Stochastic Control and Reinforcement Learning

Sam Cohen, Leandro Sanchez-Betancourt, Christoph Knochenhauer

This version: January 19, 2026

Contents

Introduction

This book grew out of an undergraduate masters course taught at the Mathematical Institute, University of Oxford. The book is aimed at mathematicians and it does not assume any prior knowledge on optimal control (deterministic nor stochastic). The book also introduces some of the mathematical results supporting the growing field of reinforcement learning.

We thank Xiaolu Tan, Lingyi Yang and Wojtek Anyszka for comments and pointing out errors in early versions of these notes.

Assumed knowledge

Although we aimed for the book to be self contained, we assume that the reader has some familiarity with:

(i) measure theoretic probability (some background material is in the appendix),

(ii) stochastic differential equations (some background material is in the appendix),

(iii) basic first and second order PDE theory and numerical methods (e.g., finite differences),

(iv) and fundamentals of coding for scientific computing in Python if reviewing the implemented examples we have online.

Notation

We will try and be consistent with notation throughout this book.

(1) For the avoidance of doubt, $0 \notin \mathbb{N}$, but $0 \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

(2) A process (whether random or deterministic, in either discrete or continuous time) will be denoted with a capital letter (say X), and the value it takes at time t will be either X_t or $X(t)$ as convenient. The space it takes values in is the calligraphic $\mathcal{X}$, and a typical value in the set is denoted x.

(3) The set of times which we are considering in our problem will be $\mathbb{T}$, and may be $\{0, 1, ..., T\}$, $\{0, 1, ...\}$, $[0, T]$ or $[0, \infty)$ as context requires. We will use s and t as time variables.

(4) The size of a set A (that is, the number of elements it contains), will be written $|A|$ or $\#A$ if there might be confusion.

(5) The indicator function will be written 1_A, where A is some event or condition (so $1_A = 1$ if A occurs, and $1_A = 0$ otherwise).

(6) The expectation operator will be written $\mathbb{E}$, and the variance $\mathbb{V}$. These can be augmented with various superscripts, which specify (in some way) how the probabilities are chosen, for notational convenience.

(7) Partial derivatives will be written using the shorthand $\partial_t = \frac{\partial}{\partial t}$, and when there is a clear spatial variable x we write ∇ or D_x for the column vector with components ∂_{x_i}, so $\nabla v = D_x v$ is the gradient of v . Similarly, we write $D^2_{xx} v$ for the Hessian of v.

(8) We think of vectors as column vectors.

(9) The Euclidean norm will be denoted $\|x\|$, the ℓ^∞ norm denoted $\|x\|_\infty = \max_i\{|x_i|\}$, and the (Euclidean) inner products will be denoted either $\langle x, y\rangle$ or $x^\top y$.

(10) The minimum of two quantities will be written $\min\{x, y\} = x \wedge y$, and the maximum $\max\{x, y\} = x \vee y$.

References

While these lectures are aiming to be self-contained (and the proofs may differ from those which are 'standard'), this is an area with many good books. However, you will find that there is a range of styles, with varying levels of rigour and applicability. A few sources (in a roughly increasing level of complexity/rigour) are:

1. Sutton and Barto *Reinforcement Learning: An Introduction*, MIT 1998
2. Whittle, P. *Optimal Control: Basics and Beyond*, Wiley, 1996
3. Meyn, S. *Control Systems and Reinforcement Learning*, Cambridge University Press, 2022
4. Bertsekas, Dimitri P. *A Course in Reinforcement Learning (2nd Edition)*, Athena Scientific, 2024
5. Szepesvári, Theoretical Foundations of Reinforcement Learning, `https://rltheory.github.io/`
6. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, Wiley, 2014
7. Bensoussan, *Estimation and Control of Dynamical Systems*, Springer 2018
8. Pham, *Continuous-time Stochastic Control and Optimization with Financial Applications*, Springer 2009
9. Bertsekas and Shreve, *Stochastic Optimal Control: The Discrete-time case*, Athena Scientific, 1996
10. Yong and Zhou *Stochastic Controls: Hamiltonian Systems and HJB equations*, Springer 1999
11. Touzi, *Optimal Stochastic Control, Stochastic Target Problems and Backward SDE*, Fields Lecture Notes 2010
12. Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, Springer 2006
13. Krylov, *Controlled Diffusion Processes*, Springer 1980

Chapter 1

Discrete-time Deterministic Control

In the first five chapters of the book, we will look at discrete-time optimal control problems. In this chapter we will begin by considering deterministic problems, and then, in the next chapter, introduce randomness in our system.

In optimal control, we wish to make decisions about actions which modify the state of the world. To make a mathematical model of this, we first need to describe what we mean by 'the state of the world', and how this is affected by our actions. We will begin with a simple discrete-time deterministic setting, which avoids technicalities, while showing us some of the basic properties of these problems.

We begin with a simple motivating example:

Example 1.0.1. *Suppose you are running a lemonade stall for a week (days $t = 0, 1, ..., 6$). Each day, you choose the quantity of new ingredients to buy, and you set the price of lemonade that day. Depending on the price you charge, you will sell a variable amount of lemonade, which reduces your inventory of ingredients. Your aim is to make the most profit after a week and to keep the inventory level "close" to given baseline quantity.*

To build a mathematical model of this, we write X_t for the ingredients you have at the start of day t (where for simplicity, we describe ingredients using a single variable which tells us how many servings we can make). We know the initial inventory $X_0 = x_0$. On day t, you choose a control $U_t = (\delta_t, p_t)$, where δ_t is the quantity of ingredients you purchase (you cannot buy a negative amount and there a maximum amount you can buy), and p_t is the price you charge during the day. The demand for lemonade we model using a (deterministic) decreasing function D, where $D(p_t)$ is the number of servings we sell if the price is p_t.

