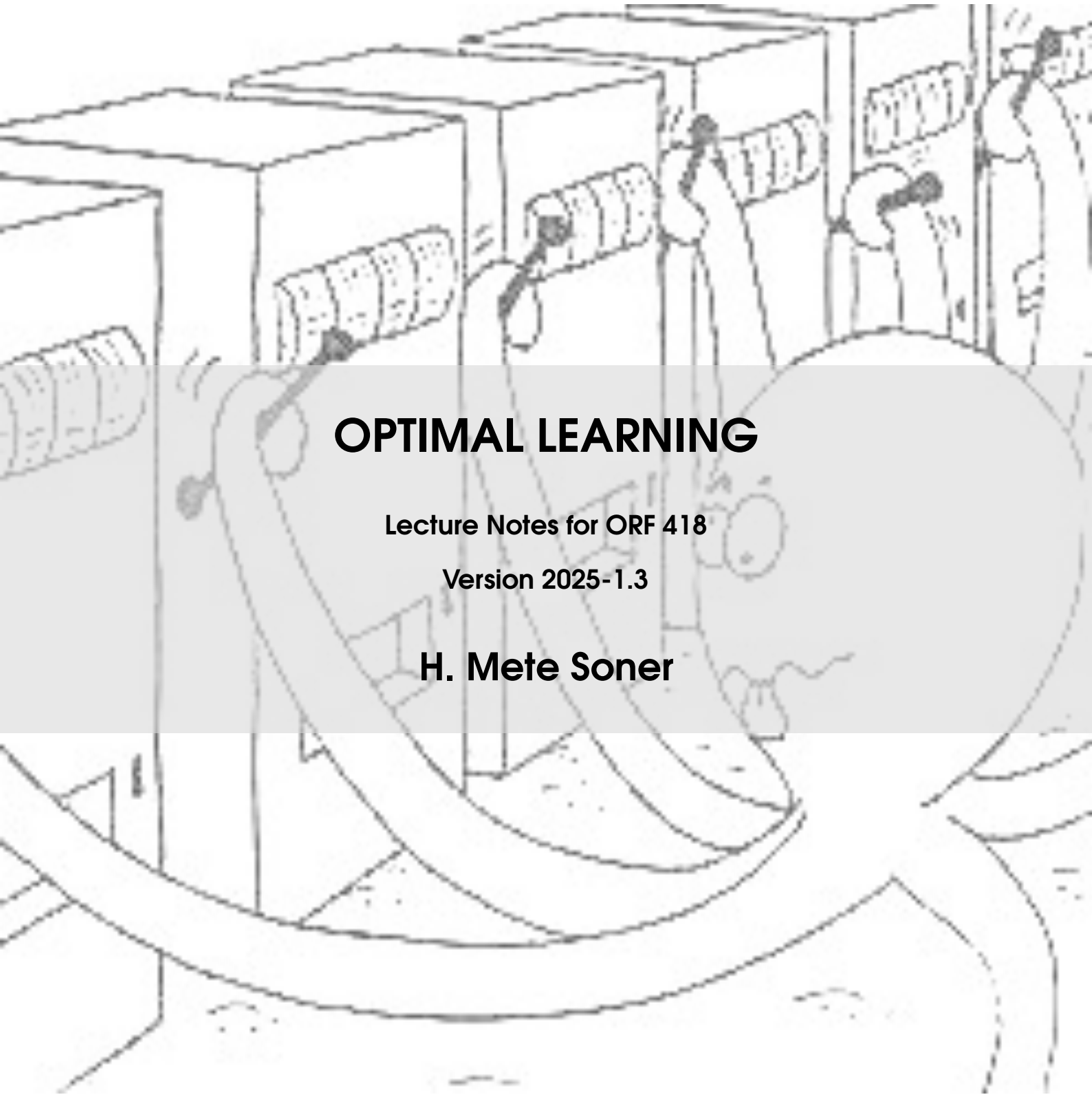


OPTIMAL LEARNING

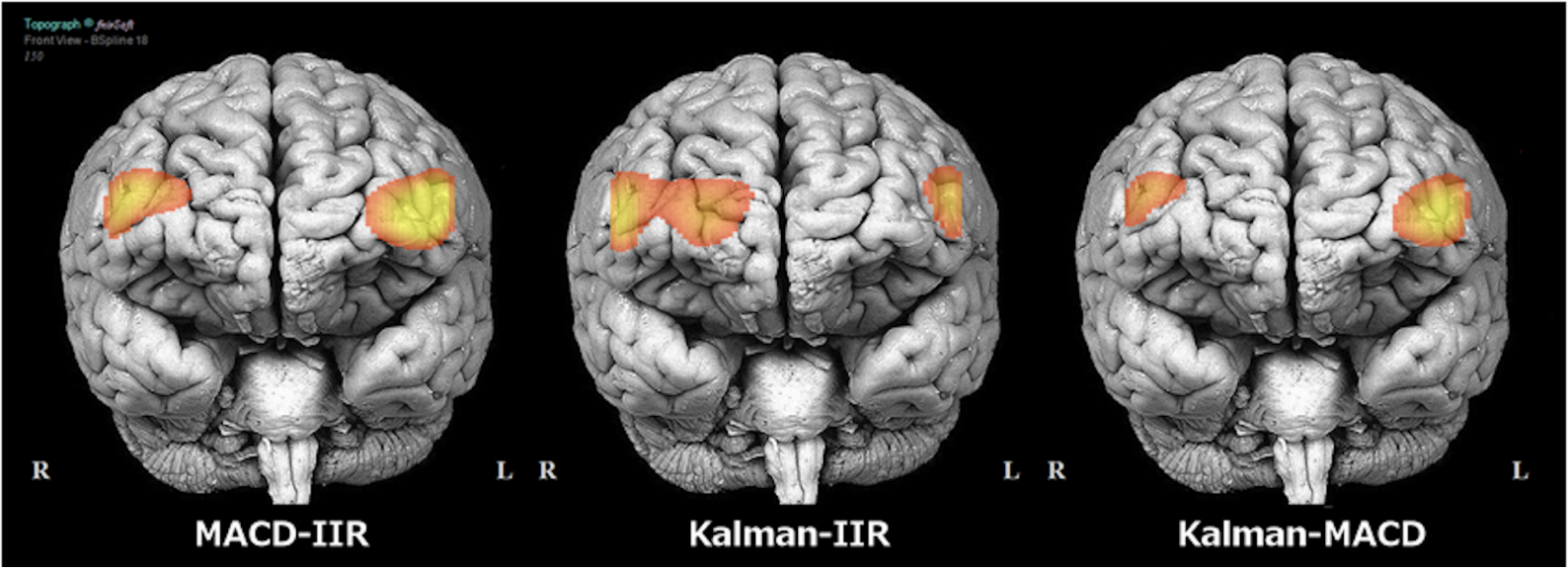
Lecture Notes for ORF 418

Version 2025-1.3

H. Mete Soner

First version, September 2021

Contents



1. Introduction

The main objective in machine learning is to generalize or "learn" from experience. Based on the nature of the data available, modern applications can be divided into three broad categories:

- *Supervised learning*: in which labeled training data is available. Then, the goal is to generalize this set to unseen examples or to find a function that maps inputs to outputs.
- *Unsupervised learning:* in which a training data of unlabeled examples are given. A classical example of this is to learn the statistical distribution of a given data.
- *Reinforcement learning:* in which the learner interacts with an uncertain environment and simultaneously tries to perform a certain task such as self-driving cars or auto-pilots in planes. Therefore, optimization and learning tasks are coupled.

This course develops tools that are centrally used in reinforcement learning. In particular, we focus on methods from decision theory, optimal control and statistical estimation. Control is needed to perform the task at hand optimally, while estimation techniques are required to be able to learn the uncertain environment properly. Additionally, one should be able to combine these methods in an effective and computationally tractable manner.

