section, we provide a brief overview of the core concepts and results.

Problem Setup: We consider a continuous-time linear system

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{2.68}$$

and a quadratic cost function

$$J(x_0) = \underbrace{x(T)^T Q_T x(T)}_{=\ell_T(x)} + \int_0^T \underbrace{\Big(x(t)^T Q(t) x(t) + u(t)^T R(t) u(t)\Big)}_{=\ell(t,x,u)} \, dt, \tag{2.69}$$

where t is the continuous-time variable, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $A(t)$ and $B(t)$ are matrices with consistent dimensions, T is the time horizon, $Q_T \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$ are symmetric positive semidefinite matrices, and $R \in \mathbb{R}^{m\times m}$ is a symmetric positive definite matrix.

The DPA: In the previous chapter, we introduced the DPA for solving discrete-time optimal control problems. In continuous-time optimal control problems, the dynamic programming condition leads to a terminal value problem, and the backward recursion gives rise to a partial differential equation (PDE) for $J(t,x)$, known as the Hamilton–Jacobi–Bellman (HJB) equation (see Exercise 2.2):

$$0 = \min_{u\in\mathcal{U}} \left(\ell(t,x,u) + \nabla_t J(t,x) + \nabla_x J(t,x) f(t,x,u)\right), \quad \forall\, x \in \mathcal{X},\ \forall t \in [0,T], \tag{2.70}$$

where $\nabla_t J(t,x) \in \mathbb{R}$ and $\nabla_x J(t,x) \in \mathbb{R}^{1\times n}$ denote the gradient of $J(t,x)$ with respect to t and x, respectively. The initialization step of the DPA becomes a terminal condition in the continuous-time setting:

$$J(T,x) = \ell_T(x), \quad \forall\, x \in \mathcal{X}. \tag{2.71}$$

Note that, for clarity, we have omitted the explicit time dependence when writing the signals x and u in the equations above.

Finite-Horizon Continuous-Time LQR: By substituting the linear dynamics and the quadratic cost into (2.70)-(2.71) and further parametrizing the optimal cost-to-go as

$$J(t,x) = x^T S(t) x, \tag{2.72}$$

we can derive the *continuous-time Riccati equation* for the finite-horizon linear quadratic optimal control problem (see Exercise 2.3):

$$\dot{S}(t) = -Q(t) - S^T(t)A(t) - A(t)^T S(t) + S(t)B(t)R(t)^{-1}B(t)^T S(t), \quad \forall\, t \in [0,T], \tag{2.73a}$$

$$S(T) = Q_T, \tag{2.73b}$$

where $S(t)$ is a symmetric time-varying matrix. Suppose $S(t)$ is the solution to the

terminal value problem in (2.73), then the corresponding optimal policy is

$$u(t) = \mu^*\big(t, x(t)\big) = \underbrace{-R(t)^{-1}B(t)^T S(t)}_{=K(t)}\, x(t) = K(t)x(t). \tag{2.74}$$

Infinite-Horizon Continuous-Time LQR: We now consider the infinite-horizon case in which the cost function is defined as

$$J(x_0) = \int_0^\infty \left(\frac{1}{2}x(t)^T Q x(t) + \frac{1}{2}u(t)^T R u(t)\right) dt, \tag{2.75}$$

and the dynamics are given by

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{2.76}$$

By making the optimal control problem infinite-horizon, we obtain a time-invariant policy and cost-to-go. We assume that $J(t, x)$ has the following form:

$$J(t, x) = x^T S x, \tag{2.77}$$

where S is a constant symmetric matrix. The terminal value problem in (2.73) becomes

$$0 = -Q - SA - A^T S + SBR^{-1}B^T S, \tag{2.78}$$

which is the continuous-time algebraic Riccati equation (CARE) for the infinite-horizon continuous-time linear quadratic optimal control problem. Let S be the solution of (2.78). The optimal policy can be similarly computed as

$$u(t) = \mu^*\big(x(t)\big) = \underbrace{-R^{-1}B^T S}_{=K}\, x(t) = Kx(t). \tag{2.79}$$

Similar to the discrete-time case, we obtain a valid optimal control policy if (A, B) is stabilizable and $(A, \sqrt{Q})$ is detectable. We provide the definitions of stabilizability and detectability for continuous-time systems along with the key theorem for CARE below; a formal derivation of the result can be found in [5, Sections 12.3-12.4].

Definition 2.8 (Stabilizability for Continuous-Time System)**.** For a continuous-time system, the pair (A, B) is said to be stabilizable if there exists a matrix K such that $(A + BK)$ is Hurwitz (i.e., all eigenvalues have negative real parts).

Definition 2.9 (Detectability for Continuous-Time System)**.** For a continuous-time

system, the pair $(A, \sqrt{Q})$ is said to be detectable if there exists a matrix L such that $(A + L\sqrt{Q})$ is Hurwitz.

Theorem 2.6 (Continuous-Time, Infinite-Horizon, Linear Quadratic Optimal Control)**.** For a continuous-time, infinite-horizon, linear quadratic optimal control problem, if (A, B) is stabilizable and $(A, \sqrt{Q})$ is detectable, then there exists a unique solution S, in the class of positive semidefinite matrices, to the CARE in (2.78). Moreover, the closed-loop system matrix $(A + BK)$ is Hurwitz.

2.5 Formulating a Linear Quadratic Optimal Control Problem

Linearization: Many practical systems, especially in robotics, are nonlinear. However, for many cases, we can obtain a linear representation of the dynamics, either exactly or approximately, to apply the LQR results derived in this chapter. Below, we summarize a typical approach, where we linearize the system dynamics around an equilibrium point. This approximation enables the use of a linear controller in a neighbourhood of the operating point.

Remark 2.8 (System Linearization)**.** A common approach to obtain a linear approximation of a nonlinear system is to perform a first-order Taylor expansion around an equilibrium of the nonlinear system. Consider a generic continuous-time system:

$$\dot{x} = f(x, u). \tag{2.80}$$

An equilibrium of the system is a state-input pair $(x_{\text{eq}}, u_{\text{eq}})$ that satisfies $f(x_{\text{eq}}, u_{\text{eq}}) = 0$. The linearized dynamics are given by

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}, \tag{2.81}$$

where $\tilde{x} = x - x_{\text{eq}}$, $\tilde{u} = u - u_{\text{eq}}$, $A = \frac{\partial f}{\partial x}(x_{\text{eq}}, u_{\text{eq}})$, and $B = \frac{\partial f}{\partial u}(x_{\text{eq}}, u_{\text{eq}})$.

We covered a common linearization approach above. While simple, the approximation only holds in the neighbourhood of the operating point. Other methods also exist for obtaining a linear representation of a nonlinear system (e.g., through feedback linearization, exploiting differential flatness properties of systems, or leveraging Koopman operator theory). Some of these methods use nonlinear transformations that result in exact linearizations, which can be beneficial in robotics problems, especially when the robots are expected to perform aggressive maneuvers.

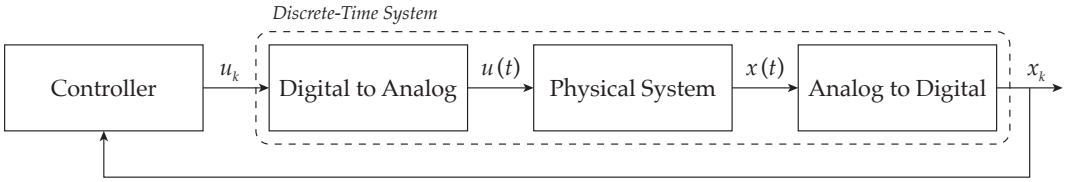


Figure 2.1: A block diagram showing the typical setup of a system. The controller receives the feedback from the system at fixed rates and computes control commands at fixed time intervals. While the dynamics of a physical system from input $u(t)$ to state $x(t)$ are often described in continuous time by an ODE, it is often more practical from a control design perspective to work with the corresponding discrete-time model capturing the dynamics from u_k to x_k.

Time Discretization: While physical systems evolve continuously, computers operate in the digital domain where sensors provide feedback from at fixed rates and controller commands are computed at fixed time intervals (see Figure 2.1). Models derived from first principles are typically expressed as ordinary differential equations (ODEs). A common method for discretizing a dynamics model is to assume that the control input u_k remains constant over the sampling interval δt (i.e., $u(t) = u_k$ for $t \in [k\delta t, (k+1)\delta t)$) and integrate the continuous-time dynamics to obtain the discrete-time representation. This approach is known as the zero-order-hold discretization (ZOH). For linear time-invariant systems, we can obtain an exact discrete-time representation (see Remark 2.9). In contrast, for nonlinear systems, we often rely on numerical integration methods (e.g., Euler or Runge-Kutta) to approximate the discrete-time dynamics (see Remark 2.10).

Remark 2.9 (Linear System Time Discretization)**.** Given a continuous-time linear system with matrices (A_c, B_c), the corresponding discrete-time system matrices (A_d, B_d) under ZOH discretization can be computed through the method of variation of parameters[2]. The resulting expressions relating (A_c, B_c) and (A_d, B_d) are as follows:

$$A_d = \exp(A_c \delta t) \quad \text{and} \quad B_d = \int_0^{\delta t} \exp(A_c \tau)\, d\tau\, B_c. \tag{2.82}$$

[2]This result can be derived using the method of variation of parameters, where the complementary solution has the form $x(t) = \exp(A_c t)v(t)$ where $v(t)$ is an unknown function. Consider the time interval from $t_0 = k\delta t$ to $t_1 = (k+1)\delta t$ with $x(t_0) = x_k$. The general solution to the ODE $\dot{x}(t) = Ax(t) + Bu(t)$ is given by $x(t) = \exp(A_c(t-t_0))x(t_0) + \int_{t_0}^{t} \exp(A_c(t-\tau))B_c u(\tau)\, d\tau$. The expression of x_{k+1} can be obtained by setting $t = t_1$: $x_{k+1} = \exp(A_c \delta t)\, x_k + \int_{t_0}^{t_1} \exp(A_c(t_1-\tau))B_c u(\tau)\, d\tau$. Under the ZOH assumption, the input $u(t)$ is held constant over the sampling interval, and we can further simplify the expression to $x_{k+1} = \exp(A_c \delta t)\, x_k + \int_0^{\delta t} \exp(A_c \tau)\, d\tau\, B_c u_k = A_d x_k + B_d u_k$ with $A_d = \exp(A_c \delta t)$ and $B_d = \int_0^{\delta t} \exp(A_c \tau)\, d\tau\, B_c$.

To compute the matrices in practice, we can first define a block matrix

$$M = \begin{bmatrix} A_c \delta t & B_c \delta t \\ 0_{m\times n} & 0_{m\times m} \end{bmatrix} \tag{2.83}$$

and compute its matrix exponential

$$\exp(M) = \begin{bmatrix} \exp(A_c \delta t) & \int_0^{\delta t} \exp(A_c \tau)\, d\tau\, B_c \\ 0_{m\times n} & I_{m\times m} \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ 0_{m\times n} & I_{m\times m} \end{bmatrix}, \tag{2.84}$$

where n and m denote the state and input dimension, respectively. The discrete-time system matrices can be directly extracted from the matrix exponential in (2.84).

Remark 2.10 (Nonlinear System Time Discretization)**.** There are numerous ways to discretize a nonlinear system. A widely used first-order integration scheme is the forward Euler method. With $x_k = x(\delta t \cdot k)$ and under ZOH discretization, we have

$$x_{k+1} = x_k + \int_{k\delta t}^{(k+1)\delta t} f(x_k, u_k)\, dt \approx x_k + \delta t\, f(x_k, u_k). \tag{2.85}$$

A common higher-order method is the Runge-Kutta fourth-order method (RK4):

$$x_{k+1} = x_k + \int_{k\delta t}^{(k+1)\delta t} f(x_k, u_k)\, dt \approx x_k + \tfrac{\delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \tag{2.86}$$

where $k_1 = f(x_k, u_k)$, $k_2 = f\left(x_k + \frac{\delta t}{2} k_1, u_k\right)$, $k_3 = f\left(x_k + \frac{\delta t}{2} k_2, u_k\right)$, and $k_4 = f\left(x_k + \delta t\, k_3, u_k\right)$. In general, higher-order methods offer better accuracy but can be computationally more expensive, both when used as a model in controller synthesis or in a numerical simulation. The integration step δt, either fixed or adaptive, must also be properly chosen to ensure numerical stability while balancing accuracy and computation cost.

Remark 2.11 (Discrete-Time versus Continuous-Time LQR)**.** When applying LQR to real-world problems, it may be possible to use the continuous-time formulation, but this is only sensible when the control loop can run sufficiently fast. Generally, it is better to explicitly take the discrete-time nature into account by discretizing the dynamics and applying the discrete-time formulation.

2.6 Verifying Stabilizability and Detectability Conditions

In the infinite-horizon case, the existence of a unique positive semidefinite solution to the DARE or CARE and thus a stabilizing optimal policy relies on two conditions: the pair (A, B) being stabilizable and $(A, \sqrt{Q})$ being detectable. A natural question, then, is how to verify these conditions in practice. We provide the key intuition and theorems for checking stabilizability and detectability conditions in this section and refer the reader to [7] for a more in-depth review on this topic.

The stronger but closely connected conditions are controllability (Definition 2.10) and observability (Definition 2.11). Intuitively, a controllable system provides enough control authority to move the system's state from any initial condition to any other point in the state space (i.e., every state in the state space is reachable). Observability, on the other hand, is often discussed in the context of state estimation. This means that we have enough information from system outputs to fully infer the internal state vector. In the context of linear quadratic optimal control, observability implies that the cost function provides enough information to reason about the system's behaviour.

Definition 2.10 (Controllability)**.** A pair (A, B) with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ is controllable if the controllability matrix $Q_c \in \mathbb{R}^{n\times nm}$ has full row rank:

$$\text{rank}(Q_c) = \text{rank}\left(\begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}\right) = n. \tag{2.87}$$

Definition 2.11 (Observability)**.** A pair (A, C) with $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{p\times n}$ is observable if the pair (A^T, C^T) is controllable, or equivalently, the observability matrix $Q_o \in \mathbb{R}^{nq\times n}$ has full column rank:

$$\text{rank}(Q_o) = \text{rank}\left(\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n. \tag{2.88}$$

In general, controllability and observability imply stabilizability and detectability, respectively (i.e., we have enough authority and information to control and observe the state or the behaviour of the system). In other words, if (A, B) is controllable and $(A, \sqrt{Q})$ is observable, then we are guaranteed to have a unique

positive semidefinitive solution to the DARE and CARE, and the resulting optimal policy is stabilizing. A common approach to check controllability, and, analogously, observability, is via the Popov-Belevitch-Hautus (PBH) test (Theorem 2.7).