From this, we know that our ingredients satisfy the dynamics

$$X_{t+1} = X_t - D(p_t) + \delta_t,$$

and we have the practical requirement that $X_t \geq 0$.

We now need to describe our costs: let the cost of ingredients[1] *per single serving be* $C(\delta)$ *for a given function* C*, that is, at time* t *we will spend* $\delta_t\, C(\delta_t)$*. We suppose it may be expensive to hold inventory, which is described by a function* Γ *which adds a cost* $\Gamma(X_t)$ *at time* t*. However, these costs are offset by our revenue from sales, which is given by* $p_t\, D(p_t)$*. We combine these to give an objective which we want to minimize, that is,*

$$J(x_0) = \sum_{t=0}^{6} \Big(\delta_t\, C(\delta_t) + \Gamma(X_t) - p_t\, D(p_t) \Big)$$

subject to the requirement that X_t *satisfies our dynamics and is nonnegative, and starts at the value* $X_0 = x_0$*. The challenge is to find the optimal choice of* $U_t = (\delta_t, p_t)$*, and the minimal cost.*

Motivated by the example we have just seen, we set up the mathematical notation needed to model a general decision problem. We suppose we have a *state process* $X = \{X_t\}_{t\in\mathbb{T}}$, which describes all (relevant) properties of the world, with $\mathbb{T} = \{0, 1, ..., T\}$ and $T \in \mathbb{N}$. We will assume that X takes values in $\mathcal{X} \subseteq \mathbb{R}^d$ for some $d \geq 1$. This process will be affected by a control process, which we denote $\{U_t\}_{t\in\mathbb{T}}$, and takes values in some set $\mathcal{U}$.

We will assume that X can be described through its one-step dynamics, which we write as

$$X_{t+1} = f(t, X_t, U_t),$$

where $f : \mathbb{T} \times \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a function (which we will assume known, for now). We will make assumptions about f as we go. This is known as the *state dynamics* or *plant equation*.

An agent wishes to optimize their rewards and costs. There are two conventions – in the mathematical control and optimization community, we usually think about minimizing some cost; in the reinforcement learning community, we usually think about maximizing rewards. For the sake of consistency, we will follow the convention of minimizing costs (even for when presenting reinforcement learning algorithms), but the only difference is a change of sign.

We describe the agent's costs by a function $g : \mathbb{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, so that $g(t, X_t, U_t)$ represents the cost which the agent must pay at time t, in state X_t, if they choose control U_t. We seek to find an optimal control, that is, a control which minimizes

$$J(x, U) = \sum_{t\in\mathbb{T}} g(t, X_t^U, U_t) \tag{1.1}$$

with respect to $U = \{U_t\}_{s\in\mathbb{T}}$, where $X_0 = x$ is the initial value of X (which is where the system begins) and where X^U is the solution of the plant equation (for $t \in \mathbb{T}$) with control U.

[1] We allow C to be non-constant, which captures the idea that it may be very expensive to purchase a very large quantity of ingredients. When C is linear in δ, we obtain a model that is similar to the 'temporary price impact' framework in algorithmic trading; see e.g., the books [2, 3].

Remark 1.0.2. We have said that X should contain all relevant information. What do we mean by relevant? Clearly X should be enough to allow us to determine our costs/rewards at every time (we will define these later), as this allows us to describe our preferences about the world. Furthermore, it is important that X is enough to determine the future dynamics of the world, without needing to know any additional information. In particular, we will assume that the *current* state is enough to build a model of the future – we gain nothing by remembering more information (for example the past values of X and U). In a stochastic setting, this is closely related to a Markov assumption (but this is made complicated by the control, as we will see later). If we want to include more memory, we can expand X to include its past values, at the cost of increasing the dimension of X.

1.1 Alternative optimization approaches

Now that we have specified our problem, there are a few ways that we could try and resolve the optimization problem.

1. We could try and find the cheapest U_t for each pair (X_t, X_{t+1}), and so define

$$c(t, X_t, X_{t+1}) = \min_{U_t} \Big\{ g(t, X_t, U_t) : X_{t+1} = f(t, X_t, U_t) \Big\}.$$

   and $c(T, X_T, X_{T+1}) = \min_{U_T} g(T, X_T, U_T)$. This would convert our problem into minimizing the new functional $\sum_{t\in\mathbb{T}} c(s, X_t, X_{t+1})$, which is the problem of calculus of variations. Doing this conversion is not always simple, and it doesn't easily allow us to include randomness.

2. We could consider minimizing J with respect to $\{X_t, U_t\}_{t\in\mathbb{T}}$, by treating $X_{t+1} = g(t, X_t, U_t)$ as a constraint, which we can handle with Lagrange multipliers. This is a very high dimensional problem though, so can be tricky to solve (but we will return to this approach later, see also Exercise 1.4.2!).