To illustrate these methods, we study simple models in uncertain environments and with partial information that require estimation or learning. The classical and successful application of the *autopilot* provides the first historical example. An autopilot is a system used to control the path of an aircraft, marine craft or spacecraft without requiring constant manual control by a human operator. The theory of *Kalman filter* has been very effective in this problem and is now widely used. In this example, the deterministic model is derived from classical mechanics. Then, noise is added to the model in order to take into account approximations in measurement, or random wind and weather conditions. In more modern applications, however, there is more uncertainty about the model and the noise. It is therefore necessary to learn the environment while performing the task. The classical *armed-bandit problem* or the widely used *Q-Learning algorithm* are instances of such applications, the final goal of the course is to model and study these problems. However, before addressing the challenges of learning, one needs to develop the general theory of optimal control in discrete time.

The focus of this course is optimal learning and its relation with optimal control, and therefore we start by properly defining the optimal control problem in discrete-time. In particular, two particular control problems that play a central role in this course are studied in detail: the *Linear Quadratic (LQ) problem*, in Chapters 2 and 4, and *Markov Decision Processes (MDP)*, in Chapters 5 and 6. Through relevant examples, we argue why these problems are central to the theory of optimal control and learning. Additionally, through these problems we develop the powerful solution technique *dynamic programming principle* that is commonly used in these two models, and in fact in all control problems.