Theorem 2.7 (PBH Tests for Controllability and Observability)**.** Let $\sigma(A)$ denote the spectrum of A. A pair (A, B) is controllable if and only if

$$\big(\forall \lambda \in \sigma(A)\big) \ \text{rank} \left(\begin{bmatrix} A - \lambda I & B \end{bmatrix} \right) = n. \tag{2.89}$$

A pair (A, C) is observable if and only if

$$\big(\forall \lambda \in \sigma(A)\big) \ \text{rank} \left(\begin{bmatrix} C \\ A - \lambda I \end{bmatrix} \right) = n. \tag{2.90}$$

When controllability and observability do not hold, a linear quadratic optimal control problem may still satisfy stabilizability and detectability conditions. These weaker conditions require only that the unstable modes of the system are controllable and observable. The stabilizability and detectability conditions can be similarly verified using PBH tests (Theorem 2.8).

Theorem 2.8 (PBH Tests for Stabilizability and Detectability)**.** Let $\sigma_{\text{un}}(A) \subseteq \sigma(A)$ denote the set of eigenvalues corresponding to the unstable modes of the system (i.e., eigenvalues λ satisfying $\text{Re}(\lambda) \geq 0$ for continuous-time systems and $|\lambda| \geq 1$ for discrete-time systems). A pair (A, B) is stabilizable if and only if

$$\big(\forall \lambda \in \sigma_{\text{un}}(A)\big) \ \text{rank} \left(\begin{bmatrix} A - \lambda I & B \end{bmatrix} \right) = n. \tag{2.91}$$

A pair (A, C) is detectable if and only if

$$\big(\forall \lambda \in \sigma_{\text{un}}(A)\big) \ \text{rank} \left(\begin{bmatrix} C \\ A - \lambda I \end{bmatrix} \right) = n. \tag{2.92}$$

In practice, when using any toolbox to solve the DARE or CARE, it is important to verify that the stabilizability and detectability conditions are satisfied to ensure the problem is well-posed. If these conditions are not met, the solver may still return a stabilizing controller in some cases, but in others, it may fail to converge.

Chapter Summary

Linear Quadratic Optimal Control

- **Discrete-Time Linear Quadratic Regulator**
  - Dynamics model:

$$x_{k+1} = A_k x_k + B_k u_k, \quad \forall k \in \{0, 1, ..., N\}, \tag{2.93}$$

    where $k \in \mathbb{Z}_{\geq 0}$ is the discrete-time index, $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the input, (A_k, B_k) are system matrices with consistent dimensions, and $N \in \mathbb{Z}_{\geq 0}$ is the time horizon.
  - Cost:

$$J(x_0) = x_N^T Q_N x_N + \sum_{k=0}^{N-1} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right), \tag{2.94}$$

    where $Q_k \in \mathbb{R}^{n \times n}$ and $Q_N \in \mathbb{R}^{n \times n}$ are symmetric positive semidefinite matrices, and $R_k \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix.
  - Finite-horizon solution:

    Initialization: $S_N = Q_N$
    Backward recursion:
    $S_k = Q_k + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k (R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k$
    (discrete-time algebraic Riccati equation for the finite-horizon problem)
    Optimal policy: $u_k = \mu_k^*(x_k) = K_k x_k$ with $K_k = -(R_k + B^T S_{k+1} B)^{-1} B^T S_{k+1} A$
  - Infinite-horizon solution (with $A_k = A$, $B_k = B$, $Q_k = Q$, and $R_k = R$):

    Matrix quadratic equation:
    $S = Q + A^T S A - A^T S B (R + B^T S B)^{-1} B^T S A$
    (discrete-time algebraic Riccati equation for the infinite-horizon problem, which has a unique positive semidefinite solution if (A, B) is stabilizable and $(A, \sqrt{Q})$ is detectable)
    Optimal policy: $u_k = \mu^*(x_k) = K x_k$ with $K = -(R + B^T S B)^{-1} B^T S A$

- **Continuous-Time Linear Quadratic Regulator**

- Dynamics model:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad \forall t \in [0, T], \tag{2.95}$$

where $t \in \mathbb{R}_{\geq 0}$ is the continuous-time variable, $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the input, and (A, B) are system matrices with consistent dimensions.

- Cost:

$$J(x_0) = x(T)^T Q_T x(T) + \int_0^T \Big(x(t)^T Q(t) x(t) + u(t)^T R(t) u(t) \Big)\, dt, \tag{2.96}$$

where $Q(t) \in \mathbb{R}^{n \times n}$ and $Q_T \in \mathbb{R}^{n \times n}$ are symmetric positive semidefinite matrices, and $R(t) \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix.

- Finite-horizon solution:

  Initialization (terminal condition): $S(T) = Q_T$

  Matrix differential equation:

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

  (continuous-time Riccati differential equation for the finite-horizon problem)

  Optimal policy: $u(t) = \mu_t^*(x(t)) = K(t)x(t)$ with $K(t) = -R(t)^{-1}B^T(t)S(t)$

- Infinite-horizon solution (with $A(t) = A$, $B(t) = B$, $Q(t) = Q$, and $R(t) = R$):

  Matrix quadratic equation: $0 = -Q - SA - A^T S + SBR^{-1}B^T S$

  (continuous-time algebraic Riccati equation for the infinite-horizon problem, which has a unique solution if (A, B) is stabilizable and $(A, \sqrt{Q})$ is detectable)

  Optimal policy: $u(t) = \mu^*(x(t)) = Kx(t)$ with $K = -R^{-1}B^T S$

Infinite-Horizon Optimal Control

- **Stationarity in the Infinite-Horizon Case:** Under the assumption that $\mathbb{E}_w\{\ell(x, u, w)\} \geq 0$ for all $(x, u) \in \mathcal{X} \times \mathcal{U}(x)$, the optimal cost-to-go function and the corresponding optimal policy for a time-invariant nonlinear system $f(x_k, u_k, w_k)$ with stage cost $\ell(x_k, u_k, w_k)$ become time-invariant as the horizon $N \to \infty$.

- **Bellman Equations**
  - The optimal cost-to-go function $J^*(x)$ associated with an optimal policy satisfies

$$J^*(x) = \min_{u \in \mathcal{U}(x)} \mathbb{E}_w \left\{ \ell(x,u,w) + J^*\big(f(x,u,w)\big) \right\} = \mathbb{B}_* J^*(x), \quad \forall x \in \mathcal{X}, \tag{2.97}$$

    where $\mathbb{B}_*$ is the Bellman optimality operator.
  - The cost-to-go function $J_\pi(x)$ associated with a stationary policy π satisfies

$$J_\pi(x) = \mathbb{E}_w \left\{ \ell(x,\mu(x),w) + J_\pi\big(f(x,\mu(x),w)\big) \right\} = \mathbb{B}_\pi J_\pi(x), \quad \forall x \in \mathcal{X}, \tag{2.98}$$

    where $\mathbb{B}_\pi$ is the Bellman operator associated with the policy π.
- **Optimality Condition:** A stationary policy π is optimal if and only if

$$\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J^*(x), \quad \forall x \in \mathcal{X}. \tag{2.99}$$

2.7 Exercises

Exercise 2.1 (Short Questions)**.** For the problems below, select the option(s) that best answer each question and justify your selection(s).

a) Which of the following statements about LQRs are correct?

(A) The LQR is the optimal solution to an optimal control problem with linear dynamics and quadratic cost and no constraints.

(B) A unique optimal stabilizing policy exists for the infinite-horizon LQR problem if the pair (A, B) is stabilizable and the pair $(A, \sqrt{Q})$ is detectable.

(C) The LQR can be derived using dynamic programming.

(D) For the infinite-horizon case, a stabilizing LQR solution can always be found.

b) Consider a discrete-time dynamical system

$$x_{k+1} = A_k x_k + B_k u_k. \tag{2.100}$$

Which of the following statements about discrete-time LQR are correct?

(A) LQR can be used for setpoint stabilization and reference trajectory tracking.

(B) Infinite-horizon LQR can handle time-varying system dynamics.

(C) For an infinite horizon and time-invariant dynamics, the optimal feedback gain is constant.

(D) The solutions of the DARE for a finite-horizon problem can be computed iteratively in a forward manner starting from $k = 0$.

(E) If A_k is time-invariant, $B_k = [0, 1]^T$ and we consider an infinite horizon, the DARE is guaranteed to have a positive semidefinite solution.

c) Which of the following statements are correct regarding an infinite-horizon optimal control problem?

(A) The optimal policy depends only on the current state of the system.

(B) The optimal cost-to-go function depends only on the current state of the system.

(C) Infinite-horizon optimal control problems cannot consider input constraints.

(D) For the infinite-horizon LQR problem, the Bellman optimality equation $J^*(x) = \mathbb{B}_* J^*(x)$ leads to the DARE.

d) Suppose we are given the optimal cost-to-go $J^*(x)$ for an infinite-horizon optimal control problem as well as an arbitrary stationary policy π. We can we say about the policy π if $\mathbb{B}_* J^*(x) = \mathbb{B}_\pi J^*(x)$ holds for all $x \in \mathcal{X}$?

(A) The policy π is optimal, and the optimal policy is unique.

(B) The policy π is optimal.

(C) The cost-to-go associated with the policy $J_\pi(x)$ equals to the optimal cost-to-go $J^*(x)$ for all $x \in \mathcal{X}$.

(D) We cannot conclude anything about the policy π.

e) Consider a continuous-time linear-quadratic optimal control problem where the system dynamics are given by $\dot{x}(t) = Ax(t) + Bu(t)$, and the cost function to be minimized is $J = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t)\right) dt$, with $Q \geq 0$ and $R > 0$. Assume that the pair (A, B) is stabilizable. The optimal control law is derived using the CARE. Select all true statements.

(A) The CARE is a differential equation that determines the optimal state feedback gain matrix K.

(B) The CARE for this problem is given by $Q = -A^T S - SA + SBR^{-1}B^T S$.

(C) The solution of the CARE, S, is used to compute the optimal control law $u(t) = -R^{-1}B^T S x(s)$.

(D) The CARE ensures that the closed-loop system is stable if Q and R are chosen appropriately.

(E) None of the above.

f) In the context of linear-quadratic optimal control, what is the main difference between continuous-time and discrete-time LQR?

(A) Continuous-time LQR employs differential equations to describe system dynamics, whereas discrete-time LQR utilizes difference equations.

(B) Continuous-time LQR is only applicable to finite-horizon problems, whereas discrete-time LQR can be applied to both finite and infinite-horizon problems.

(C) Discrete-time LQR does not consider constraints on control inputs, while continuous-time LQR incorporates such constraints.

(D) Discrete-time LQR is more suitable for a discrete-time system, whereas continuous-time LQR only works if the control frequency is sufficiently high.

Exercise 2.2 (HJB Derivation)**.** Consider a continuous-time optimal control problem, where the dynamics are given by

$$\dot{x}(t) = f\big(t, x(t), u(t)\big), \tag{2.101}$$

and the cost to be minimized is

$$\ell_T(x(T)) + \int_0^T \ell\big(t, x(t), u(t)\big)\, dt. \tag{2.102}$$

Here, $t \in \mathbb{R}_+$ denotes the continuous-time variable, $x(t) \in \mathcal{X}$ is the system state, $u(t) \in \mathcal{U}$ is the system input, $f : \mathbb{R}_+ \times \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}$ is the system dynamics equation, and $\ell : \mathbb{R}_+ \times \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}$ and $\ell_T : \mathcal{X} \mapsto \mathbb{R}$ are the stage and terminal cost functions, respectively. Derive the HJB equation in (2.70) for a continuous-time optimal control problem starting from the Bellman equation in (2.44).

Exercise 2.3 (Finite-Horizon CARE Derivation)**.** Derive the CARE for the finite-horizon linear quadratic optimal control problem using the terminal value problem given in (2.70)–(2.71).

Exercise 2.4 (Finite-Horizon LQR [8, Example 13.2])**.** For the discrete-time linear system

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

with initial condition $x_0 = \bar{x}_0$ and $x_k, u_k \in \mathbb{R}$, we want to design an optimal feedback controller $u_k = K_k x_k$ that minimizes the cost

$$J = q x_N^2 + \sum_{k=0}^{N-1} x_k^2 + r u_k^2$$

with cost weights $q \geq 0$ and $r \geq 0$.

a) Find the recursion for the Riccati Matrix S_k and the controller K_k.

b) Assume $N = 4$ and $r = 1$. Compute $S_0, S_1, \ldots, S_4$ and $K_0, K_1, \ldots, K_3$.

c) Compare and discuss the resulting trajectories of x_k for $q = 0$ and $q \to \infty$ for an initial value of $x_0 = 2$.

d) What is the resulting optimal cost J^* for the realizations $q = 0$ and $q \to \infty$?

Exercise 2.5 (Relationship between LQR and PD Control)**.** We can model the

continuous-time dynamics of a mobile robot moving towards a target position as

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \tag{2.103}$$

where $x(t) = [p(t), v(t)]^T$ represents the position and velocity, and $u(t)$ is the commanded acceleration. The system is controlled by a proportional-derivative (PD) controller of the form

$$u(t) = -k_p p(t) - k_v v(t) \tag{2.104}$$

with $k_p = 4$ and $k_d = 5$.

a) Express the PD controller (2.104) as a state feedback control law of the form $u(t) = Kx(t)$.

Let us investigate the relationship between the PD control law (2.104) and an LQR policy that minimizes the infinite-horizon cost

$$J(x_0) = \int_0^\infty \left(\frac{1}{2} x(t)^T Q x(t) + \frac{1}{2} u(t)^T R u(t) \right) dt. \tag{2.105}$$

Recall that the optimal policy for such a continuous-time linear optimal control problem can be calculated as

$$u(t) = -R^{-1} B^T S x(t) = Kx(t), \qquad 0 = -Q - SA - A^T S + SBR^{-1} B^T S. \tag{2.106}$$

b) Assuming $R = 1$, what is the matrix S?

c) Find a diagonal cost matrix Q such that the PD control law (2.104) is optimal with respect to the cost (2.105). Consider two cases: $R = 1$ and $R = 2$.