3. We can embed our optimization within a family of optimization problems, by considering the behaviour over a single step. This exploits the dynamic nature of our problem, allowing us to reduce our high-dimensional problem (of finding the best control at all times) to a sequence of low-dimensional problems (of finding the control at each time, given the future controls). This will lend itself to stochastic problems as well.

1.2 Building a dynamic programming problem

Instead of just optimizing J in (1.1), we will consider the family of problems given by minimizing the *cost-to-go* (or *remaining-cost*), which we abuse notation and write as

$$J(t,x,U) = \sum_{s \geq t} g(s, X_s^{t,x,U}, U_s)$$

where $X^{t,x,U}$ solves the plant equation with control U and initial value $X_t = x$. With this notation, $J(0,x,U)$ is our original optimization objective in (1.1).

The basic principle of dynamic programming is then fairly simple. We observe that $J(t,x,U)$ depends on U only through the values of U_s for $s \geq t$. We then write

$$J(t,x,U) = g(t,x,U_t) + J(t+1, X_{t+1}^{t,x,U}, U).$$

So, with a further abuse of notation

$$J(t,x,U) = g(t,x,U_t) + J\big(t+1, f(t,x,U_t), \{U_s\}_{s \geq t+1}\big). \tag{1.2}$$

We can then optimize with respect to U_t and $\{U_s\}_{s \geq t+1}$ independently, to get the *Bellman equation*

$$\begin{aligned} V(t,x) &:= \inf_U J(t,x,U) \\ &= \inf_{U_t} \Big\{ g(t,x,U_t) + \inf_{\{U_s\}_{s \geq t+1}} J\big(t+1, f(t,x,U_t), \{U_s\}_{s \geq t+1}\big) \Big\} \\ &= \inf_{U_t} \Big\{ g(t,x,U_t) + V\big(t+1, f(t,x,U_t)\big) \Big\}. \end{aligned}$$

Note that the Bellman equation also holds for $t = T$ if we define $V(T+1,x) \equiv 0$, or equivalently $V(T,x) = \inf_u g(T,x,u)$ (and in many cases we will assume that $g(T,x,u)$ does not depend on u, so this is trivial).

This allows us to compute the optimal cost function (or *value function*) V sequentially backward in time t. Using V, we can then identify U_t as the arg min in the Bellman equation, which describes the (set of) optimal controls.

Remark 1.2.1. Even in this simple setting, there are some interesting things to say about dynamic programming, which we will explore in more detail later.

(i) One way of looking at dynamic programming is as a computational tool. Instead of having to solve the high-dimensional constrained optimization problem where we find the optimal U subject to X being constrained to satisfy the specified dynamics, we solve a family of low dimensional, unconstrained optimization problems given by the dynamic programming equation. This may be computationally much easier, depending on the context.

(ii) Another, more modelling-driven perspective, is that we might have an agent who is allowed to change their mind at any time. The dynamic programming equation tells us that our agent is dynamically-consistent, in that if we find an optimal strategy at time zero, then that strategy remains optimal at all future times (with the remaining-cost being used at time t) and, furthermore, if the agent changes to a different strategy, which at time t they might consider optimal, then at time $t = 0$ we are

indifferent about such a change – the resulting changed policy will also be optimal.

The key fact that ensures dynamic programming holds here is the additive structure of J in (1.2), which ensures that J is monotone with respect to the future cost-to-go – there's no situation where you don't want to minimize tomorrow's costs unless it's expensive today.

1.3 Some examples

Example 1.3.1. *Let us explore Example 1.0.1 further. Assume the demand for lemonade is piecewise linear and of the form* $D(p) = \max\{0, \bar{D} - d\,p\}$ *with* $\bar{D} = 10$ *and* $d = 5$. *Let the cost per unit of ingredient be linear* $C(\delta) = q + r\,\delta$ *for* $q = 3$ *and* $r = 0.5$. *The inventory cost function* $\Gamma(X_t)$ *is taken to be* $\Gamma(x) = \gamma\,(x - \bar{x})^2$ *for simplicity, where the eagerness parameter to keep inventory close to* $\bar{x} = 5$ *be* $\gamma = 1$. *Figure 1.1 shows the functions* $C(\delta)$, $D(p)$, *and* $\Gamma(x)$.

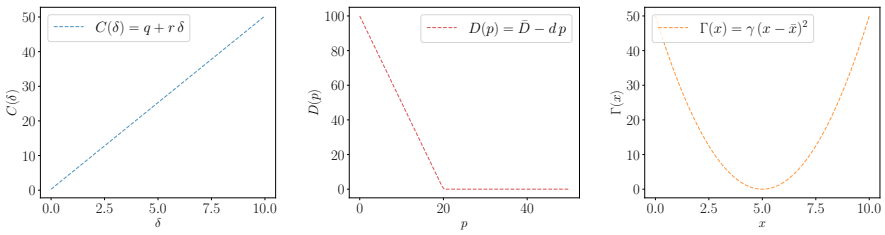


Figure 1.1: Left panel: cost function $C(\delta)$. Middle panel: demand function $D(p)$. Right panel: penalty function $\Gamma(x)$. Model parameters are $q = 3$, $r = 0.5$, $\bar{D} = 10$, $d = 5$, $\gamma = 1$, and $\bar{x} = 5$.