As introduced above, Chapter 2 considers a specific framework of optimal control problems, called the *Linear Quadratic (LQ) problem*, as well as its stochastic version, the *Linear Quadratic Gaussian (LQG) problem*. These problems are solved through the dynamic programming principle. The advantages of this classical but simple LQ theory is that, first, it can be solved explicitly, and second, it provides a good approximation for many general control problems. However, the relevance of the approximation will not be discussed in the course.

The highly popular LQ problem is modeled under the assumption of full knowledge, and as such it does not involve learning. Nevertheless, after a short detour in Chapter 3 through *Bayesian Learning*, which can be considered as the simplest example of unsupervised learning, we then introduce learning to the LQG problem. Bayesian Learning, also called *Bayesian inference*, is a method of statistical inference in which Bayes formula is used to update the probability for a hypothesis as more evidence or information becomes available. More precisely, Bayesian inference derives the *posterior probability* as a consequence of two antecedents: a *prior probability* and a *likelihood function* derived from a statistical model for the observed data. Bayesian inference computes the posterior probability (or equivalently updates the prior probability) according to Bayes' theorem, every time information is revealed. We also compare this approach to a more standard one in statistics, called the Maximum Likelihood Estimator (MLE).

The Bayesian learning theory developed in Chapter 3 is used in Chapter 4, to study a version of the LQG control problem, in which the controller can only access noisy information about the state variable. This more complicated control problem first requires formulating a relevant estimate of the system state, which is done through what is called a *Kalman filter*, mostly based on the Bayesian estimator. Then, optimal control of the system is performed using this relevant estimation.

The second part of the course is devoted to *Markov Decision Processes (MDP)*, based on Markov chains. This type of control problems is first described in Chapter 5 in a context without learning. The main idea of a MDP is to optimally control a Markov chain: at a given state x, an action is chosen in order to move (with certainty or not) to a new state y where a reward is obtained (or a cost is paid). As in the LQ problems, the dynamic programming principle is the central method used to study this class of problems.

Finally, in Chapter 6 we discuss learning, by introducing dynamic programming based numerical techniques of value and policy iteration. These effective computational tools also form the foundations of many modern reinforcement algorithms, and we use them centrally with learning techniques, in particular to study the *multi-armed-bandit problem*. The name of this problem comes from imagining a gambler at a row of slot machines, who has to decide which machines to play, how many times to play each machine and in which order to play them, and whether to continue with the current machine or try a different machine. This is a classic reinforcement learning problem that exemplifies the *exploration–exploitation* trade-off. We finally conclude by a study of the well-known Q-learning algorithm.

All examples and solution techniques considered in this course are set in discrete time. This is justified by the fact that in many applications, the observation of the system and/or its control can only be done in discrete time. It is also important to note that most continuous-time problems can be discretized, and that the solution of the discrete-time problem usually provides a good approximation of the solution to the associated continuous-time problem. Moreover, any numerical method requires in any cases a discretization of the problem. Obviously, this choice for a discrete-time formulation has been made above all to simplify, because in continuous time, stochastic control problems, and thus learning problems, require the use of advanced theories such as the theory of Stochastic Calculus and Ito's integral.

1.1 Optimal control

As mentioned above, it is important first to develop general notions about control problems. There exist different categories of optimal control problems: in discrete or continuous-time, with or without uncertainty, with finite or infinite horizon. In this course, we mostly focus on discrete-time problems with uncertainty, since most learning problems are precisely based on learning about uncertainty.

Succinctly defined, the theory of optimal control is concerned with operating a dynamical system at minimum cost. In our mathematical models, the state of the system is denoted by $\mathbf{x}$, is called the *controlled process*. The *control process* denoted by $\mathbf{u}$ is chosen at any admissible time t (or continuously) depending on available information, and impacts the future dynamics of the state process $\mathbf{x}$. The goal is to optimally choose the control process $\mathbf{u}$, in order to minimize a *cost functional*, or to maximize a reward.

1.1.1 Deterministic case

In a general discrete-time and deterministic framework, the general dynamic of $\mathbf{x}$ at time t can be written as a known function of its states before t and the control up to time t, *i.e.*

$$x_{t+1} = f\big(t, x_0, \dots, x_t, u_0, \dots, u_t\big), \quad t = 0, 1, 2, \dots.$$