Exercise 2.6 (Mobile Robot Output-Based Regulation)**.** We consider an infinite-horizon optimal control problem in which the objective is to drive a mobile robot system to a target position. In addition to the system dynamics, we include an output equation, and the controller has access only to corrupted position measurements denoted by y_k. The discrete-time system is described by

$$\begin{aligned} x_{k+1} &= x_k + u_k + w_k, \\ y_k &= x_k + v_k, \end{aligned}$$

where $x_k \in \mathbb{R}$ is the true position of the mobile robot, $u_k \in \mathbb{R}$ is the control

signal, $y_k \in \mathbb{R}$ is the measured position corrupted by measurement noise v_k, and w_k represents disturbances in the terrain. We assume that the measurement noise v_k and the disturbances w_k are independent random variables with zero mean (i.e., $\mathbb{E}[v_k] = 0$ and $\mathbb{E}[w_k] = 0$) and unit variance (i.e., $\mathbb{E}[v_k^2] = 1$ and $\mathbb{E}[w_k^2] = 1$). Our goal is to minimize the following cost:

$$J = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{E}_{w_k, v_k} \left[\frac{x_k}{2}^2 + u_k^2 \right].$$

For this problem, we are given an output feedback controller $u_k = -\frac{1}{2} y_k$ that tries to move the robot's position to $x = 0$.

a) Write out the closed-loop dynamics (i.e, write x_{k+1} as a function of x_k, v_k, and w_k).

b) Show that this closed-loop system is stable in expectation. *Hint: Note that a discrete-time system is stable if the eigenvalues of the dynamics lie inside of the unit circle. That is, you should see that the closed-loop dynamics in expectation have the form $x_{k+1} = r x_k$. The eigenvalue of this 1D discrete-time system is r, and if $|r| < 1$, then the system is stable. Why might this be?*

c) What is the infinite horizon cost J under the given feedback policy? *Hint: To find $\mathbb{E}[x_k^2]$, you can use the fact that the system is stable and thus $\mathbb{E}[x_{k+1}^2] \approx \mathbb{E}[x_k^2]$ as $k \to \infty$.*

d) Find the optimal closed-loop feedback gain α for the control law $u_k = \alpha y_k$. You can assume there is no noise for this problem. *Hint: you have to solve the DARE for the infinite-horizon problem.*

Exercise 2.7 (Infinite-Horizon LQR [8, Example 13.3])**.** Consider the discrete-time linear system

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

with initial condition $x_0 = \bar{x}_0$ and the cost function

$$J(x_0) = \sum_{k=0}^{\infty} x_k^2 + r u_k^2, \quad r \geq 0.$$

a) Compute the stationary Riccati matrix S.

b) Give an equation for K of the optimal feedback control $u_k = K x_k$.

c) Give the dynamics of the closed-loop system and discuss the stability of the closed-loop.

d) Discuss the closed-loop behaviour for $r = 0$ and $r \to \infty$ and interpret the cost function.

Exercise 2.8 (Infinite-Horizon LQR)**.** Consider the system dynamics

$$x_{k+1} = Ax_k + Bu_k, \tag{2.107}$$

with state $x_k \in \mathbb{R}^2$ and input $u_k \in \mathbb{R}$, and the following infinite-horizon cost function

$$J = \sum_{k=0}^{\infty} x_k^T Q x_k + r u_k^2, \tag{2.108}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad r = 1.$$

a) Compute the solution S to the DARE by determining the parameters $b, c \in \mathbb{R}$, $c > 0$, of the parameterized solution matrix defined as

$$S = \begin{bmatrix} c+1 & b \\ b & c \end{bmatrix}.$$

b) For the given system and cost function, the pair (A, B) is stabilizable, and the pair $(A, \sqrt{Q})$ is detectable. Without computing the optimal control gain K, discuss which of the following values could be or could not be eigenvalues of the closed-loop matrix $(A + BK)$:

(i) $\lambda_1 = 0.3$

(ii) $\lambda_2 = -1$

(iii) $\lambda_3 = 0.25 + 0.25i$

(iv) $\lambda_4 = -3$.

c) Compute the optimal control gain K.

d) Compute the eigenvalues of the matrix $(A + BK)$ and comment on the stability of the closed-loop system.

Exercise 2.9 (LQR for Mobile Robot Steering Problem)**.** Consider a mobile robot moving in a plane with unit speed. For this question, we are only interested in its lateral motion $y \in \mathbb{R}$. The dynamics of the robot are:

$$\dot{y}(t) = \sin\theta(t), \quad \dot{\theta}(t) = u(t), \tag{2.109}$$

where $y(t) \in \mathbb{R}$ is the position of the robot along the y-axis in meters, $\theta(t) \in \mathbb{R}$ is the heading angle of the robot in radians, and $u(t) \in \mathbb{R}$ is the input to the system in radians per second. Our goal is to design a state feedback policy $u(t) = \mu\big(x(t)\big)$ with $x(t) = [y(t),\, \theta(t)]^T$ that allows the robot to move from a given initial position $x_0 \in \mathbb{R}^2$ to a given target position $x_T \in \mathbb{R}^2$. The cost function has the following form:

$$J(x_0) = \int_0^\infty \big(x(t) - x_T\big)^T Q\big(x(t) - x_T\big) + r\, u(t)^2\, dt. \tag{2.110}$$

a) Consider the linearized dynamics of (2.109) about the equilibrium of the system $\big(x_{\text{eq}}, u_{\text{eq}}\big) = \big([\bar{y}_T, 0]^T, 0\big)$, where $\bar{y}_T$ is a constant target y-position. Show that the linearized system is $\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$ with $\tilde{x} = x - x_{\text{eq}}$ and $\tilde{u} = u - u_{\text{eq}}$, and

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.111}$$

b) Is the linearized system stabilizable? What does it mean intuitively if a system is stabilizable? *Hint: Start by finding the eigenvalues λ_i of A. A linear system is stabilizable if and only if $[A - \lambda_i I,\, B]$ has full rank for any unstable eigenvalue (i.e., any eigenvalue with non-negative real part: real$(\lambda_i) \geq 0$), where I is the identity matrix with consistent dimensions.*

c) Consider the cost function in (2.110) with $x_T = x_{\text{eq}} = [\bar{y}_T,\, 0]^T$, the linearized system dynamics obtained in part a), and a linear feedback policy of the form $u(t) = K\tilde{x}(t)$. What are the conditions on Q and r to guarantee that the linear-quadratic optimal control problem is solvable and has a unique and bounded solution?

d) Set the cost function parameters Q and r as follows:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad r = 1. \tag{2.112}$$

Find the optimal feedback policy $u^*(t) = K\tilde{x}(t)$. Show all important steps of your derivation.

Chapter 3

Optimization Fundamentals

In the previous chapter, we focused on linear quadratic optimal control problems, a fundamental class of problems in which linear system dynamics are coupled with a quadratic cost function. This structure allows us to simplify the DPA to a set of (recursive) matrix equations known as the algebraic Riccati equations. The resulting optimal controllers are referred to as LQRs.

Besides deriving the algebraic Riccati equations for different variants of linear quadratic optimal control problems, we also examined the infinite-horizon setting and showed that, under a mild condition, the optimal cost-to-go and the resulting control policy become time-invariant. Lastly, we touched on common techniques such as linearization and discretization, which allow us to apply the LQR to nonlinear, continuous-time systems.

In this chapter, we will review a few core ideas from optimization. We start with unconstrained optimization problems and introduce optimality conditions as well as numerical algorithms for solving them. We then extend the discussion to general constrained optimization problems. The results covered in this chapter serve as the basis for implementing iterative optimal control algorithms and model predictive control in the subsequent chapters.

3.1 Why is Optimization Relevant?

Optimization is ingrained in every aspect of robotics, including perception, state estimation/localization, motion planning, control, as well as robot learning. Below are a few examples:

- *motion planning:* find the most energy-efficient path through a cluttered environment;
- *localization:* estimate the most probable position of a robot in a map based on the current image observed;
- *control:* find the next actuator command that causes the robot to stay as close as possible to a desired path;
- *machine learning:* find a model that best explains the data that the robot collected;
- *skill acquisition:* optimize the robot's performance over many training runs.

While this course focuses on control, almost all problems in robotics, whether it is perception or decision-making, can be thought of as optimization problems. The optimization problems in each of the subfields in robotics have different challenges that often motivate specialized methods to obtain tractable problems and efficient solutions. Despite these differences, the fundamentals of optimization, however, remain to be the basis of the different fields in robotics.

Example 3.10 (The Basic Problem Revisited)**.** The basic problem introduced in Chapter 1 can generally be formulated as a constrained optimization problem. Suppose the system is initialized at $\bar{x}_0$. The basic problem can be written as an optimization problem:

$$J^*(\bar{x}_0) = \min_{\pi \in \Pi} \quad \mathbb{E}_{w_k} \left\{ \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k) \right\} \tag{3.1a}$$

$$\text{subject to} \quad x_{k+1} = f_k(x_k, u_k, w_k) \quad w_k \sim Pr_k(\cdot \mid x_k, u_k), \quad \forall k \in \{0, 1, , N-1\} \tag{3.1b}$$

$$u_k = \mu_k(x_k), \quad \forall k \in \{0, 1, , N-1\} \tag{3.1c}$$

$$x_0 = \bar{x}_0 \tag{3.1d}$$

$$x_k \in \mathcal{X},\ u_k \in \mathcal{U}(x_k), \quad \forall k \in \{0, ..., N-1\}, \tag{3.1e}$$

where k is the discrete-time index or stage, $x_k \in \mathcal{X}_k$ is the system state, $u_k \in \mathcal{U}_k(x_k)$ is the system input, $w_k \in \mathcal{D}_k$ represents system disturbance and follows a probability distribution $Pr_k(\cdot \mid x_k, u_k)$, N is the time horizon, f_k is the dynamics model, ℓ_k is the stage cost, ℓ_N is the terminal cost, $\pi = \{\mu_0, \mu_1, ..., \mu_N\}$ is the policy to be optimized, and Π is the set of admissible policies.

From the first two chapters, we saw how optimal control problems can be addressed by leveraging the principle of optimality and the DPA, and further delved into the class of linear quadratic problems, where the structure of the dynamics and cost function enabled analytical or efficiently computable solutions. The general formulation in (3.1), however, does not possess a nice structure in the presence of nonlinear dynamics, non-quadratic costs, or complex constraints. As we will see in the following chapters, this general optimization problem is often simplified through linear quadratic approximations and/or tackled through a receding-horizon approach. These simplified subproblems, typically convex optimization problems, can be solved efficiently using the state-of-the-art numerical solvers. The most challenging aspect of applying such techniques to real-world robotics problems often lies in understanding how to properly formulate the subproblems, analyze their properties, and efficiently exploit the underlying structure of the optimal control problem (e.g., dynamics [9] and constraints [10]).

3.2 Problem Definition

A generic optimization problem has the following form:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0 \quad \forall\, i = \{1, 2, ..., N\}, \\ & g_j(x) \leq 0, \quad \forall\, j = \{1, 2, ..., M\}, \end{aligned} \tag{3.2}$$

where $x \in \mathbb{R}^n$ is the *optimization variable* of the problem, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the *objective function*, and $h_i : \mathbb{R}^n \mapsto \mathbb{R}$ and $g_j : \mathbb{R}^n \mapsto \mathbb{R}$ are the *equality and inequality constraint functions*, respectively. The objective function and the constraint functions can be general nonlinear functions. A general optimization problem with at least one nonlinear function either as the objective function, as the inequality constraints or as the equality constraints is called a *nonlinear program*. In this chapter, we generally assume that f, h_i, and g_j are differentiable functions.

For the convenience of discussion, we first introduce a few key terminologies.

Definition 3.12 (Feasible Point and Feasible Set)**.** A *feasible point* of a constrained optimization problem is any point x satisfying $h_i(x) = 0$ for all $i = \{1, 2, ..., N\}$ and $g_j(x) \leq 0$ for all $j = \{1, 2, ..., M\}$. The *feasible set* is the set containing all feasible

points:

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid h_i(x) = 0,\ \forall\, i = \{1, 2, ..., N\} \text{ and } g_j(x) \leq 0,\ \forall\, j = \{1, 2, ..., M\}\}. \tag{3.3}$$

Definition 3.13 (Optimal Value and Optimal Solution)**.** Consider the optimization problem in (3.2). The lowest possible cost f^*, also called the *optimal value*, is given by an *optimal solution* (or optimizer) x^*:

$$f(x) \geq f(x^*) = f^*, \quad \forall x \in \mathcal{X} \text{ with } x^* \in \mathcal{X}. \tag{3.4}$$

A point $x^* \in \mathcal{X}$ is a *global minimum* if

$$(\forall x \in \mathcal{X}) \quad f(x^*) \leq f(x). \tag{3.5}$$

A point $x^* \in \mathcal{X}$ is a *local minimum* if there exists some $\varepsilon > 0$ such that

$$(\forall x \in \mathcal{X} \text{ with } ||x - x^*|| \leq \varepsilon) \quad f(x^*) \leq f(x). \tag{3.6}$$

Definition 3.14 (Critical Point of a Constrained Optimization)**.** A critical point of a constrained optimization problem is a local maximum, minimum or saddle point of f within the feasible set.

Note that, if there is no point x satisfying the constraints (i.e., the set $\mathcal{X}$ is empty), then the optimization problem is infeasible. For a feasible nonlinear program, finding the global optimal solution is generally challenging. One way to think about solving the general problem is in two steps: *(1)* find all feasible x that satisfy all constraints and *(2)* amongst all feasible x, find the point that minimizes the objective function. However, finding all feasible x itself can be a difficult problem. One alternative way to solve nonlinear programs is by iteratively solving simpler sub-problems. These numeric solvers often start with an initial guess that is feasible and converge to locally optimal solutions.

3.3 Unconstrained Optimization

Consider the optimization problem in (3.2). The optimization problem is said to be *unconstrained* if $M = N = 0$.

3.3.1 Gradient of the Objective Function and Direction of the Steepest Descent

The gradient at any point $x = [x_1, x_2, ..., x_n]^T$ is given by

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right] \in \mathbb{R}^{1\times n}. \tag{3.7}$$

The negation of the gradient vector (i.e., $-\nabla f$) describes the direction of the steepest descent.