Let $V(t, x)$ *be the value function and* $U = (\delta, p)$. *The functions* f *and* g *are given by*

$$f(t, x, U) = x - (\bar{D} - d\,p) + \delta\,,$$
$$g(t, x, U) = \delta(q + r\,\delta) + \gamma\,(x - \bar{x})^2 - p(\bar{D} - d\,p)\,.$$

We will temporarily ignore the requirement the the inventory is nonnegative, and attempt to find an optimal control without this restriction.

From the dynamic programming principle, we have that for $t \in \{0, 1, \ldots, 6\}$

$$V(t, x) = \inf_{(\delta, p) \in [0, \bar{\delta}] \times \mathbb{R}^+} \left\{\delta(q + r\,\delta) + \gamma(x - \bar{x})^2 - p(\bar{D} - d\,p) + V(t+1, x - (\bar{D} - d\,p) + \delta)\right\}$$

with the convention that $V(6, x) = \Gamma(x)$ *(to capture a terminal penalty on being away from* $\bar{x}$*). By inspection of the functions* g *and* f *we employ the ansatz*

$V(t,x) = h_0(t) + h_1(t)\,x + h_2(t)\,x^2$ *to obtain*

$$\begin{aligned} h_0(t) + h_1(t)\,x + h_2(t)\,x^2 \\ = \inf_{(\delta,p)} \Big\{ \delta(q + r\,\delta) + \gamma\,(x - \bar{x})^2 - p(\bar{D} - d\,p) + h_0(t+1) \\ + h_1(t+1)\big(x - (\bar{D} - d\,p) + \delta\big) \\ + h_2(t+1)\big(x - (\bar{D} - d\,p) + \delta\big)^2 \Big\}. \end{aligned}$$

Using the first order conditions to find the optimizer, we obtain that

$$\begin{aligned} p^* &= \frac{\bar{D}\,r - d\,r\,h_1(t+1) + (\bar{D} + d\,q + 2\,d\,\bar{D}\,r - 2\,d\,r\,x)h_2(t+1)}{2\,d(r + (1 + d\,r)h_2(t+1))}, \\ \delta^* &= -\frac{q + h_1(t+1) + (-\bar{D} + d\,q + 2\,x)h_2(t+1)}{2(r + (1 + d\,r)h_2(t+1))}\,. \end{aligned}$$

Plugging these back in the Bellman equation and taking all terms to one side, we match coefficients of powers of x to obtain an equation of the form

$$0 = \{\cdots\} + x\,\{\cdots\} + x^2\{\cdots\}.$$

From here, given that this equation should be zero for all values of x, we conclude that each of the expressions in the curly brackets should be zero and obtain a system of equations that characterize h_0, h_1, and h_2.[2] *For example, the equation for h_2 is*

$$h_2(t) = \frac{r\,\gamma + (r + \gamma + d\,r\,\gamma)h_2(t+1)}{r + (1 + d\,r)h_2(t+1)}\,,$$

similarly,

$$h_1(t) = \frac{-2\,r\,\bar{x}\,\gamma + r\,h_1(t+1) - (q + \bar{D}r + 2\,\bar{x}\,\gamma + 2\,d\,r\,\bar{x}\,\gamma)h_2(t+1)}{r + (1 + d\,r)h_2(t+1)},$$

and we do not show the value of h_0 because it is not needed to determine the control. We can then solve these equations numerically; Figure 1.2 shows the optimal strategy (top panels), inventory trajectory (bottom left), and costs (bottom right), for a variety of initial inventory levels. Note that negative costs are profits.

[2] In the GitHub repository for the book there is a Mathematica notebook with the details.

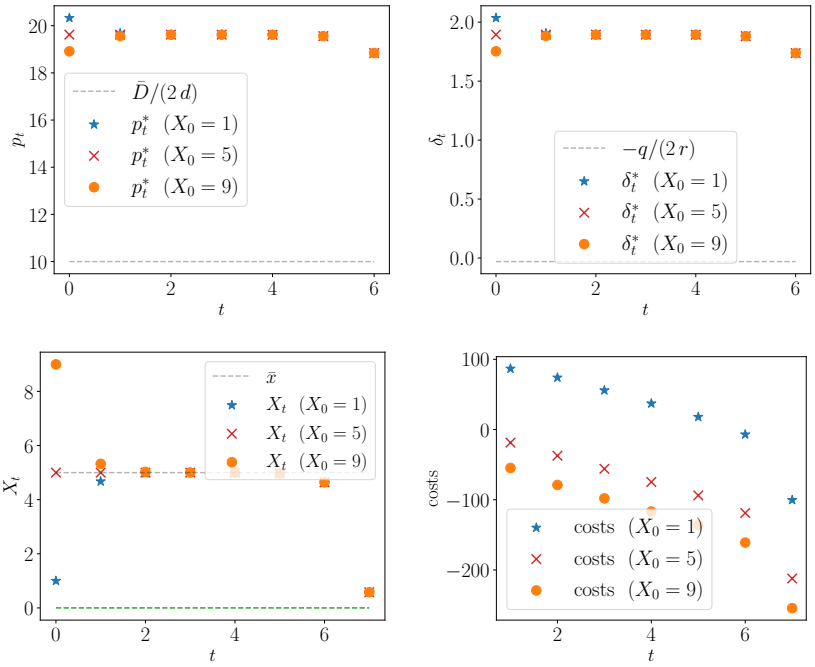


Figure 1.2: Implementation of the lemonade problem. Top panels: optimal strategies (p_t^*, δ_t^*). Bottom left panel: inventory trajectory. Bottom right panel: cumulative costs (negative values are profits).