The control is chosen so as to optimize some cost or reward. In the finite-horizon case, for a given *time horizon* T, the functional to optimize has the following form:

$$J_T(\mathbf{x}, \mathbf{u}) := \sum_{t=0}^{T-1} g\big(t, x_t, u_t\big) + h\big(x_T\big),$$

for some appropriate functions g and h. Alternatively, in the infinite horizon case, we consider a functional as an infinite sum:

$$J_\infty(\mathbf{x}, \mathbf{u}) := \sum_{t=0}^{\infty} \rho^t g\big(x_t, u_t\big),$$

for a given discount factor $\rho > 0$. In this case, some restrictions on the admissible control $\mathbf{u}$ may apply to ensure that the sum is finite, and the set of *admissible controls* is denoted by $\mathscr{U}$. The problem is then to find an admissible optimal control $\mathbf{u}^\star$ that optimizes the above functionals. For example, in the case of a minimization problem with finite horizon T, the optimal control problem is equivalent to

$$\min_{\mathbf{u} \in \mathscr{U}} J_T(\mathbf{x}, \mathbf{u}), \quad \textit{i.e.} \text{ finding } \mathbf{u}^\star \text{ s.t } J_T(\mathbf{x}^\star, \mathbf{u}^\star) \leqslant J_T(\mathbf{x}, \mathbf{u}), \text{ for all } \mathbf{u} \in \mathscr{U},$$

where $\mathbf{x}^\star$ is the state of the system when the control $\mathbf{u}^\star$ is chosen.

1.1.2 Stochastic case

To consider a model with uncertainty, it suffices to add a random process ω in order to consider the following dynamics:

$$x_{t+1} = \widetilde{f}\big(t, x_0, \dots, x_t, u_0, \dots, u_t, \omega_0, \dots, \omega_t\big), \qquad t = 0, 1, 2, \dots.$$

The reward to be optimized is defined almost as before, except that we take expectations, in order to obtain a real-valued (not random) objective function. This leads to the following objective functions:

$$J_T(\mathbf{x}, \mathbf{u}) := \sum_{t=0}^{T-1} \mathbb{E}\big[g(t, x_t, u_t)\big] + \mathbb{E}\big[h(x_T)\big], \qquad \text{for the finite horizon case,}$$

$$J_\infty(\mathbf{x}, \mathbf{u}) := \sum_{t=0}^{\infty} \rho^t \mathbb{E}\big[g(x_t, u_t)\big], \qquad \text{for the infinite horizon case.}$$

Again the problem is to find an admissible optimal control $\mathbf{u}^\star$ that optimizes these functionals. For example, in the case of a maximization problem with infinite horizon, the optimal control problem is equivalent to

$$\max_{\mathbf{u} \in \mathscr{U}} J_\infty(\mathbf{x}, \mathbf{u}), \quad \textit{i.e.} \text{ finding } \mathbf{u}^\star \text{ s.t } J_\infty(\mathbf{x}^\star, \mathbf{u}^\star) \geqslant J_\infty(\mathbf{x}, \mathbf{u}), \text{ for all } \mathbf{u} \in \mathscr{U}.$$

We continue by listing several classical examples that have been central to the development of the general theories of control and learning in order to motivate our study.

1.2 Historical examples

The following two examples provide an illustration of optimal control problems without uncertainty. Although we mainly focus on problems with uncertainty and learning and these celebrated applications are not further developed in these notes, we include their brief discussions to showcase the breadth and the variety of the applications that can be studied by the techniques developed in this course.

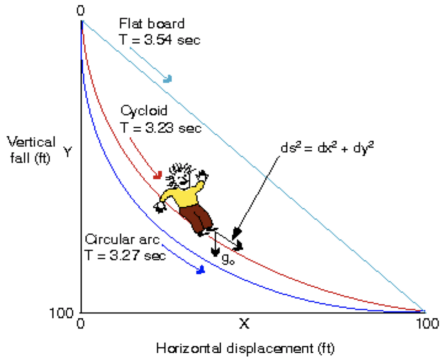
1.2.1 Brachistochrone

We outline this problem for its historical importance. This beautiful problem, its unexpected solution, and the solution technique are not developed in these notes.

The *brachistochrone* problem is considered to be the first optimal control problem formulated in the history. It was posed by Johann Bernoulli in June 1696 and solutions were offered by Newton, Jakob Bernoulli, Gottfried Leibniz, Ehrenfried Walther von Tschirnhaus and Guillaume de l'Hopital.

> The question is to find the shape of the curve down which a bead sliding from rest and accelerated by gravity will slip (without friction) from one point to another in the *least time*.

More formally, a *brachistochrone curve*, or curve of fastest descent, is the one lying on the plane between an original point A and an arrival point B, on which a bead slides under the influence of a uniform gravitational field from A to B in the shortest time. More precisely, consider a bead starting at the initial point A at rest, the question is to find the shape of the curve between A and the arrival point B so that the sliding time is minimal. For simplicity, we will assume that the bead slides from A to B without friction, so that the only force at play is the gravitational one, but the following computations can in fact be adapted to accommodate a more realistic environment.