To see why the gradient direction determines the direction of *steepest* descent (or ascent), we first write the first-order Taylor expansion around a point $x_0 \in \mathbb{R}^n$:

$$f(x) \approx f(x_0) + \nabla f(x_0)(x - x_0). \tag{3.8}$$

If we take a small step in the direction of the gradient, $x - x_0 = \alpha\nabla^T f(x_0)/||\nabla f(x_0)||_2$ with $\alpha \ll 1$, we have

$$f(x) = f\left(x_0 + \frac{\alpha\nabla^T f(x_0)}{||\nabla f(x_0)||_2}\right) \approx f(x_0) + \alpha||\nabla f(x_0)||_2. \tag{3.9}$$

Note that the sign of α determines whether the value of f increases or decreases locally (i.e., whether it is an ascent or a descent). If we now take a step along an arbitrary direction $v \in \mathbb{R}^n$ with v being a unit vector, then we have

$$f(x_0 + \alpha v) \approx f(x_0) + \alpha\nabla f(x_0)\, v = f(x_0) + \alpha \underbrace{||\nabla f(x_0)||_2 \cos(\theta)}_{\leq ||\nabla f(x_0)||_2}, \tag{3.10}$$

where θ is the angle between v and the gradient direction. Note that, if we keep the step size α the same, then the steepest descent (or ascent) happens when $\cos(\theta) = 1$, which is the case when v is aligned with the gradient direction (i.e., when $v = \nabla^T f(x_0)/||\nabla f(x_0)||_2$).

3.3.2 First-Order and Second-Order Optimality Conditions

Motivated by the idea of gradient descent, we see that if x is a local minimum, then $\nabla f(x) = 0$, which is a necessary but not a sufficient condition. This leads to the first-order optimality condition, which we formalize below.

Definition 3.15 (Stationary Point or Critical Point)**.** A *stationary point* or a *critical point* of a differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a point $x \in \mathbb{R}^n$ satisfying $\nabla f(x) = 0$.

Theorem 3.9 (First-Order Optimality Condition)**.** If x^* is a local minimum of $f : \mathbb{R}^n \mapsto \mathbb{R}$, then x^* is a stationary point with $\nabla f(x^*) = 0$.

The first-order optimality condition allows us to devise a high-level strategy for the unconstrained optimization problem: *(1)* finding all points x_i satisfying $\nabla f(x_i) = 0$ and *(2)* evaluating $f(x_i)$ for all i and finding the x_i with the lowest function value, and *(3)* checking the value of $f(x)$ as $x \to \pm\infty$ (which can be lower than the smallest function value from the first two steps). This process allows us to find the global minimum. However, it is not always tractable to find all local minima in practice.

The first-order optimality condition is necessary but not sufficient. To obtain a necessary and sufficient condition of a local minimum, we define the Hessian matrix of a twice differentiable function f as

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}. \tag{3.11}$$

We now write the second-order Taylor expansion of f about a point x_0:

$$f(x) \approx f(x_0) + \nabla f(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T H_f(x_0)(x - x_0). \tag{3.12}$$

At a stationary point x^*, we have $\nabla f(x^*) = 0$ and

$$f(x) \approx f(x^*) + \frac{1}{2}(x - x^*)^T H_f(x^*)(x - x^*). \tag{3.13}$$

If $H_f(x^*)$ is positive definite (i.e., $x^T H_f x > 0$ for all $x \neq 0$ or equivalently all eigenvalues are real and positive), then $f(x) \geq f(x^*)$ for all x close to x^*. This implies that x^* is a local minimum. We formally write the second-order optimality condition below, which is necessary and sufficient.

Theorem 3.10 (Second-Order Optimality Condition)**.** If (and only if) $H_f(x^*)$ is positive definite, then the stationary point x^* is a local minimum of f.

We can use the second-order optimality condition to test if a stationary point is a local minimum without knowing the existence of any other stationary points. If we know more about the function f, we can make even stronger statements about the optimum. One important property of f that is often exploited is convexity; this also leads to an important class of optimization problems, which we cover in Section 3.4.3.

3.3.3 Algorithms for Solving Unconstrained Problems

In this subsection, we outline two algorithms for solving the unconstrained optimization problem, gradient descent and Newton's method.

Gradient Descent. One common method for solving an unconstrained optimization problem is gradient descent, which iteratively updates the solution for the unconstrained optimization problem by taking a step along the direction of the steepest descent.

Let $\widetilde{x}^0 = x_0$ be an initial guess of the optimizer for the unconstrained problem. The gradient descent algorithm iteratively updates the solution $\widetilde{x}^l$ as follows:

$$\widetilde{x}^{l+1} \longleftarrow \widetilde{x}^l - \alpha \nabla^T f(\widetilde{x}^l), \tag{3.14}$$

where l denotes the iteration number, and $\alpha > 0$ is the step size, which can be either fixed or determined via line search methods (Remark 3.12). The algorithm repeats until a stopping criterion is met (Remark 3.13).

Remark 3.12 (Line Search Methods)**.** Given a search direction v^l (e.g., the negative gradient direction in the case of gradient descent), the idea of line search is to determine the optimal step size α that would maximally reduce the value of the objective function:

$$\alpha = \min_{\alpha' \geq 0} \; f(\widetilde{x}^l + \alpha' v^l). \tag{3.15}$$

The optimized α from the exact line search in (3.15) yields an iterate $\widetilde{x}^{l+1} = \widetilde{x}^l + \alpha v^l$ that is at least as good as the current one (i.e., $f(\widetilde{x}^{l+1}) \leq f(\widetilde{x}^l)$). In practice, for a generic objective function $f(x)$, it can be expensive to solve the one-dimensional problem in (3.15) exactly at each iteration; inexact methods such as backtracking [11] are often used to find a sufficiently good step size that balances between computational efficiency and convergence performance.

Remark 3.13 (Stopping Criterion)**.** There are several ways to define a stopping criterion. Two common approaches are based on the change in the objective function value or the change in the solution between consecutive iterations (e.g.,

$||f(\widetilde{x}^{l+1}) - f(\widetilde{x}^{l})||_2 \leq \varepsilon$ or $||\widetilde{x}^{l+1} - \widetilde{x}^{l}||_2 \leq \varepsilon$ with ε being a small positive number). Besides these two criteria, a maximum number of iterations is often specified as a supplementary criterion to limit the overall computation budget and/or prevent running into infinite loops.

Note that, in general, the objective function $f(x)$ can be highly nonlinear (e.g., as in neural network optimization problems), and the optima found by numerical methods such as gradient descent are only local optima.

Newton's Method. Gradient descent is a first-order method. Newton's method is an alternative algorithm for solving an unconstrained optimization problem by further leveraging second-order information. To apply Newton's method, we assume that the objective function f is twice differentiable. Intuitively, Newton's method iteratively solves the unconstrained optimization problem by approximating the objective function using a second-order Taylor expansion about the current guess of the solution.

The second-order Taylor expansion of the cost function f about a point x_0 is

$$f(x) \approx f(x_0) + \nabla f(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T H_f(x_0)(x - x_0), \tag{3.16}$$

where $H_f(x_0)$ denotes the Hessian of f evaluated at x_0. We can find the minimum of the second-order approximation (3.16) by taking the derivative with respect to x and setting it to zero:

$$x^* = x_0 - H_f^{-1}(x_0)\nabla^T f(x_0). \tag{3.17}$$

Note that x^* is a minimum (of the approximated problem) only if the Hessian matrix $H_f(x_0)$ is positive definite.

Similar to gradient descent, let $\widetilde{x}^0 = x_0$ be an initial guess of the optimizer for the unconstrained problem. Newton's method iteratively updates the solution $\widetilde{x}^l$ as follows:

$$\widetilde{x}^{l+1} \longleftarrow \widetilde{x}^{l} - \alpha H_f^{-1}(\widetilde{x}^{l})\nabla^T f(\widetilde{x}^{l}), \tag{3.18}$$

where $\alpha > 0$ is the step size that is often determined through line search methods (Remark 3.12). The algorithm repeats until a stopping criterion is met (Remark 3.13).

Note that the solution to the second-order approximation does not involve α. A line search on α ensures that, at each iteration, the new iterate is at least as good as the previous iterate. In the case of Newton's method, the line search direction is given by $v^l = -H_f^{-1}(\widetilde{x}^{l})\nabla^T f(\widetilde{x}^{l})$.

As compared to (3.14), Netwon's method effectively scales the update along each dimension of x based on the inverse of the Hessian matrix. Newton's method converges faster than gradient descent because of the additional second-order information. Generally, Newton's method has quadratic convergence if we start close enough to a stationary point x^*, whereas the gradient descent algorithms typically have sublinear or linear convergence rates depending on the properties of the objective function. That being said, computing the inverse Hessian matrix in practice can be expensive, and there exist approximations to speed up computation (see, for example, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm).

</> See Jupyter notebook 3.1 example 1.1 for the examples of unconstrained optimization.

3.4 Constrained Optimization

In this section, we start with equality-constrained problems and then derive first-order optimality conditions for the general constrained optimization problem in (3.2).

3.4.1 Optimization with Equality Constraints

We first consider a problem with a single equality constraint and derive an optimality condition for this simplified problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & h(x) = 0, \end{aligned} \tag{3.19}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective, and $h : \mathbb{R}^n \mapsto \mathbb{R}$ is the constraint function. We show that the gradient $\nabla f(x)$ and the gradient $\nabla h(x)$ must be collinear at the location of a minimum:

$$\nabla f(x^*) + \lambda \nabla h(x^*) = 0, \tag{3.20}$$

where $\lambda \in \mathbb{R}$. An illustration is shown in Figure 3.1.

To see why an optimum of the equality-constrained problem needs to satisfy (3.20), we make two observations. For convenience, we define $\mathbb{S} = \{x \mid h(x) = 0\}$ as the set of points x satisfying the equality constraint and consider two points $x^* \in \mathbb{S}$ and $x = x^* + \delta x \in \mathbb{S}$, where δx is an infinitesimal change in x along the curve $h(x) = 0$.

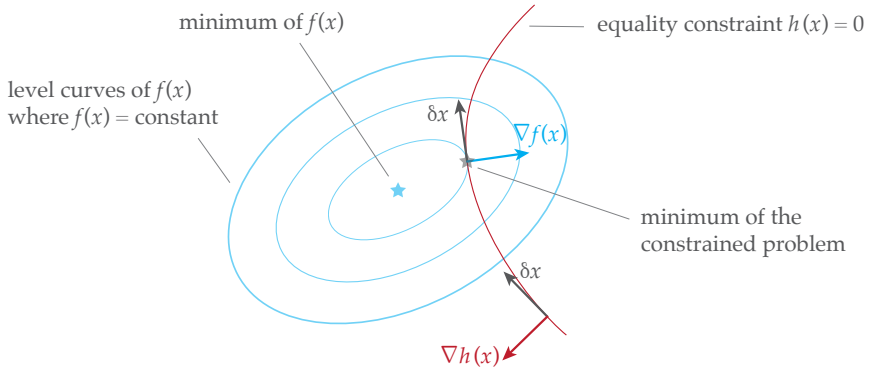


Figure 3.1: An illustration of the stationarity condition for the equality-constrained problem in (3.19). The ellipses in blue are the level curves of $f(x)$, and the curve in red shows the equality constraint. The blue star is the minimum of the unconstrained problem, while the grey star is the minimum of the constrained problem. For the unconstrained problem, the optimum needs to satisfy the condition that $\nabla f(x^*) = 0$; analogously, for the equality-constrained problem, the optimum needs to satisfy $\nabla f(x^*) + \lambda \nabla h(x^*) = 0$, which means that $\nabla f(x)$ is collinear with $\nabla h(x)$ at $x = x^*$.

- By performing a first-order Taylor expansion about x^*, we can show that

$$h(x) - h(x^*) \approx \nabla h(x^*)\, \delta x. \tag{3.21}$$

  As $x^*, x \in \mathbb{S}$, we have $h(x) - h(x^*) = 0$ and thus

$$\nabla h(x^*)\, \delta x \to 0 \quad \text{as} \quad x \to x^*. \tag{3.22}$$

  This implies that to have x staying on the line $h(x) = 0$ (i.e., satisfying the equality constraint), we must move it in a direction that is orthogonal to $\nabla h(x^*)$. In general, for a point to stay on a level curve (e.g., $h(x) = c$ with c being a constant), it must move in a direction that is tangent to the level curve, which is orthogonal to the gradient of the function evaluated at the point (i.e., $\nabla h(x)$).

- Suppose x^* is a local minimum of the constrained problem. Then,

$$\nabla f(x^*)\, \delta x \to 0 \quad \text{as} \quad x \to x^*. \tag{3.23}$$

  Intuitively, if $\nabla f(x^*)$ and δx are not orthogonal, then we can move along $h(x) = 0$ towards the greatest descent direction $-\nabla f(x^*)$ to find a lower $f(x)$, which contradicts the fact that $f(x^*)$ is optimal. We only stop when the two directions are orthogonal.

Since δx is an arbitrary infinitesimal step along the curve $h(x) = 0$, by combining

(3.22) and (3.23), we conclude that $\nabla f(x)$ and $\nabla h(x)$ must be collinear at an optimum x^*, which leads to (3.20). This is a condition that needs to be satisfied by an optimizer of the constrained problem (3.19). Note that λ is often referred to as the Lagrange multiplier.

Motivated by the fact above, we define the Lagrangian,

$$\Lambda(x, \lambda) = f(x) + \lambda h(x), \tag{3.24}$$

associated with the constrained problem (3.19). Let (x^*, λ^*) be a critical point of the Lagrangian $\Lambda(x, \lambda)$. The critical point of the Lagrangian $\Lambda(x, \lambda)$ satisfies

$$\left.\frac{\partial \Lambda}{\partial \lambda}\right|_{x=x^*,\, \lambda=\lambda^*} = h(x^*) = 0, \tag{3.25}$$

$$\left.\frac{\partial \Lambda}{\partial x}\right|_{x=x^*,\, \lambda=\lambda^*} = \nabla f(x^*) + \lambda^* \nabla h(x^*) = 0. \tag{3.26}$$

The condition in (3.25) is the feasibility condition of the equality-constrained problem, and the condition in (3.26) is the stationarity condition we derived above. In general, critical points of the Lagrangian are critical points of the constrained problem (3.19).

We can extend the above result to a more general case where we have multiple equality constraints (i.e., when $h : \mathbb{R}^n \mapsto \mathbb{R}^N$).

Theorem 3.11 (Method of Lagrange Multipliers)**.** The critical points of the equality-constrained optimization problem are the critical points of the Lagrangian

$$\Lambda(x, \lambda) = f(x) + \lambda^T h(x) = f(x) + \sum_{i=1}^{N} \lambda_i h_i(x) \tag{3.27}$$

with respect to both x and λ, where $h(x) = [h_1(x), h_2(x), ..., h_N(x)]^T$ is the vector of equality constraints and $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]^T$ is the Lagrange multiplier vector.