We observe that in all these scenarios, the inventory is positive at all times, and so this requirement (which was omitted in our derivation) is automatically satisfied. Similar to the previous example, we see the turnpike effect show again, and by looking at the optimal controls p^* and δ^*, we see that they are fairly far from their myopic minimisers $\bar{D}/(2\,d)$ and $-q/(2\,r)$, that is, the minimisers of $p \to -p(\bar{D} - d\,p)$ and $\delta \to \delta(q + r\,\delta)$ respectively. This shows the eagerness of the controller to keep X_t close to $\bar{x}$ and the planning involved.

We next consider an abstract special case, for which we can give a closed-form solution.

Example 1.3.2. *Consider the one-dimensional Linear-Quadratic problem, where $\mathcal{X} = \mathcal{U} = \mathbb{R}$ and for $t \le T$,*

$$X_{t+1} = a + bX_t + U_t \qquad \Rightarrow \qquad f(t,x,u) = a + bx + u,$$
$$g(t,x,u) = \alpha + \beta(x - \mu_x)^2 + \gamma(u - \mu_u)^2.$$

We make an ansatz (i.e. an educated guess) that the value function is quadratic, so can be written in the form

$$V(t,x) = \pi_t + \rho_t(x - \xi_t)^2,$$

for some values of π_t, ρ_t, ξ_t. We have the trivial value $V(T+1, x) \equiv 0$, so can write $\pi_{T+1} = \rho_{T+1} = \xi_{T+1} = 0$. The Bellman equation then is

$$V(t,x) = \inf_{U_t} \Big\{ \underbrace{\alpha + \beta(x-\mu_x)^2 + \gamma(U_t - \mu_u)^2}_{g(t,x,U_t)} + \underbrace{\pi_{t+1} + \rho_{t+1}\big(a + bx + U_t - \xi_{t+1}\big)^2}_{V(t+1, f(t,x,U_t))} \Big\}.$$

Basic calculus shows that the optimal strategy is of the form

$$U_t^* = \frac{\gamma\mu_u + \rho_{t+1}(\xi_{t+1} - a)}{\gamma + \rho_{t+1}} - \frac{b\rho_{t+1}}{\gamma + \rho_{t+1}} x =: h_t + k_t x \tag{1.3}$$

and hence, by substitution in the Bellman equation,

$$\begin{aligned} V(t,x) &= \alpha + \beta(x-\mu_x)^2 + \gamma(U_t^* - \mu_u)^2 + \pi_{t+1} + \rho_{t+1}\big(a + bx + U_t^* - \xi_{t+1}\big)^2 \\ &= \alpha + \beta(x-\mu_x)^2 + \gamma(h_t + k_t x - \mu_u)^2 \\ &\quad + \pi_{t+1} + \rho_{t+1}\big(a + bx + h_t + k_t x - \xi_{t+1}\big)^2 \\ &= \alpha + \pi_{t+1} + \beta(x-\mu_x)^2 + k_t^2\gamma\Big(x - \frac{\mu_u - h_t}{k_t}\Big)^2 \\ &\quad + \rho_{t+1}(b + k_t)^2\Big(x - \frac{\xi_{t+1} - a - h_t}{b + k_t}\Big)^2, \end{aligned}$$

which one can rearrange to obtain

$$\begin{aligned} V(t,x) = \Big[\beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2\Big] &\times \\ \Big(x - \frac{\beta\mu_x + k_t\gamma(\mu_u - h_t) + \rho_{t+1}(b+k_t)(\xi_{t+1} - a - h_t)}{\beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2}\Big)^2 &\\ + \frac{\beta\gamma(k_t\mu_x - \mu_u + h_t)^2 + \beta\rho_{t+1}((b+k_t)\mu_x - \xi_{t+1} + a + h_t)^2}{\beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2} &\\ + \frac{\gamma\rho_{t+1}(b(\mu_u - h_t) + k_t(\mu_u - \xi_{t+1} - a))^2}{\beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2} + \alpha + \pi_{t+1}\,. & \end{aligned}$$

From the above, together with our ansatz, we can write the backward recursion

$$\begin{aligned} \rho_t &= \beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2, \\ \xi_t &= \frac{\beta\mu_x + k_t\gamma(\mu_u - h_t) + \rho_{t+1}(b+k_t)(\xi_{t+1} - a - h_t)}{\beta + k_t^2\gamma + \rho_{t+1}(b+k_t)^2}, \end{aligned}$$

and similarly for π_t (but note that π_t is not needed to compute the optimal strategy). Various algebraic simplifications of this are possible, as is making the parameters $\alpha, \beta, \gamma, \mu_x, \mu_u$ time dependent.

In order to make the above abstract results more concrete, we give a computational example.

Example 1.3.3. *Consider a control problem where the controller wishes to keep X_t as close as possible to $\mu_x = 5$ and where U_t different from zero induces a cost. The dynamics of the system in this example are*

$$X_{t+1} = 0.5 + 0.5\, X_t + U_t,$$

that is, in the absence of interventions (i.e., if $U_t = 0$ for all t), the dynamics of the system bring X_t back to the equilibrium level $x = 1$. The costs are given by

$$g(t, x, u) = (x - 5)^2 + \gamma\,(u - 0)^2,$$

thus, the controller incurs costs if $U_t \neq 0$ and if $X_t \neq 5$. Below, we evaluate the solutions for $X_0 = 0$, $\mu_x = 5$, $\mu_u = 0$, $a = b = 0.5$, $T = 20$, $\alpha = 0$, $\beta = 1$, and $\gamma \in \{0.01, 0.5, 1, 2\}$.