We parametrize the arc by the horizontal displacement x and let $y(x)$ be the height at that x value. Suppose that the starting point is $(0,0)$ and the target is (a,h). Then, any arc

$$A = \{(x, y(x)) \ : \ x \in [0,a]\}$$

satisfying $y(0) = 0, y(a) = h$ is a candidate for the optimal solution. Let $s(x)$ be the total arc length. Then,

$$s'(x) \ = \ \sqrt{1 + (y'(x))^2} \ ,$$

where $'$ denotes the differentiation with respect to the x-variable. To complete the description of the problem, let $v(x)$ be the velocity of the bead. We calculate it by using the conservation of energy. Initially, the bead has only potential energy. As it slides down the arc, part of the potential energy turns into kinetic energy. Hence,

$$\text{kinetic energy}(x) \ = \ \frac{1}{2} m v^2(x) \ = \ \text{potential energy difference} \ = \ m g y(x),$$

where m is the mass of the bead and g is the gravitational constant. Hence,

$$v(x) = \sqrt{2 g y(x)} \ , \quad x \in [0,a].$$

Then, the time it takes is

$$\text{total time} = \int_0^a \frac{\mathrm{d}s}{v} = \int_0^a \sqrt{\frac{1 + (y'(x))^2}{2gy(x)}} \ \mathrm{d}x.$$

We are now ready to state the *brachistochrone* problem mathematically.

$$\text{minimize} \quad T(y) := \int_0^a \sqrt{\frac{1 + (y'(x))^2}{2gy(x)}} \ \mathrm{d}x,$$

over all differentiable maps $y : [0,a] \mapsto \mathbb{R}$ satisfying the boundary conditions $y(0) = 0, y(a) = h$, where $a, h > 0$ are given in the description of the problem and g is the gravitational constant. The solution is the *cycloid* $A = \{(x(\theta), y(\theta)) \ : \ \theta \in [0, \theta^*]\}$ where

$$x(\theta) = \frac{1}{2} k^2 (\theta - \sin(\theta)), \qquad y(\theta) = \frac{1}{2} k^2 (1 - \cos(\theta)),$$

and the constants k, θ^* are determined by a, h.

The derivation uses the *Euler-Lagrange* equations and will not be developed in this course. It is shown that the solution is given by a *cycloid*, parametrized by some constants that have to be chosen so that the curve fits the starting point A and the ending point B. In particular, the curve is independent of both the mass of the bead and the local strength of gravity. In the solution, the bead may actually travel uphill along the cycloid for a distance, but the path is nonetheless faster than a straight line (or any other line). If the body is given an initial velocity at A, or if friction is taken into account, then the curve that minimizes time differs from the tautochrone curve.

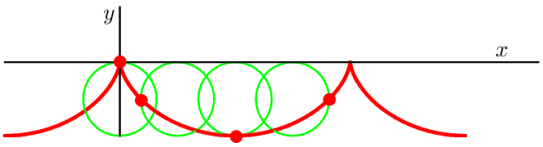
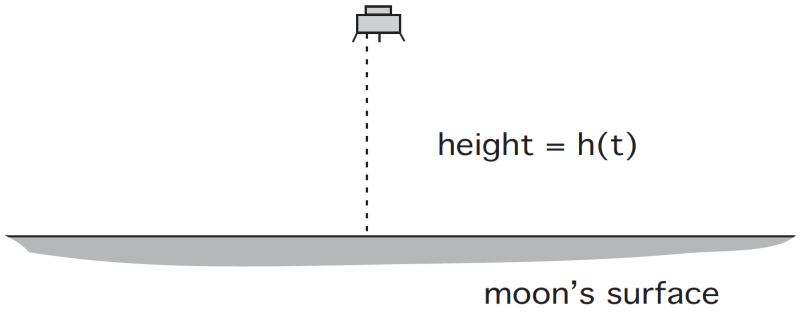


Figure 1.1: Illustration of a cycloid.

1.2.2 Lunar landing

We describe another deterministic optimal control problem for its historical importance that is not directly covered in this course. This problem has ignited an immense development by mathematicians during the race to the moon in the 50's and 60's.



We follow the simplified version given in the lecture notes of L. C. Evans posted in the Canvas and a paper also posted in Canvas outlines the more realistic problem. The problem is to bring a spacecraft to a soft landing on the lunar surface using the *least amount of fuel*. As in the above picture, $h(t)$ is the height from the lunar surface. We let $v(t)$ be its velocity, $\alpha(t)$ is the upwards thrust provided by burning fuel and $m(t)$ be its mass. The weight is decreasing in time as the fuel is burned. Although in most problems this change is very small relative to the total mass, in the original lunar landing problem it was not.