Remark 3.14 (Method of Lagrange Multipliers)**.** By applying Theorem 3.11, we can convert an equality-constrained problem into an unconstrained problem, and any unconstrained solver can be used to find the minima.

3.4.2 General Constrained Optimization

We now return to the general constrained optimization problem in (3.2) and introduce the corresponding optimality conditions. Analogous to the equality-constrained problem, we define the Lagrangian of the general constrained problem as follows:

$$\Lambda(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x) = f(x) + \sum_{i=1}^{N} \lambda_i h_i(x) + \sum_{j=1}^{M} \mu_j g_j(x), \quad (3.28)$$

where $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]^T$ and $\mu = [\mu_1, \mu_2, ..., \mu_M]^T$ are the Lagrange multiplier vectors associated with equality and inequality constraints, respectively.

Primal Feasibility. Let x^* be a critical point of the constrained problem in (3.2). Then, x^* has to satisfy all constraints:

$$h_i(x^*) = 0, \quad \forall i = \{1, 2, ..., N\}, \quad (3.29)$$

$$g_j(x^*) \leq 0, \quad \forall j = \{1, 2, ..., M\}. \quad (3.30)$$

Complementary Slackness. For a general constrained problem, not all inequality constraints are active (see Figure 3.2). We can account for this fact by adding the following condition:

$$\mu_j g_j(x) = 0, \quad \forall j = \{1, 2, ..., M\}. \quad (3.31)$$

This condition requires either $g_j(x)$ or μ_j being zero. Intuitively, this implies that *(i)* if $g_j(x)$ is active (i.e., $g_j(x^*) = 0$), then we treat it like an equality constraint, and *(ii)* otherwise, if $g_j(x)$ is inactive (i.e., $g_j(x^*) < 0$), then, we effectively ignore the inequality constraint by letting $\mu_j = 0$.

Stationarity. As motivated in the previous subsection, a critical point of the constrained problem needs to satisfy the stationary condition. For the general constrained problem, the corresponding stationarity condition is

$$\nabla f(x^*) + \sum_{i=1}^{N} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{M} \mu_j \nabla g_j(x^*) = 0. \quad (3.32)$$

Dual Feasibility. In addition to the conditions above, we also need to specify the sign of the Lagrange multipliers associated with the inequality constraints. If x^* lies on the boundary of an inequality constraint (i.e., $g_j(x^*) = 0$), then we expect that there is a way to further reduce f by violating the constraint $g_j(x) \leq 0$. Since f decreases in the direction of $-\nabla f(x^*)$ and g increase in the direction of $\nabla g(x^*)$,

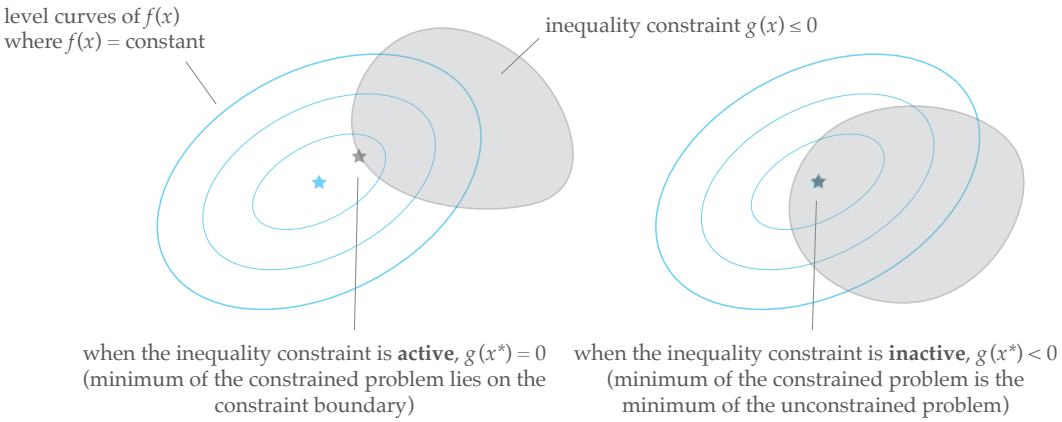


Figure 3.2: An illustration of active versus inactive inequality constraints. In general, not all inequality constraints are active (i.e., influencing the value of the optimum for the constrained problem). We thus differentiate the two cases: *(i)* when an inequality constraint is active, $g(x^*) = 0$ (the minimum of the constrained problem lies on the boundary of the inequality constraint), and *(ii)* when the inequality constraint is inactive, $g(x^*) < 0$ (the minimum of the constrained problem is simply the minimum of the unconstrained problem).

violating the constraint while decreasing f means

$$\nabla f(x^*) \nabla^T g_j(x^*) \leq 0. \tag{3.33}$$

This condition tells us that, when the optimum of the unconstrained problem lies outside the set $g_j(x) \leq 0$ (i.e., the constraint is active), the gradients $\nabla f(x)$ and $\nabla g(x)$ face opposite directions at the optimum $x = x^*$ (i.e., we can further move x^* to reduce f by moving along the direction $-\nabla f(x^*)$ if we do not care about increasing the constraint value or violating the constraint).

On the other hand, by rearranging (3.32) and taking out the inactive inequality constraints, we have

$$-\nabla f(x^*) = \sum_{i=1}^{N} \lambda_i \nabla h_i(x^*) + \sum_{j \in \mathbb{A}} \mu_j \nabla g_j(x^*), \tag{3.34}$$

where $\mathbb{A}$ indicates the set of active inequality constraints. We then replace $h_i(x) = 0$ by $h_i(x) \leq 0$ and $-h_i(x) \leq 0$ and augment the inequality constraints to $g(x)$:

$$-\nabla f(x^*) = \sum_{j \in \mathbb{A}^+} \mu_j \nabla g_j(x^*), \tag{3.35}$$

where $\mathbb{A}^+$ indicates the set of active inequality constraints augmented with the ones corresponding to the equality constraints. Geometrically, $-\nabla f(x^*)$ represents

the direction of steepest descent. For x^* to be a local minimum, there cannot exist any feasible descent direction. Consider an infinitesimal step δx originating from x^*. To remain feasible with respect to all constraints in our augmented active set, this step must not point outward from any boundary, meaning $\nabla g_j(x^*)^T \delta x \leq 0$ for all $j \in \mathbb{A}^+$. By taking the inner product of our augmented stationarity condition with this feasible step δx, we obtain

$$\underbrace{-\nabla f(x^*)^T \delta x}_{\leq 0} = \sum_{j \in \mathbb{A}^+} \mu_j \underbrace{\nabla g_j(x^*)^T \delta x}_{\leq 0}. \tag{3.36}$$

If we move in the feasible direction δx, the objective function cannot strictly decrease (due to the optimality of x^*). Therefore, the directional derivative must be non-negative (i.e., $\nabla f(x^*)^T \delta x \geq 0$), which implies $-\nabla f(x^*)^T \delta x \leq 0$. Under appropriate regularity conditions (e.g., linear independence of the active constraint gradients at x^*), for the sum on the right side of the equation to be guaranteed non-positive for any valid feasible step δx, every multiplier μ_j must be non-negative. Intuitively, if any μ_j were negative, there would exist a valid step δx that decreases the objective function without violating constraints, which contradicts the optimality of x^*. Therefore, all multipliers in the augmented set must satisfy $\mu_j \geq 0$.

Note that, for the original equality constraints, combining the non-negative multipliers of $h_i(x) \leq 0$ and $-h_i(x) \leq 0$ yields a net multiplier $\lambda_i = \mu_i^+ - \mu_i^-$, which has an unconstrained sign. Moreover, for the inactive inequality constraints (i.e., $j \notin \mathbb{A}$), complementary slackness requires $\mu_j = 0$, which trivially satisfies $\mu_j \geq 0$. Overall, the Lagrangian multipliers associated with the inequality constraints of the original constrained problem (3.2) need to satisfy

$$\mu_j \geq 0, \quad \forall j \in \{1, 2, ..., M\}. \tag{3.37}$$

A more formal derivation of the conditions can be found in [12, Ch. 3.1-3.3].

The collection of the conditions above results in the Karush–Kuhn–Tucker (KKT) conditions of the general constrained optimization problem.

Theorem 3.12 (KKT Conditions)**.** Consider the general optimization problem in (3.2). The vector $x^* \in \mathbb{R}^n$ is a critical point for the constrained optimization problem when there exist $\lambda \in \mathbb{R}^N$ and $\mu \in \mathbb{R}^M$ such that

(i) $g(x^*) \leq 0$ and $h(x^*) = 0$, *(primal feasibility)*

(ii) $(\forall j)\ \mu_j g_j(x^*) = 0$, *(complementary slackness)*

(iii) $\nabla f(x^*) + \sum_{i=1}^{N} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{M} \mu_j \nabla g_j(x^*) = 0$, and *(stationarity)*

(iv) $(\forall j)\ \mu_j \geq 0$. *(dual feasibility)*

Remark 3.15 (Optimality Conditions of a General Constrained Problem)**.** A critical point of any constrained optimization problem with differentiable f, h_i, and g_j must satisfy the KKT conditions. One way of determining whether x^* is a local minimum (not a maximum or a saddle point) is to compute the Hessian of f over the subspace of $\mathbb{R}^n$ in which x can move without violating the constraints.

Example 3.11 (KKT Conditions for Equality-Constrained Quadratic Problem)**.** Consider the following equality-constrained optimization problem:

$$\min_x \quad \frac{1}{2} x^T H x + q^T x + r \tag{3.38a}$$

$$\text{subject to} \quad Ax = b, \tag{3.38b}$$

where $x \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times n}$, $q \in \mathbb{R}^n$, $r \in \mathbb{R}$, $A \in \mathbb{R}^{N \times n}$, and $b \in \mathbb{R}^N$. The matrix H is symmetric positive definite.

The Lagrangian for the problem is

$$\Lambda(x, \lambda) = \frac{1}{2} x^T H x + q^T x + r + \underbrace{\sum_{i=1}^{N} \lambda_i (Ax - b)_i}_{=\lambda^T (Ax - b)}, \tag{3.39}$$

where λ_i are the Lagrange multipliers associated with the equality constraints. The KKT conditions for the problem are

$$Ax^* = b \tag{3.40}$$

$$Hx^* + q + A^T \lambda^* = 0 \tag{3.41}$$

with $\lambda^* = [\lambda_1^*, \lambda_2^*, ..., \lambda_N^*]^T$. In matrix form, we have

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}. \tag{3.42}$$

As we discuss in the next section, the problem in (3.38a) is a convex quadratic program, which has a unique optimal solution. The optimal solution can be found by solving the linear system of equations in (3.42).

3.4.3 Convex Optimization

A special class of optimization problems is *convex optimization*. Convex optimization problems are of significant importance because of their property that local optimizers are also global optimal solutions. The optimization problem in (3.2) is convex if the objective is a convex function (Definition 3.16) and the feasible set is a convex set (Definition 3.17).

Definition 3.16 (Convex and Strictly Convex Function)**.** A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *convex* (Figure 3.3) if it satisfies

$$f\big(ax_1 + (1-a)x_2\big) \leq af(x_1) + (1-a)f(x_2), \tag{3.43}$$

where $a \in [0,1]$ and $x_1, x_2 \in \mathbb{R}^n$. The function is *strictly convex*, if the inequality in (3.43) holds with strict inequality for all $\alpha \in (0,1)$ and $x_1 \neq x_2$.

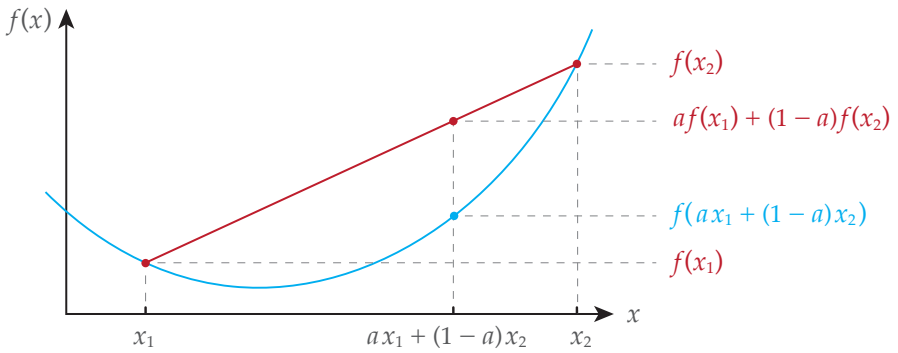


Figure 3.3: Example of a convex function.

Definition 3.17 (Convex Set)**.** A set $\mathcal{X}$ is convex (Figure 3.4) if every point on the line segment connecting two arbitrary points in $\mathcal{X}$ lies in $\mathcal{X}$:

$$ax_1 + (1-a)x_2 \in \mathcal{X}, \tag{3.44}$$

where $a \in [0,1]$ and $x_1, x_2 \in \mathcal{X}$.

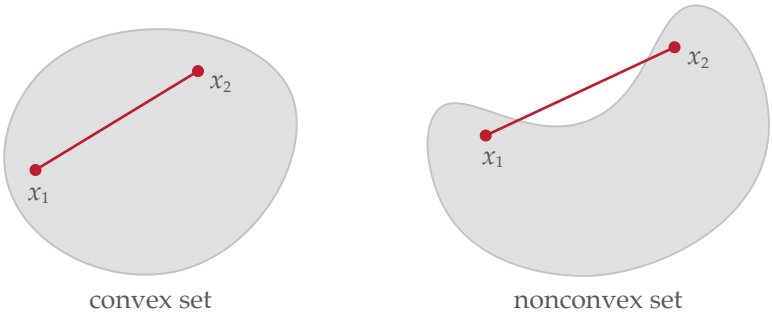


Figure 3.4: Example of a convex set *(left)* and a nonconvex set *(right)*.

Definition 3.18 (Affine Functions)**.** A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ of the form $f(x) = a^T x - b$ is called an affine function, where a and b are parameters with consistent dimensions.

Theorem 3.13 (Convex Feasible Set)**.** The problem in (3.2) has a convex feasible set $\mathcal{X}$ if the equality constraint functions h_i are affine and the inequality constraint functions g_j are convex.