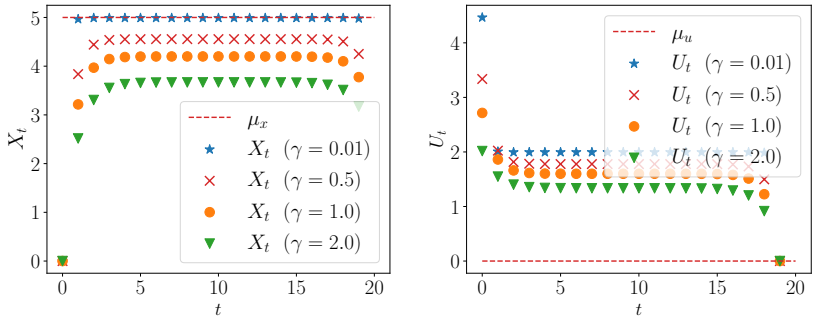


Figure 1.3: Implementation of the one-dimensional deterministic LQ problem. Left panel: optimal trajectory for the state X_t. Right panel: optimal control U_t.

As expected, when the cost parameter γ is small ($\gamma = 0.01$ in the example), the controller takes X_t close to μ_x from the start and compensates any decrease from the mean-reversion to one to keep X_t around $\mu_x = 5$. In the graphs we can also appreciate the so-called turnpike phenomenon, where the trajectory of X stays close to μ_x for most of the time window $[0, 20]$ with deviations at the beginning and at the end. We return to this in the final chapter. Next, we look at the coefficients h_t and k_t from the optimal control in (1.3).

For the trajectory with the least costs to use non-zero controls (the case $\gamma = 0.01$), we see that the optimal strategy is almost of the form $U_t \approx 4.5 - 0.5\, X_t$ making the next step of the state be $X_{t+1} \approx 0.5 + 0.5\, X_t + 4.5 - 0.5\, X_t = 5$. As expected there is a relaxation in the trajectories of h_t and k_t towards T culminating in the terminal condition $h_T = k_T = 0$.

The following example shows another application of discrete-time deterministic control involving graphs.

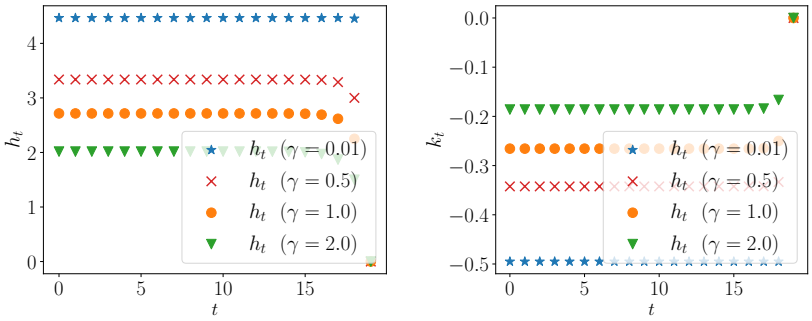


Figure 1.4: Implementation of the one-dimensional deterministic LQ problem. Left panel: trajectory for the auxiliary variable h_t. Right panel: trajectory for the auxiliary variable k_t.

Example 1.3.4 (Shortest path in a directed graph)**.** *Consider a finite directed graph, that is a set of nodes* $V = \{1, 2, ..., N\}$ *and edges* $E \subset V \times V$*. We assume that all self-connections are possible, that is,* $(x, x) \in E$ *for all* $x \in V$*. Suppose the graph is connected, that is, for any* $x, x' \in V$ *there exists* $m \in \mathbb{N}$ *and a sequence* $x = x_0, x_1, x_2, ..., x_m = x'$ *with* $(x_i, x_{i+1}) \in E$ *for all* i*. We call such a sequence a path from* x *to* x'*. Figure 1.5 shows an example of a directed graph, with seven nodes, that is connected and where all self-connections are possible.*

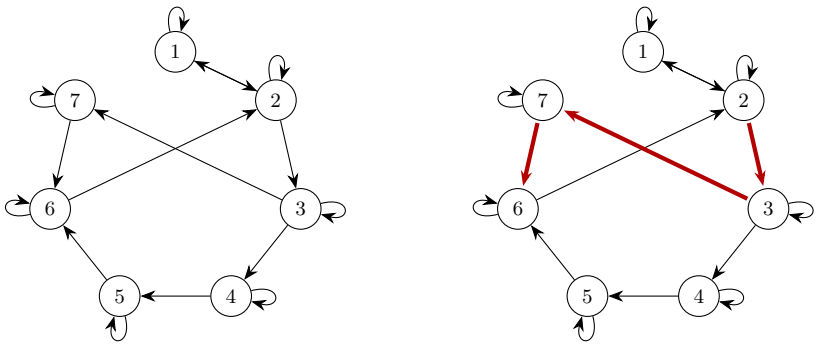


Figure 1.5: Example of a directed graph with seven nodes. The graph is connected and all self-connections are possible. In this example the nodes are $V = \{1, 2, 3, 4, 5, 6, 7\}$, and the edges are $E = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), \ldots, (7, 7)\}$. The right panel highlights the shortest path from $x_0 = 2$ to $x^* = 6$ with length equal to 3.