All the other functions are determined by the control function $\alpha(t)$ and of course by physics. We assume that $\alpha(t) \in [0, A]$ where A is a given constant. Then, by Newton's law

$$m(t)h''(t) = -g_l m(t) + \alpha(t),$$

where g_l is the lunar gravitational constant.

We write all the relevant equations as a first order differential system:

$$\begin{aligned} v'(t) = h'(t) &= -g_l + \frac{\alpha(t)}{m(t)}, \\ h'(t) &= v(t), \\ m'(t) &= -k\alpha(t), \end{aligned}$$

where k is a constant that relates how much trust one unit mass of burned fuel provides.

An important outcome of the control is the *landing time* τ defined to be the first time $h(\tau) = 0$. Soft landing requirement translates into $v(\tau) = 0$. Then, the mathematical description of the problem is given as follows:

$$\text{minimize} \int_0^\tau \alpha(t)\,\mathrm{d}t,$$

over all $\alpha : [0,\tau] \mapsto [0,A]$ subject to

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} v(t) \\ h(t) \\ m(t) \end{bmatrix} = \begin{bmatrix} v - g_l \\ v(t) \\ 0 \end{bmatrix} + \alpha(t)\begin{bmatrix} 1/m(t) \\ 0 \\ -k \end{bmatrix}.$$

The optimal solution is to use either the full trust A or nothing.

1.3 A modern example: the multi-armed bandit problem

This is a classical example of the theory of learning that can be taken as a guiding theme for this course. In these notes we develop several solution techniques. In this problem, a fixed limited set of resources must be allocated between competing (alternative) choices in a way that maximizes their expected gain, when each choice's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice. This is a classic reinforcement learning problem that exemplifies the *exploration–exploitation* trade-off. The name comes from imagining a gambler at a row of slot machines, who has to decide which machines to play, how many times to play each machine and in which order to play them, and whether to continue with the current machine or try a different machine. A detailed analysis of this problem will be provided in Chapter 6.

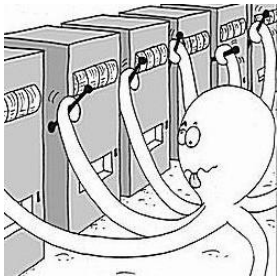
In this introduction, we focus on one example entitled *News website* which considers a news website that presents personalized news headers for a user, in order to maximize the number of clicks. To achieve this, the news website should learn about the characteristics of particular users in order to present them with interesting headers and increasing the probability of a click.

To model this problem, we consider each possible header as an arm in a bandit problem, that is defined later. When the user visits the website for the first time, the site does not have any information on the user, and therefore cannot personalize the header. But then, at every visit of the user, the website can collect information on personal interests, and therefore on the personal "click probability" given a header. This information learned by the website can then be used to personalize the header for this given user. The paper by A. Slivkins, *Introduction to Multi-Armed Bandits*, provides more details.

1.3.1 Mathematical formulation

A decision maker has K different choices. Each choice provides a random reward, chosen from an unknown but fixed distribution. The goal is to *explore and exploit* the different choices in order to maximize the expected return.

An armed-bandit problem is characterized by K choices. Each choice $a \in \{1,\dots,K\}$ is called an *arm*. If chosen, this arm a produces a random reward $R^{(a)}$. We denote $q(a) := \mathbb{E}[R^{(a)}]$. At each period of time $t = 1,2,\dots$, the action of the player is to choose an arm based on the information collected up to that time. The action at time t is denoted a_t. Consider a finite time horizon T, and let $\alpha := (a_1, a_2, \dots, a_T)$ be the action chosen by the player, and $R := (r_1, r_2, \dots, r_T)$ the associated random rewards.



The goal of the player is to choose α in order to maximize the expected sum of rewards:

$$\max_{\alpha} J(\alpha), \qquad \text{where } J(\alpha) := \mathbb{E}\Big[\sum_{t=1}^{T} r_t\Big].$$

Note that, even if it is not explicit in the previous definition of the criteria, the sequence of rewards actually depends on the sequence of actions. The algorithm developed below to (partially) solve this optimization/learning problem will be analyzed later in the course.

1.3.2 Exploit-explore algorithm

If the horizon time T is relatively small, then the problem is not very interesting in the sense that the player does not have time to explore and learn about the reward probability of each arm. We therefore assume that T is sufficiently large, to make the problem more interesting.

It is clear that at the initial time $t = 1$, the player does not have any information on the reward probability, and should therefore randomly choose a first arm a_1 to start with. Then, at each time t, the player acquires more information by choosing an arm and getting a reward, and can therefore use this information to estimate the reward probability of this given arm. Then, with this estimation, she can decide if the player wants to continue playing this arm, or if the player wants to switch to another arm.