Proof. Consider two points $x_1, x_2 \in \mathcal{X}$. Let $x = \alpha x_1 + (1-\alpha)x_2$ with $\alpha \in [0,1]$. For convexity, we need to show that $x \in \mathcal{X}$. Since the equality constraints are affine, we have $h_i(x) = a_i^T x - b_i = a_i^T(\alpha x_1 + (1-\alpha)x_2) - (\alpha + 1 - \alpha)b_i = \alpha h_i(x_1) + (1-\alpha)h_i(x_2)$. Moreover, since $x_1, x_2 \in \mathcal{X}$, $h_i(x_1) = h_i(x_2) = 0$ and thus $h_i(x) = 0$. Next, since the inequality constraints are convex, we have $g_j(x) = g_j\big(\alpha x_1 + (1-\alpha)x_2\big) \leq \alpha g_j(x_1) + (1-\alpha)g_j(x_2)$. Similarly, since $x_1, x_2 \in \mathcal{X}$, $g_j(x_1) \leq 0$ and $g_j(x_2) \leq 0$, which imply $g_j(x) \leq 0$. These two arguments together show that x is a point in the feasible set, and hence $\mathcal{X}$ is convex. □

Equipped with the result above, we can write the standard form of a convex optimization problem as follows:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & a_i^T x - b_i = 0, \quad \forall\, i = \{1, 2, ..., N\}, \\ & g_j(x) \leq 0, \quad \forall\, j = \{1, 2, ..., M\}, \end{aligned} \tag{3.45}$$

where $x \in \mathbb{R}^n$ is the optimization variable of the problem, $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g_j : \mathbb{R}^n \mapsto \mathbb{R}$ are convex functions, and $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are parameters associated

with the i-th equality constraint. As hinted earlier, convex optimization problems possess unique properties, which we now introduce more formally below.

Theorem 3.14 (Global Optimality of Solutions to Convex Problems)**.** If f is convex and the feasible set $\mathcal{X}$ is convex, then any local minimum is also a global minimum.

Proof. Suppose $x_1^* \in \mathcal{X}$ is a local minimum. By definition, this means that for every point $x \in \mathcal{X}$ in the neighbourhood of x_1^*, we have $f(x_1^*) \leq f(x)$. We further assume that there exists a global minimum $x_2^* \in \mathcal{X}$ such that $f(x_2^*) < f(x_1^*)$. Consider a point $x = \alpha x_1^* + (1-\alpha)x_2^*$ with $\alpha \in (0,1)$ being sufficiently close to 1 such that x is in the neighbourhood of x_1^*. Since the set $\mathcal{X}$ is convex, it follows that $x \in \mathcal{X}$. By convexity of the objective function, we have $f(x) \leq \alpha f(x_1^*) + (1-\alpha)f(x_2^*)$. Furthermore, since $f(x_2^*) < f(x_1^*)$, we have $f(x) < \alpha f(x_1^*) + (1-\alpha)f(x_1^*) = f(x_1^*)$, which leads to a contradiction that x_1^* is a local minimum. Thus, by contradiction, this shows that if a point x_1^* is a local minimum, it is also a global minimum. □

The convexity of an optimization problem does not always mean that the problem is well-posed (i.e., the existence of a solution). For instance, with a linear objective function $f(x) = x$ or an exponential objective function $f(x) = \exp(x)$ subject to a linear constraint $g(x) = x \leq 0$, the problem is either not bounded from below or does not attain a minimum over the feasible set. Moreover, when a solutions exist, it may not be unique. As a trivial example, the constant $f(x) = 0$ would admit an infinite number of optimal solutions. In general, the more structure we know about the objective function and the feasible set, the more we can conclude about the properties of the optimal solution(s) in a convex optimization problem.

Theorem 3.15 (Uniqueness of the Global Minimum for Strictly Convex Problems)**.** If f is strictly convex and the feasible set $\mathcal{X}$ is convex, then the optimization problem possesses at most one (global) minimum.

Proof. Suppose $x_1^*, x_2^* \in \mathcal{X}$ are both (global) minima of the objective function $f(x)$ in the feasible set $\mathcal{X}$ and $x_1^* \neq x_2^*$. Consider a point $x = \alpha x_1^* + (1-\alpha)x_2^*$ with $\alpha \in (0,1)$. Since the set $\mathcal{X}$ is convex, it follows that $x \in \mathcal{X}$. By strict convexity,

we have $f(x) = f\big(ax_1^* + (1-a)x_2^*\big) < af(x_1^*) + (1-a)f(x_2^*)$. Since both x_1^* and x_2^* are (global) minima of the constrained problem, $f(x_1^*) = f(x_2^*)$. This implies $f(x) < af(x_1^*) + (1-a)f(x_2^*) = f(x_1^*) = f(x_2^*)$, which contradicts the fact that x_1^* and x_2^* are (global) minima. Thus, by contradiction, if x_1^* is a (global) minimum, it is the unique (global) optimum in the feasible set $\mathcal{X}$. □

Remark 3.16 (Existence and Uniqueness of Solutions to Convex Problems)**.** From Theorem 3.14, we know that all local minimizers of convex problems are global minimizers. Theorem 3.15 further shows that if the objective function is strictly convex, then any existing minimizer is the unique global minimizer. However, the existence of a solution is not guaranteed solely by the convexity or strict convexity of the problem (see the counterexamples above Theorem 3.15). Additional conditions (e.g., the compactness of the feasible set or the radial unboundedness of the objective) are needed to ensure that the problem is well-posed and a minimum over the feasible set exists.

There are different subclasses of convex optimization problems. Below, we introduce two common types for which solutions can be obtained efficiently in practice.

Linear Program (LP). The optimization problem in (3.2) is a LP if the objective and the constraint functions are linear in x:

$$\begin{aligned} \min_x \quad & c^T x + d && (3.46)\\ \text{subject to} \quad & Ax = b && (3.47)\\ & Gx \preceq w, && (3.48) \end{aligned}$$

where $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $A \in \mathbb{R}^{N\times n}$, $b \in \mathbb{R}^N$, $G \in \mathbb{R}^{M\times n}$, $w \in \mathbb{R}^M$, and "$\preceq$" denotes the generalized inequality with respect to the non-negative orthant (or, elementwise comparison). All LPs are convex optimization problems; any existing solution(s) to an LP are global minima.

Quadratic Program (QP). The optimization problem in (3.2) is a QP if the objective function is quadratic in x and the constraint functions are linear in x:

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T Hx + q^T x + r && (3.49)\\ \text{subject to} \quad & Ax = b && (3.50)\\ & Gx \preceq w, && (3.51) \end{aligned}$$

where $x \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times n}$, $q \in \mathbb{R}^n$, $r \in \mathbb{R}$, $A \in \mathbb{R}^{N \times n}$, $b \in \mathbb{R}^N$, $G \in \mathbb{R}^{M \times n}$, and $w \in \mathbb{R}^M$. The QP is convex if H is symmetric positive definite. In this case, if the feasible set is not empty, then the QP admits a unique optimal solution.

</> See Jupyter notebook 3.1 example 2.1 for an example configuration of QP solver.

Example 3.12 (Convex Problem Reformulations)**.** Consider a slightly modified mobile robot goal-reaching problem:

$$\min_{u_0,\, u_1} \quad \sum_{k=1}^{2} |u_k| \tag{3.52a}$$

$$\text{subject to} \quad x_{k+1} = x_k + u_k, \quad \forall k \in \{0, 1\} \tag{3.52b}$$

$$x_0 = \bar{x}_0, x_2 = x_g \tag{3.52c}$$

$$-1 \leq u_k \leq 1, \quad \forall k \in \{0, 1\}, \tag{3.52d}$$

where $x_k \in \mathbb{R}$ is the robot position, $u_k \in \mathbb{R}$ is the velocity input, $\bar{x}_0 \in \mathbb{R}$ is the initial position, and $x_g \in \mathbb{R}$ is the desired goal position at the final time step. As compared to our earlier setup, the quadratic cost function is replaced by the ℓ_1-norm of the vector of inputs over the task horizon (i.e., $||\mathfrak{u}||_1 = \sum_{k=0}^{1} |u_k|$ with $\mathfrak{u} = [u_0, u_1]^T$), and the goal state is now enforced as a hard constraint. We can reformulate this problem as an LP. For convenience, we rewrite the constraints in (3.52b)-(3.52c) and in (3.52d), respectively, as follows:

$$A\mathfrak{u} = b \quad \text{with} \quad A = \mathbf{1}_2^T \quad \text{and} \quad b = x_g - \bar{x}_0, \tag{3.53a}$$

$$G_a\mathfrak{u} \preceq w_a \quad \text{with} \quad G_a = \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \quad \text{and} \quad w_a = \mathbf{1}_4, \tag{3.53b}$$

where $\mathfrak{u} = [u_0, u_1]^T$ is the augmented input vector, $I_s \in \mathbb{R}^{s \times s}$ denotes an identity matrix, and $\mathbf{1}_s \in \mathbb{R}^s$ denotes a column vector of ones. The constraint in (3.53a) can be derived by "simulating" the dynamics forward: $x_1 = \bar{x}_0 + u_0$ and $x_g = \bar{x}_0 + u_0 + u_1$. The constraint (3.53b) is the augmentation of the lower and upper input constraints with $G_a\mathfrak{u} = [u_0, u_1, -u_0, -u_1]^T$ capturing the input variables to be upper- or lower-bounded at each time step and G_a defining the values of the upper and lower bounds.

The optimization problem in (3.52) can be equivalently written as

$$\begin{aligned} \min_{u,\,q} \quad & \mathbf{1}_2^T q && \text{(3.54a)} \\ \text{subject to} \quad & Au = b && \text{(3.54b)} \\ & G_a u \preceq w_a && \text{(3.54c)} \\ & |u_i| = q_i, \quad \forall i \in \{0,1\}, && \text{(3.54d)} \end{aligned}$$

where $q = [q_0, q_1]^T$ is an auxiliary variable, and the subscript i denotes the i-th component of the vector. By noting that the constraint in (3.54d) can be replaced by two inequality constraints $u \preceq q$ and $-u \preceq q$ (or, $u_i \leq q_i$ and $-u_i \leq q_i$ for $i = \{0,1\}$), we have the following equivalent LP for the original problem in (3.52):

$$\begin{aligned} \min_{u,\,q} \quad & \mathbf{1}_2^T q && \text{(3.55a)} \\ \text{subject to} \quad & Au = b && \text{(3.55b)} \\ & G_a u \preceq w_a && \text{(3.55c)} \\ & G_b u \preceq w_b, && \text{(3.55d)} \end{aligned}$$

where

$$G_b = \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \quad \text{and} \quad w_b = \begin{bmatrix} q \\ q \end{bmatrix}. \tag{3.56}$$

Now, what if the cost function in (3.52) is replaced by the infinity norm of u (i.e., $||u||_\infty = \max_i |u_i|$)? We can similarly introduce an auxiliary variable $q \in \mathbb{R}$ to help us rewrite the problem as an LP:

$$\begin{aligned} \min_{u,\,q} \quad & q && \text{(3.57a)} \\ \text{subject to} \quad & Au = b && \text{(3.57b)} \\ & G_a u \preceq w_a && \text{(3.57c)} \\ & \max_i |u_i| \leq q, \quad \forall i \in \{0,1\}. && \text{(3.57d)} \end{aligned}$$

Note that the maximization in (3.57d) can be enforced by $|u_i| \leq q$ for all i. Intuitively, this means that, if all elements satisfy the upper bound, then the maximum value of the elements also satisfies the upper bound. Analogous to the l_1-norm minimization case, we can further replace the absolute operator with two

inequality constraints to arrive at the following LP:

$$\begin{aligned} \min_{u,\,q} \quad & q && \text{(3.58a)} \\ \text{subject to} \quad & Au = b && \text{(3.58b)} \\ & G_a u \preceq w_a && \text{(3.58c)} \\ & G_b u \preceq w_b, && \text{(3.58d)} \end{aligned}$$

where G_b is the same as that in (3.56) and $w_b = q\mathbf{1}_4$.

3.4.4 Algorithms for Solving Constrained Optimization Problems

In this subsection, we introduce two algorithms for solving constrained optimization problems.

Barrier Methods

One approach to solving the optimization problem is through barrier methods. The idea is to incorporate constraints into the objective function as barrier or penalty functions and perform unconstrained optimization. As an example, for an inequality-constrained problem, we could minimize an "augmented" objective function as follows:

$$f_\rho(x) = f(x) - \rho \log\big(-g(x)\big). \tag{3.59}$$

Notice that for a strictly feasible point, $g(x) < 0$, which makes $-g(x)$ a positive number. As the solution approaches the constraint boundary ($g(x) \to 0^-$), the term $-\log\big(-g(x)\big)$ approaches $+\infty$, creating a steep penalty that prevents the optimizer from leaving the feasible set. A minimizer of $f_\rho(x)$ gets arbitrarily close to a minimizer of the constrained problem when the barrier parameter $\rho \to 0$.

Barrier methods turn a constrained problem into an augmented unconstrained problem and iteratively solve the unconstrained problem. In an interior-point method, the initial point must be strictly feasible. In each iteration, the barrier parameter ρ is decreased, which allows the optimal solution of the subproblem to approach the constraint boundary more closely. This approach works naturally for inequality constraints (see Figure 3.5); however, when ρ is small, the Hessian of f_ρ can become increasingly ill-conditioned.

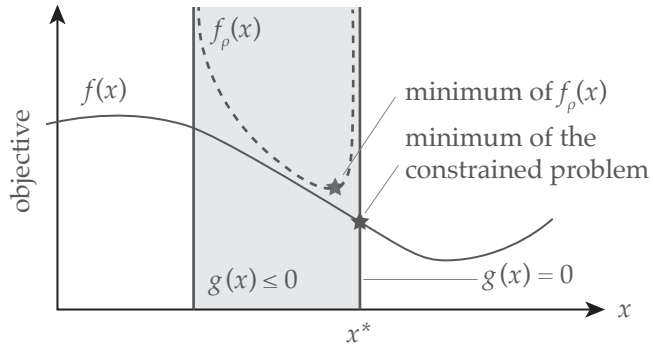


Figure 3.5: An illustration of barrier methods for an inequality-constrained problem. The original inequality-constrained problem with objective $f(x)$ and inequality constraint $g(x) \le 0$ is turned into an unconstrained problem where we minimize a modified objective function $f_\rho(x)$ that includes an additional term penalizing constraint violations.