From such simple setup, we can already infer that for any $x, x' \in V$ *there exists a path from* x *to* x' *with exactly* N *steps. To see this, observe that if there exists a path from* x *to* x'*, then there exists a path without repeated nodes. As there are* N *nodes in total, we know that a path without repeats will have length*

at most N. Now allowing repeats in the final node, we see that there must be a path with exactly N steps.

With such a setup one can study versions of Bellman–Ford's algorithm to find the shortest path between two nodes [1], and we formulate this in our optimal control notation.

We define the cost of following a path of length $T \geq N$ to be

$$J(0, x_0, U) = \left(\sum_{t=1}^{T-1} f(X_t, U_t) \right) + \Phi(X_T)$$

where $U_t \in V$ determines the next step in our path (so $X_{t+1}^U = U_t$), $X_0 = x_0 \in V$, and for a fixed state $x^* \in V$

$$f(x,u) = \begin{cases} 0 & \text{if } x = x^*, \\ 1 & \text{if } (x,u) \in E, x \neq x^*, \\ \infty & \text{if } (x,u) \notin E, \end{cases} \qquad \Phi(x) = \begin{cases} 0 & \text{if } x = x^*, \\ N+1 & \text{if } x \neq x^*. \end{cases}$$

These costs are chosen so that $\inf_U J(t, U)$ is the length of the shortest path to x^*, and we will write down the dynamic programming formulation to find the shortest path.

As there exists a path with at most N steps, and we can repeat the final node with zero cost, we see that $J(0, x_0, U) < N$. In particular, for U minimizing $J(0, x_0, U)$, we know that $\Phi(X_T^U) < N + 1$, so $X_T^U = x^*$. In this case, the costs are simply the number of non-repeating steps in the path. The dynamic programming formula then states that the length of the shortest path is $V(0, x_0) = \min_U J(0, U)$, and we have

$$V(t,x) = \min_{u:(x,u)\in E} \Big\{ f(x,u) + V_{t+1}(u) \Big\}$$

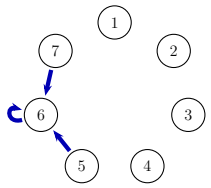
with terminal value $V(T, x) = \Phi$. We also observe that $V(t, x) < N$ if and only if there is a path from x to x^* in at most $T - t$ steps.

Next, we employ the dynamic programming formula above to find the shortest path in the directed graph of Figure 1.5; by inspection it is easy to see that the actual shortest path from $x_0 = 2$ to $x^* = 6$ is the one in red on the right panel of the figure. To solve this mathematically (or computationally) we start by computing $V(T, x)$ for $x \in V$ and $T = 7$. From the terminal condition it follows that

$$V(7,x) = \begin{cases} 0 & \text{if } x = 6, \\ 8 & \text{otherwise.} \end{cases} \tag{1.4}$$

Then, from the dynamic programming formula we have that at time $t = 6$ the value function is given by $V(6, x) = \min_{u:(x,u)\in E}\{f(x, u) + V(7, u)\}$. It follows that

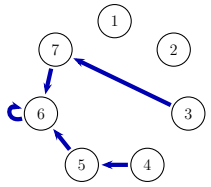
$$V_6(x) = \begin{cases} 0 & \text{if } x = 6, \\ 1 & \text{if } x \in \{5, 7\}, \\ 8 & \text{otherwise.} \end{cases}$$



Here we draw (on the right) the set of transitions which are known to be optimal[3] *at time* $t = 6$.

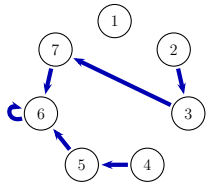
Similarly, using $V(5, x) = \min_{u:(x,u)\in E}\{f(x, u) + V(6, u)\}$ *we have*

$$V(5, x) = \begin{cases} 0 & \text{if } x = 6, \\ 1 & \text{if } x \in \{5, 7\}, \\ 2 & \text{if } x \in \{3, 4\}, \\ 8 & \text{otherwise.} \end{cases}$$



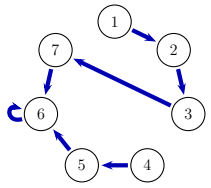
Proceeding in the same way, we have that

$$V(4, x) = \begin{cases} 0 & \text{if } x = 6, \\ 1 & \text{if } x \in \{5, 7\}, \\ 2 & \text{if } x \in \{3, 4\}, \\ 3 & \text{if } x = 2, \\ 8 & \text{if } x = 1, \end{cases}$$



and lastly, for $t \in \{0, 1, 2, 3\}$

$$V(t, x) = \begin{cases} 0 & \text{if } x = 6, \\ 1 & \text{if } x \in \{5, 7\}, \\ 2 & \text{if } x \in \{3, 4\}, \\ 3 & \text{if } x = 2, \\ 4 & \text{if } x = 1. \end{cases}$$



Thus, the shortest path from $x_0 = 2$ *to* $x^* = 6$ *has length* $V(0, 2) = 3$ *and this is achieved by following the (allowed) path* $x \to x'$ *that satisfies* $V(t, x) = V(t + 1, x') - 1$. *In our example this would be* $x_0 = 2 \to 3 \to 7 \to 6$.