Given the form of the criteria to be maximized, the player is interested in estimating the expected reward of a given arm $a \in \{1,\dots,K\}$, *i.e.*, $q(a) := \mathbb{E}[R^{(a)}]$. Once again, at the beginning, the player has no information of this expectation. But, if the player decides to play arm a at time $t = 1$, *i.e.* $a_1 = a$, and gets the reward r_1, the player can update her beliefs on the reward probability of arm a. One possible estimation is to consider that $\mathbb{E}[R^{(a)}] \approx r_1$. Then, if the player decides to continue with arm a, *i.e.* $a_2 = a$, and gets the reward r_2, the player can update their beliefs on the reward probability of arm a through the natural approximation $\mathbb{E}[R^{(a)}] \approx (r_1 + r_2)/2$. If we continue this reasoning, we obtain the following estimation updating process.

Let $a \in \{1,\dots,K\}$, and denote by $\widehat{q}_t(a)$ the estimation at time t of the reward expectation $q(a)$ for arm a. If we follow the previous reasoning, we obtain the following natural and widely used estimate:

$$\widehat{q}_t(a) := \begin{cases} \widehat{q}_0(a), & \text{if } N_t(a) = 0, \\ \dfrac{1}{N_t(a)} \sum_{k=1}^{t} r_k \mathbb{1}_{a_k=a}, & \text{if } N_t(a) \geqslant 1, \end{cases} \qquad \text{where } N_t(a) := \sum_{k=1}^{t} \mathbb{1}_{a_k=a},$$

and $\mathbb{1}_A$ is the characteristic function of the set A, i.e. it is equal to one on the set A and zero outside. The initial estimate $\widehat{q}_0(a)$ can be set arbitrarily in $(0,1)$. Remark that, if as T tends to infinity $N_T(a)$ converges to infinity with probability one, then, by the law of large numbers we have:

$$\lim_{T\to+\infty} \widehat{q}_T(a) = \lim_{T\to+\infty} \frac{1}{N_T(a)} \sum_{k=1}^{T} r_k \mathbb{1}_{a_k=a} = q(a), \qquad \text{with probability one.}$$

In the previous example of a news website, we can imagine that for a news header a, the reward is 1 if the user clicks, and 0 otherwise. That is r_k takes values in $\{0,1\}$. Then when $N_t(a) \geqslant 1$,

$$\widehat{q}_t(a) := \frac{1}{N_t(a)} \sum_{k=1}^{t} r_k \mathbb{1}_{a_k=a}.$$

Therefore, $\widehat{q}_t(a)$ is the average number of clicks of the user when the header a is shown.

In the general setting, with this estimation in mind, the player can *exploit* by choosing the action a_t at time t that maximises $\{\widehat{q}_t(1),\dots,\widehat{q}_t(K)\}$, *i.e.* $a_t = a^\star$ with $a^\star := \arg\max_{a\in\{1,\dots,K\}} \widehat{q}_t(a)$. However, note that if we choose for example the initialization $\{\widehat{q}_0(1),\dots,\widehat{q}_0(K)\} = (0,\dots,0)$, and arbitrarily decide to play arm 1 at $t = 1$. If the reward r_1 is strictly positive, then we obtain the estimation $\widehat{q}_1(1) = r_1 > 0$. Therefore, if the player only *exploit*, the player will then always choose arm 1, and may never find the optimal arm: the player should also *explore* at some frequency, by choosing an arm that was never or little used.

The ε-Greedy algorithm incorporates some exploration times in the previous algorithm. More precisely, we can choose an exploration constant $\varepsilon \in (0,1)$, usually small, so that at each time t:

$$a_t = \begin{cases} \arg\max_{a\in\{1,\dots,K\}} \widehat{q}_t(a), & \text{with probability } 1-\varepsilon, \\ \widetilde{a}, & \text{with probability } \varepsilon, \end{cases}$$

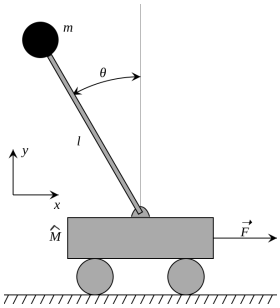
where $\widetilde{a}$ is a uniform choice on $\{1,\dots,K\}$ in order to explore other options. The parameter ε has to be chosen in order to implement this algorithm. Note that a first intuition is that it might be interesting to take it time-dependent, and more precisely decreasing to 0 as time passes. The main question is to determine if there is an optimal choice for ε.