Sequential Quadratic Programming (SQP)

Another approach for solving the general nonlinear optimization problem in (3.2) is to iteratively approximate the nonlinear problem by a QP. We briefly summarize the key steps of a typical SQP algorithm below and provide further explanations of its individual components thereafter:

- Initialization: An initial guess of the optimal solution $\widetilde{x}^0$ (e.g., a point in the feasible set, or alternatively, the solution to the unconstrained problem) and initial guesses of the Lagrange multipliers $(\widetilde{\lambda}^0, \widetilde{\mu}^0)$:

The superscripts of $\widetilde{x}$ denote the iteration number.

- Recursion $l = \{0, 1, ...\}$:

  1) Second-order Taylor expansion of the Lagrangian $\Lambda(x, \widetilde{\lambda}^l, \widetilde{\mu}^l)$ about $\widetilde{x}^l$
  2) First-order Taylor expansion of the constraints $h_i(x)$ and $g_j(x)$ about $\widetilde{x}^l$
  3) Solve the resulting QP and obtain values of the new iterate:

$$\widetilde{x}^{l+1} \longleftarrow \widetilde{x}^l + \alpha\delta\widetilde{x}^l, \quad \widetilde{\lambda}^{l+1} \longleftarrow \widetilde{\lambda}^l + \alpha\delta\widetilde{\lambda}^l, \quad \text{and} \quad \widetilde{\mu}^{l+1} \longleftarrow \widetilde{\mu}^l + \alpha\delta\widetilde{\mu}^l, \tag{3.60}$$

     where $(\delta\widetilde{x}^l, \delta\widetilde{\lambda}^l, \delta\widetilde{\mu}^l)$ are the update directions of the primal and dual variables determined based on the solution to the QP subproblem, and α is the step size that is usually determined through a line search based on a merit function $\phi_\rho(x, \lambda, \mu)$
  4) Repeat till convergence (termination condition based on optimal value or optimal solution)

If the nonlinear optimization problem in (3.2) is convex, then $\widetilde{x}^l$ converges to a

local/global minima x^* as the number of iterations l tends to infinity; if the problem is nonconvex, then under some assumptions (e.g., a sufficiently good initial guess and specific properties of the Hessian in the approximated subproblem), it can be shown that $\widetilde{x}^l$ converges to a local minimum x^* [13].

QP Subproblem Formulation. The SQP method offers an intuitive approach to solving nonlinear programs by iteratively approximating them with simpler QP subproblems. A central challenge, however, lies in designing appropriate subproblem approximations. A naive approach is to perform a second-order Taylor expansion of the objective and a first-order Taylor expansion of the constraints. While straightforward, this can result in unbounded quadratic subproblems. A more common variant of SQP uses, instead, a second-order approximation of the Lagrangian to better account for the nonlinear nature of the constraints:

$$\min_{\delta\widetilde{x}^l} \quad \frac{1}{2}(\delta\widetilde{x}^l)^T H_\Lambda(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l)\delta\widetilde{x}^l + \nabla\Lambda(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l)\delta\widetilde{x}^l \tag{3.61a}$$

$$\text{subject to} \quad \nabla h_i(\widetilde{x}^l)\delta\widetilde{x}^l + h_i(\widetilde{x}^l) = 0, \quad \forall\, i = \{1, 2, ..., N\}, \tag{3.61b}$$

$$\nabla g_j(\widetilde{x}^l)\delta\widetilde{x}^l + g_j(\widetilde{x}^l) \leq 0, \quad \forall\, j = \{1, 2, ..., M\}, \tag{3.61c}$$

where $\delta\widetilde{x}^l = x - \widetilde{x}^l$ is the decision variable, $H_\Lambda(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l)$ is the Hessian of the Lagrangian with respect to x evaluated at the current iterate, and $\nabla\Lambda(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l)$, and $\nabla h_i(\widetilde{x}^l)$, $\nabla g_j(\widetilde{x}^l)$ are, respectively, the gradients of the Lagrangian, the equality constraint functions, and the inequality constraint functions with respect to x evaluated at the current iterate. While the two formulations, approximating the objective or the Lagrangian, are equivalent in the case of linear constraints, the Lagrangian-based formulation often improves the numerical stability of the algorithm in more complex, nonlinear constraint settings. The convergence of the SQP algorithm depends on how the sequence of Hessian matrices is generated in the sequence of QP subproblems [13]; intuitively, the Hessian of the Lagrangian offers a better approximation of the original constrained optimisation problem as compared to the Hessian of the objective function alone.

A more commonly seen SQP formulation, which builds on (3.61), is given by

$$\min_{\delta\widetilde{x}^l} \quad \frac{1}{2}(\delta\widetilde{x}^l)^T H_\Lambda(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l)\delta\widetilde{x}^l + \nabla f(\widetilde{x}^l)d \tag{3.62a}$$

$$\text{subject to} \quad \nabla h_i(\widetilde{x}^l)\delta\widetilde{x}^l + h_i(\widetilde{x}^l) = 0, \quad \forall\, i = \{1, 2, ..., N\}, \tag{3.62b}$$

$$\nabla g_j(\widetilde{x}^l)\delta\widetilde{x}^l + g_j(\widetilde{x}^l) \leq 0, \quad \forall\, j = \{1, 2, ..., M\}, \tag{3.62c}$$

which is equivalent to (3.61) in the absence of inequality constraints. An assumption that justifies this formulation is that the set of active inequality constraints is known in advance, which allows us to treat the active constraints as equality constraints and ignore the inactive ones.

Lagrange Multipliers. The QP subproblem approximation, in particular, the Hessian matrix relies on the knowledge of the Lagrange multipliers of the original constrained optimization problem. One approach to approximate the Lagrange multipliers at each iteration is to update them based on the optimal Lagrange multipliers of the QP subproblem $(\widetilde{\lambda}^l_{\text{qp}}, \widetilde{\mu}^l_{\text{qp}})$:

$$\widetilde{\lambda}^{l+1} \longleftarrow \widetilde{\lambda}^l + \alpha \delta \widetilde{\lambda}^l \quad \text{with} \quad \delta \widetilde{\lambda}^l = \widetilde{\lambda}^l_{\text{qp}} - \widetilde{\lambda}^l \tag{3.63a}$$

$$\widetilde{\mu}^{l+1} \longleftarrow \widetilde{\mu}^l + \alpha \delta \widetilde{\mu}^l \quad \text{with} \quad \delta \widetilde{\mu}^l = \widetilde{\mu}^l_{\text{qp}} - \widetilde{\mu}^l. \tag{3.63b}$$

Besides the update rules above, there exist other variants that are formulated to achieve better theoretic convergence properties in the proximity of the optimal solution.

Line Search. In addition to leveraging the Lagrangian to better account for constraints in the second-order cost approximation, performing a line search over the step size α can also further improve the convergence of the SQP to the optimal solution. Typically, the step size is optimized based on a merit function ϕ such that the following condition is satisfied:

$$\phi_\rho(\widetilde{x}^{l+1}, \widetilde{\lambda}^{l+1}, \widetilde{\mu}^{l+1}) < \phi_\rho(\widetilde{x}^l, \widetilde{\lambda}^l, \widetilde{\mu}^l) \quad \text{with} \quad \widetilde{x}^{l+1} = \widetilde{x}^l + \alpha \delta \widetilde{x}^l, \tag{3.64}$$

where $\rho > 0$ is a constant parameter. One typical merit function for equality-constrained problem is the augmented Lagrangian function:

$$\phi_\rho(x, \lambda) = f(x) + \sum_{i=1}^{N} \lambda_i h_i(x) + \frac{\rho}{2} \sum_{i=1}^{N} \lambda_i \big(h_i(x)\big)^2, \tag{3.65}$$

which is the Lagrangian augmented with an additional term that penalizes constraint violations. For desired convergence behaviour, the parameter ρ is often chosen to be sufficiently large.

</> See Jupyter notebook 3.1 example 2.3 for an example implementation of SQP solver.

Chapter Summary

Optimization Fundamentals

- **General Optimization Problem:** A generic optimization problem can be written as

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0, \quad \forall\, i = \{1, 2, ..., N\}, \\ & g_j(x) \leq 0, \quad \forall\, j = \{1, 2, ..., M\}, \end{aligned} \tag{3.66}$$

  where $x \in \mathbb{R}^n$ is the optimization variable of the problem, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function, and $h_i : \mathbb{R}^n \mapsto \mathbb{R}$ and $g_j : \mathbb{R}^n \mapsto \mathbb{R}$ are the equality and inequality constraint functions, respectively.

- **Unconstrained Optimization:** For an unconstrained optimization problem, we have two optimality conditions
  - First-order optimality condition (necessary condition): If x^* is a local minimum of $f : \mathbb{R}^n \mapsto \mathbb{R}$, then x^* is a stationary point with $\nabla f(x^*) = 0$.
  - Second-order optimality condition (sufficient condition): If (and only if) $H_f(x^*)$ is positive definite, then the stationary point x^* is a local minimum of f.

- **Numerical Algorithms for Unconstrained Optimization:** Common numerical methods for solving an unconstrained optimization problem include gradient descent (first-order method) and Newton's method (second-order method). Higher-order methods generally have better convergence properties; however, these methods can be more computationally expensive (e.g., computing the Hessian matrix). There exist approximate approaches in practice to speed up computation.

- **Constrained Optimization:** The Lagrangian for a general constrained optimization problem is

$$\Lambda(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x) = f(x) + \sum_{i=1}^{N} \lambda_i h_i(x) + \sum_{j=1}^{M} \mu_j g_j(x), \tag{3.67}$$

  where $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]^T$ and $\mu = [\mu_1, \mu_2, ..., \mu_M]^T$ are the Lagrange multiplier vectors associated with equality and inequality constraints, respectively. The Karush-Kuhn-Tucker (KKT) conditions provide a set

of necessary optimality conditions for constrained optimization. In particular, the vector $x^* \in \mathbb{R}^n$ is a critical point for the constrained optimization problem when there exist $\lambda \in \mathbb{R}^N$ and $\mu \in \mathbb{R}^M$ such that

(i) $g(x^*) \leq 0$ and $h(x^*) = 0$, *(primal feasibility)*

(ii) $(\forall j)\ \mu_j g_j(x^*) = 0$, *(complementary slackness)*

(iii) $\nabla f(x^*) + \sum_{i=1}^{N} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{M} \mu_j \nabla g_j(x^*) = 0$, and *(stationarity)*

(iv) $(\forall j)\ \mu_j \geq 0$. *(dual feasibility)*

- **Numerical Algorithms for Constrained Optimization:** Numerical algorithms for constrained optimization often require approximations of the original problem into simpler problems (e.g., turning it into unconstrained subproblems or convex subproblems). Two examples are the barrier method and the sequential quadratic program (SQP) approach.

3.5 Exercises

Exercise 3.1 (Short Questions)**.** For the problems below, select the options that best answer each question and justify your selections.

a) What are the advantages of Newton's method compared to gradient descent?

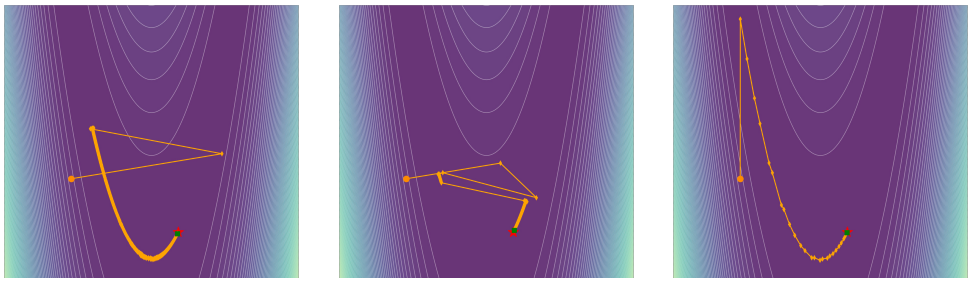
(A) Gradient descent requires computing the Hessian matrix, whereas Newton's method does not.

(B) Newton's method is guaranteed to converge to the global optimum, whereas gradient descent often gets stuck in a local optimum.

(C) In contrast to gradient descent, Newton's method uses not only first-order but also second-order information and, therefore, usually converges faster.

(D) Line search can be used for Newton's method but not for gradient descent.

b) Consider an unconstrained optimization problem where the objective is the Rosenbrock function. We solve this problem using three methods: *(i)* gradient descent with a smaller initial step size, *(ii)* gradient descent with a larger initial step size, and *(iii)* Newton's method. All algorithms are initialized from the same starting point (orange dot), but exhibit different behaviours as they converge towards the optimum (red star). Based on the plots below, which of the following convergence trajectories corresponds to Newton's method?



(A) (B) (C)

c) Consider a generic differentiable function $f(x)$. Which of the following statements are correct?

(A) The function may possess multiple local minima.

(B) If local minima exist, they are also global minima.

(C) If the function has a stationary point x where $\nabla f(x) = 0$ and $H_f(x) \succ 0$, then x is a local minimum.

(D) Even if the function possesses a global minimum, it is never possible to determine it.

d) Consider an optimization problem where the objective is to minimize a function $f(x)$ subject to a set of N equality constraints $h_i(x) = 0$ for $i = \{1, 2, \ldots, N\}$. Which of the following statements are correct?

(A) We can convert the equality-constrained problem into an unconstrained problem using the method of Lagrange multipliers.

(B) The Lagrangian can be written as $\Lambda(x, \lambda) = f(x) + \sum_{i=1}^{N} \lambda_i h_i(x)$, where λ_i is the Lagrange multiplier associated with the i-th equality constraint.

(C) The optimal solution (x^*, λ^*) is a stationary point of the Lagrangian.

(D) The equality constraints may or may not be satisfied using the method of Lagrange multipliers.

e) Which of the problems below is convex?

(A) $\begin{aligned} \min_{x \in \mathbb{R}} \quad & \exp(x) \\ \text{subject to} \quad & x^2 \geq 1 \end{aligned}$

(B) $\begin{aligned} \min_{x \in \mathbb{R}} \quad & \exp(x) \\ \text{subject to} \quad & x^2 = 1 \end{aligned}$

(C) $\begin{aligned} \min_{x \in \mathbb{R}} \quad & x^3 \\ \text{subject to} \quad & x^2 \leq 1 \end{aligned}$

(D) $\begin{aligned} \min_{x \in \mathbb{R}} \quad & \exp(x) \\ \text{subject to} \quad & x^2 \leq 1 \end{aligned}$

f) Which of the following statements about convex problems are correct?

(A) The set $\mathcal{X} = \{x \mid Ax \leq b\}$ is convex for every matrix A and vector b.