In the above example, we might also want to consider the case where, at each time step, the connections available are random. To solve such a problem

[3] At time $t = 6$, all possible transitions from states in $\{1, 2, 3, 4\}$ are equally bad, and so are not shown.

we will need a little more theory, which we will develop in the next section. We will return to this in Example **??**.

1.4 Exercises

Exercise 1.4.1. (A 'time-inconsistent' problem, where the DPP fails.) *Consider a deterministic control problem, where at each time t, our agent's preferences are described by a cost-to-go function*

$$J(t,x,U) = \sum_{s=t}^{T} \frac{1}{1+(s-t)} g(s, X_s, U_s),$$

which is sometimes known as hyperbolic discounting. *We assume $f(t,x,u) = x+u$ and $g(t,x,u) = -u+x^2$, where $x_0 \in [0,1]$ and $u \in \mathcal{U} = [0,2]$, with a horizon $T = 2$.*

Show that the dynamic programming principle fails, i.e., there are controls U which optimize $J(t, X_t, U)$ but do not optimize $J(t+1, X_{t+1}, U)$.

Exercise 1.4.2 (Discrete-time deterministic Pontryagin minimum principle)**.** *Consider a discrete time deterministic control problem, on a finite horizon, where the cost function g and the state dynamics f are both differentiable with respect to the pair (x,u). Suppose that the value function is a differentiable function of the state, and there exists a differentiable function $u^* : \mathbb{T} \times \mathcal{X} \to \mathcal{U}$ such that $u^*(t,x)$ is an optimal control when $X_t = x$. Define X_t^* to be the trajectory when following the control $U_t^* = u^*(t, X_t^*)$.*

(i) Show that

$$0 = \partial_u g(t, X_t^*, U_t^*) + \partial_x V\big(t+1, f(t, X_t^*, U_t^*)\big) \cdot \partial_u f(t, X_t^*, U_t^*)$$

(ii) By induction (or otherwise), show that

$$\partial_x V(t, X_t^*) = \partial_x g(t, X_t^*, U_t^*) + \partial_x V\big(t+1, f(t, X_t^*, U_t^*)\big) \cdot \partial_x f(t, X_t^*, U_t^*).$$

(iii) Hence show that, in order for U^* and X^* to be an optimal control–trajectory pair, they must be part of a fixed point to the following forward-backward system of equations:

$$\begin{aligned} X_{t+1}^* &= f(t, X_t^*, U_t^*); & X_0 &= x; \\ Q_t &= \partial_x g(t, X^*, U_t^*) + Q_{t+1} \cdot \partial_x f(t, X_t^*, U_t^*); & Q_T &= \partial_x g(T, X_T^*); \\ 0 &= \partial_u g(t, X_t, U_t^*) + Q_{t+1} \cdot \partial_u f(t, X_t, U_t^*). \end{aligned}$$

(Here Q is called the adjoint *process, and can be interpreted as a marginal value associated with changing the state, or as a Lagrange multiplier arising from treating the state equation as a constraint.)*

This result is a discrete-time version of Pontryagin's maximum principle, which we will return to in Chapter **??**. *The practical use of this result is that, in many cases, this system of equations can be solved numerically and, if the solution is unique, then this allows us to identify the optimal control process without solving the full Bellman equation.*

Exercise 1.4.3. *Consider the following general linear-quadratic control problem where the state is n-dimensional and the control is m-dimensional. The plant equation is linear and given by*

$$X_{t+1} = A\,X_t + B\,U_t, \qquad X_0 = x \in \mathbb{R}^n,$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. The cost function g is

$$g(x,u) = x^\top Q\,x + u^\top R\,u,$$

where $Q = Q^\top \succeq 0$, and $R = R^\top \succ 0$. The performance criterion over horizon $T \in \mathbb{N}$ with terminal weight $S = S^\top \succeq 0$ is

$$J(t,x,U) = \sum_{s=t}^{T-1} \Big(X_s^\top Q\,X_s + U_s^\top R\,U_s \Big) + X_T^\top S\,X_T.$$

Define the value function

$$V(t,x) = \inf_U J(t,x,U)$$

with the convention $V(T,x) = x^\top S\,x$.

(i) Write down the Bellman equation satisfied by $V(t,x)$.

(ii) Use the ansatz $V(t,x) = x^\top P_t\,x$ with symmetric matrices $P_t = P_t^\top \succeq 0$ and $P_T = S$ in the Bellman equation from the previous part.

(iii) By expanding $x^\top Q\,x + u^\top R\,u + V(t+1, A\,x + B\,u)$ and completing the square in u, derive the optimal control at time t as

$$U_t^* = -K_t X_t, \qquad K_t := \big(R + B^\top P_{t+1} B\big)^{-1} B^\top P_{t+1} A,$$

with the Riccati backward recursion

$$P_t = Q + A^\top P_{t+1} A - A^\top P_{t+1}\,B\big(R + B^\top P_{t+1}\,B\big)^{-1} B^\top P_{t+1} A, \qquad P_T = S.$$

Bibliography

[1] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[2] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

[3] Olivier Guéant. *The Financial Mathematics of Market Liquidity: From optimal execution to market making*. CRC Press, 2016.

Notation

X state process. 10