(B) If the objective function is strictly convex and the feasible set is non-convex, the optimization problem is still convex.

(C) Every quadratic objective function $f(x) = \frac{1}{2} x^\top H x + q^\top x + r$ is convex.

(D) A convex problem with inequality constraints cannot have more than one optimal solution.

(E) Every local minimum of a convex problem is guaranteed to be a global optimal solution.

g) Consider an optimization problem where the objective is to minimize $f(x)$ subject to equality constraints $h_i(x) = 0$ for $i \in \{1, \ldots, m\}$, where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$, $h_i : \mathbb{R}^n \to \mathbb{R}$. The Lagrangian function is defined as

$\Lambda(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$, where λ_i are the Lagrange multipliers. Select all true statements.

(A) The KKT conditions include the stationarity condition $\nabla_x \Lambda(x^*, \lambda^*) = 0$.

(B) The KKT conditions include the primal feasibility condition $h_i(x^*) = 0$ for all i.

(C) For an equality-constrained quadratic program, the KKT conditions reduce to a system of linear equations.

(D) The KKT conditions include the dual feasibility condition $\lambda_i^* \geq 0$ for all i.

h) Select all correct statements about barrier methods in constrained optimization:

(A) Barrier methods solve unconstrained optimization problems by transforming them into constrained ones.

(B) Barrier methods handle constraints by incorporating them into the objective function through a barrier term.

(C) Barrier methods iteratively decrease the penalty until the constraints are satisfied closely enough.

(D) The Hessian of the augmented objective function can be ill-conditioned.

i) Consider the optimization problem $\min_{x \in \mathbb{R}} (x - 0.5)^2$ subject to $x^2 \leq 1$. Which of the following statements are correct?

(A) The inequality constraint is inactive, and the optimal solution is $x = 0.5$.

(B) The optimization problem is convex; thus, any local minimum is also a global minimum.

(C) The inequality constraint is active, and the optimal solution is $x = -1$.

(D) The optimization problem has an infinite number of optimal solutions.

j) Consider an optimal control problem for a nonlinear system, which is solved using SQP. The nonlinear program is defined as follows: minimize $f(x)$ subject to $h(x) = 0$ and $g(x) \leq 0$, where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^N$, and $g : \mathbb{R}^n \to \mathbb{R}^M$. Select all true statements.

(A) SQP solves the nonlinear programming problem by iteratively solving a sequence of QP subproblems.

(B) The QP subproblems in SQP can be derived from a second-order Taylor series expansion of the Lagrangian.

(C) For a nonconvex optimization problem, SQP converges to a local minimum only if combined with line search.

(D) SQP can handle either equality or inequality constraints, but not both.

(E) None of the above

Exercise 3.2 (Constrained Optimization)**.** Solving static constrained minimization problems plays a central role in Optimal Control. In this tutorial, we recall some basic techniques on how to deal with equality and inequality constraints in optimization.

a) Minimize the function $f(x_1, x_2) = 2x_1^2 + x_2^2$ subject to the equality constraint $x_1 + x_2 = 1$.

b) Minimize the function $f(x_1, x_2) = (x_1 - 2)^2 + 2(x_2 - 1)^2$ subject to the inequality constraints $x_1 + 4x_2 \leq 3$ and $x_1 \geq x_2$.

c) Minimize the function $f(x_1, x_2) = (x_1 + 1)^2 + x_2$ subject to the equality constraint $x_1 + x_2 = 4$ and the inequality constraint $x_2 \geq 0$.

Exercise 3.3 (Constrained Open-Loop Optimal Control)**.** Consider the discrete-time system

$$x_{k+1} = 0.5x_k + u_k,$$

with $x_k, u_k \in \mathbb{R}$ for all $k \in \mathbb{Z}$. The initial state is assumed to be $x_0 = 0$, and a constant reference trajectory $r_k = 1$ for all $k \in \mathbb{Z}$ is given. The cost function for a horizon N is

$$J = \sum_{i=1}^{N} (x_k - r_k)^2 .$$

a) Compute the optimal input sequence (u_0^*, u_1^*) that minimizes the cost function $J(u_0, u_1)$ for a horizon $N = 2$. After how many steps does x_k reach the reference trajectory r_k?

b) Now, consider the input constraints $u_0 \leq 0.75$ and $u_1 \leq 0.75$. Solve the constrained minimization problem for u_0^* and u_1^*. How many steps are now required for x_k to reach the reference trajectory r_k?

Exercise 3.4 (Constraint Softening [14])**.**

a) Solve the following minimization problem with a hard constraint:

$$\min_x f(x) \quad \text{subject to} \quad h(x) \leq 0,$$

where $f(x) = x^2 - 6x + 11$ and $h(x) = x - 1$.

b) In practice, sometimes the hard constraints can be softened. The *softened* minimization problem uses an additional penalty term $l(\epsilon)$ and is defined as

$$\min_{x,\epsilon} f(x) + l(\epsilon) \quad \text{subject to} \quad h(x) \leq \epsilon, \quad \epsilon \geq 0,$$

where ϵ defines the degree of softness. The penalty term can be designed, e.g., as a quadratic term defined as $l(\epsilon) = v \cdot \epsilon^2$ with $v > 0$. Solve a) with the quadratic penalty and $v = 1$.

c) If the original problem has a feasible solution x^*, then the softened problem should have the same solution x^* with $\epsilon = 0$. However, the quadratic penalty does not meet this requirement for any $v > 0$. Alternatively, we can define a linear penalty term as $l(\epsilon) = u \cdot \epsilon$, which satisfies this requirement for $u > \mu^* \geq 0$, where μ^* is the optimal Lagrange multiplier for the original problem. Solve a) with the linear penalty and $u = 5$.

d) The disadvantage of the linear penalty is that it renders the function non-smooth, making numerical optimization difficult. To still retrieve a smooth penalty term in practice, we can combine the linear and quadratic penalty to obtain $l(\epsilon) = u \cdot \epsilon + v \cdot \epsilon^2$ with $u > \mu^* \geq 0$ and $v > 0$. Solve a) with $u = 5$ and $v = 1$.

Bibliography

[1] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: from learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control (4th edition), Volume I*. Athena Scientific, Belmont, MA, 2017.

[3] P. Akella, A. Dixit, M. Ahmadi, L. Lindemann, M. P. Chapman, G. J. Pappas, A. D. Ames, and J. W. Burdick, "Risk-aware robotics: tail risk measures in planning, control, and verification [focus on education]," *IEEE Control Systems*, vol. 45, no. 4, pp. 46–78, 2025.

[4] J. Buchli, F. Farshidian, A. Winkler, T. Sandy, and M. Giftthaler, "Optimal and learning control for autonomous robots," *arXiv preprint arXiv:1708.09342*, 2017.

[5] W. M. Wonham, *Linear Multivariable Control: A Geometric Approach (3rd edition)*. New York, NY: Springer, 1985.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control (4th edition), Volume II*. Athena Scientific, Belmont, MA, 2012.

[7] J. P. Hespanha, *Linear Systems Theory*. Princeton university press, 2018.

[8] M. Papageorgiou, M. Leibold, and M. Buss, *Optimierung*. Springer Berlin Heidelberg, 2015.

[9] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6740–6745.

[10] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarm: an alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1421–1427.

[11] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[13] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[14] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[15] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 2, 2004, pp. 222–229.

[16] T. Samad, "A survey on industry impact and challenges thereof [technical activities]," *IEEE Control Systems Magazine (CSM)*, vol. 37, no. 1, pp. 17–18, 2017.

[17] A. Bemporad, "Model predictive control," 2021, lecture slides. Available at: http://cse.lab.imtlucca.it/~bemporad/teaching/mpc/imt/1-linear_mpc.pdf.

[18] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[19] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems (ARC-RAS)*, vol. 3, no. 1, pp. 269–296, 2020.

[20] K. Pereida and A. P. Schoellig, "Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7831–7837.

[21] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[22] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[23] L. Ljung, *System Identification: Theory for the User*. Prentice-Hall, 1999.

[24] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[25] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press Cambridge, MA, 2006.

[26] B. Settles, *Active Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2012.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, 1989.

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[30] R. Pascanu, G. Montufar, and Y. Bengio, "On the number of response regions of deep feedforward networks with piecewise linear activations," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2014.

[31] G. F. Montufar, "Notes on the number of linear regions of deep neural networks," in *Proc. of the International Conference on Sampling Theory and Applications (SampTA)*, 2017.

[32] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks," in *Proc. of the International Conference on Machine Learning (ICML)*, 2017, pp. 2847–2854.

[33] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *arXiv preprint arXiv:1910.05446*, 2019.

[34] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology (TCST)*, vol. 28, no. 6, pp. 2736–2743, 2019.

[35] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3363–3370, 2019.

[36] C. McKinnon, "Learning-based path-tracking control for ground robots with discrete changes in dynamics," Ph.D. dissertation, University of Toronto (Canada), 2021.

[37] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 13, pp. 1547–1563, 2016.

[38] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 3769–3776, 2021.

[39] C. D. McKinnon and A. P. Schoellig, "Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 2180–2187, 2019.

[40] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066.

[41] M. Prajapat, A. Lahr, J. Köhler, A. Krause, and M. N. Zeilinger, "Towards safe and tractable Gaussian process-based MPC: efficient sampling within a sequential quadratic programming framework," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2024, pp. 7458–7465.

[42] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

[43] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[44] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019, vol. 1.

[45] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[46] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.

[47] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research (JAIR)*, vol. 47, no. 1, pp. 253–279, 2013.

[48] J. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control (TAC)*, vol. 42, no. 5, pp. 674–690, 1997.

[49] L. C. Baird, "Residual algorithms: reinforcement learning with function approximation," in *Proc. of the Twelfth International Conference on Machine Learning (ICML)*, 1995, pp. 30–37.

[50] S. Zhang, W. Boehmer, and S. Whiteson, "Deep residual reinforcement learning," in *Proc. of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2020, pp. 1611–1619.

[51] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Proc. of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6287–6296.

[52] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil, "Deep reinforcement learning and the deadly triad," *arXiv preprint arXiv:1812.02648*, 2018.

[53] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: combining improvements in deep reinforcement learning," in *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018, pp. 3215–3222.

[54] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016, pp. 2094–2100.

[55] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. of the International Conference on Machine Learning (ICML)*, 2016, pp. 1995–2003.

[56] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, 1992.

[57] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018, accessed: 2024-07-09.

[58] W. Chung, V. Thomas, M. C. Machado, and N. L. Roux, "Beyond variance reduction: understanding the true impact of baselines on policy optimization," in *Proc. of the International Conference on Machine Learning (ICML)*, 2021, pp. 1999–2009.

[59] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of the International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[61] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. of the International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897.

[62] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2016.

[63] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of the International Conference on Machine Learning (ICML)*, 2018, pp. 1587–1596.

[64] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. of the Conference on Robot Learning (CoRL)*, 2021.

[65] F. P. Bejarano, L. Brunke, and A. P. Schoellig, "Safety filtering while training: Improving the performance and sample efficiency of reinforcement learning agents," *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 1, pp. 788–795, 2025.

[66] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[67] L. Schneider, J. Frey, T. Miki, and M. Hutter, "Learning risk-aware quadrupedal locomotion using distributional reinforcement learning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 11 451–11 458.

[68] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[69] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[70] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[71] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[72] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[73] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from https://www.wandb.com/.

[74] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[75] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[76] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach*. Springer Science & Business Media, 2013, vol. 36.

[77] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "$\mathcal{L}1$-adaptive MPPI architecture for robust and agile control of multirotors," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7661–7666.

[78] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: theory and applications," in *Proc. of the European control conference (ECC)*, 2019, pp. 3420–3431.

[79] F. P. Bejarano, L. Brunke, and A. P. Schoellig, "Multi-step model predictive safety filters: reducing chattering by increasing the prediction horizon," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2023, pp. 4723–4730.

[80] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. Theodorou, "$\mathcal{L}1$-GP: $\mathcal{L}1$ adaptive control with Bayesian learning," in *Proc. of the Learning for Dynamics and Control Conference (L4DC)*, 2020, pp. 826–837.

[81] R. C. Grande, G. Chowdhary, and J. P. How, "Experimental validation of Bayesian nonparametric adaptive control using Gaussian processes," *Journal of Aerospace Information Systems (JAIS)*, vol. 11, no. 9, pp. 565–578, 2014.

[82] S. Zhou, K. Pereida, W. Zhao, and A. P. Schoellig, "Bridging the model-reality gap with Lipschitz network adaptation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 1, pp. 642–649, 2021.

[83] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *Proc. of the European Control Conference (ECC)*, 2015, pp. 2496–2501.

[84] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll, "A review of safe reinforcement learning: methods, theories and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 46, no. 12, pp. 11 216–11 235, 2024.

[85] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[86] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, pp. 47–72, 2024.

[87] C. Powers, D. Mellinger, and V. Kumar, "Quadrotor kinematics and dynamics," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, ch. 16, pp. 307–327.

[88] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020.

[89] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarmX: safe swarm coordination in complex environments via implicit non-convex decomposition of the obstacle-free space," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 555–14 561.

[90] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," *arXiv preprint arXiv:1608.05786*, 2016, Technical Report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal.

[91] S. Teetaert, W. Zhao, A. Loquercio, S. Zhou, L. Brunke, M. Schuck, W. Hönig, J. Panerati, and A. P. Schoellig, "Advancing reproducibility, benchmarks, and education with remote sim2real: remote simulation to real robot hardware," *IEEE Robotics & Automation Magazine (RAM)*, vol. 32, no. 1, pp. 117–123, 2025.

[92] Z. Wu, S. Cheng, P. Zhao, A. Gahlawat, K. A. Ackerman, A. Lakshmanan, C. Yang, J. Yu, and N. Hovakimyan, "$\mathcal{L}1$ quad: $\mathcal{L}1$ adaptive augmentation of geometric control for agile quadrotors with performance guarantees," *IEEE Transactions on Control Systems Technology (TCST)*, vol. 33, no. 2, pp. 597–612, 2025.

[93] A. P. Schoellig, F. L. Mueller, and R. D'andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1–2, pp. 103–127, 2012.

[94] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Deep neural networks as add-on modules for enhancing robot performance in impromptu trajectory tracking," *International Journal of Robotics Research (IJRR)*, vol. 39, no. 12, pp. 1397–1418, 2020.

[95] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: a survey," *IEEE Transactions on Robotics (T-RO)*, vol. 40, pp. 3044–3067, 2024.

[96] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 5330–5337.