1.4 Further Examples

We conclude this chapter by a brief discussion of three more examples.

1.4.1 Inverted Pendulum

The solution to this non-trivial engineering example showcases the power of the Linear Quadratic regulator studied in Chapter 2. For many years, it has served as a benchmark problem in robotics. The goal is to keep an inverted pendulum in the horizontal position by moving the cart on which it is located. Clearly this position is unstable and without any interference from the controller, any small disturbance would make it fall to its stable downward position. Mathematically, the state of the system is described by the position of the cart and the angle of the pendulum, while the control is the acceleration of the cart on which the pendulum is placed.



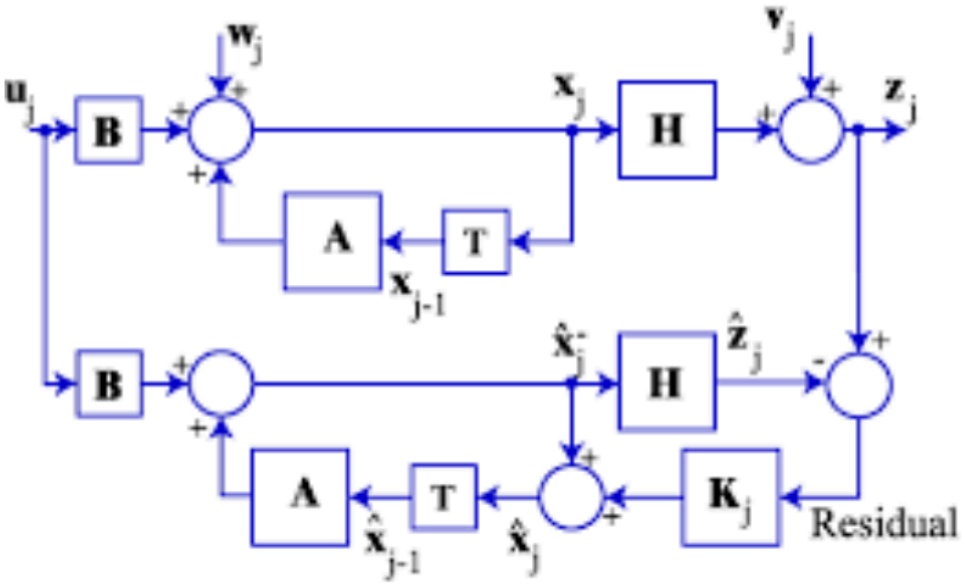
Broadly speaking, this problem is governed by the laws of mechanics, and we can therefore write a system of equations describing the dynamics of the system. Then, by using relevant approximations and discretizing the system with respect to time, we obtain a Linear Quadratic control problem. A more detailed description of this problem as well as its solution will be given in Chapter 2.

1.4.2 Autopilot

This is the example that historically motivated the Kalman filter. In fact, even today modern auto-pilots in the planes use a version of this filter. In the cruising altitude, the auto-pilot is given the task of keeping the plane on a desired trajectory. To achieve its goal, it measures the location of the plane periodically and compares it with the target trajectory. Based on this and its calculation of the future location of the plane at the next measurement time, it makes adjustments. However, the location measurement is subject to noise due to inaccuracies of the GPS and the future location calculation is also not accurate because of the random wind, other weather conditions and errors. Using its Kalman filter, the auto-pilot takes these into account and keeps the plane on its target trajectory with small adjustments. The Kalman filter is studied in Chapter 3.

1.4.3 Recycling Robot

This stylized example illustrates all the goals of this course. The main goal is to optimize the journey of a mobile robot. This robot runs on a rechargeable battery and can actively collect cans in an office on his onboard bin, wait for people to bring cans to it, or go at the charging station. However, the robot only has a rough estimate of the remaining battery, and if it runs out of battery, it has to be manually placed (by a non-robot person) at the charging station. To simplify, we will assume that the battery has two states: low and high. In each of the state, the robot should decide its optimal action, collect, wait, or go to the charging station, and this action will have an impact on the charging status of the battery. The aim is obviously to maximize the number of cans collected (a random reward) while avoiding being stuck without a battery outside the charging station, which would incur a significant penalty. This example will be studied in detail in Chapter 5.



2. Linear Quadratic Regulator - LQR

As discussed in the previous chapter, the theory of optimal control is concerned with operating a dynamic system at minimum cost. In this chapter, we study a particular optimal control problem in discrete time, called the *Linear Quadratic problem* (LQR). In this specific problem, the system dynamics are described by a set of linear difference equations and the cost is described by a quadratic function. One of the main results in the theory is that the solution is provided by the Linear Quadratic Regulator, a feedback controller whose equations will be explicitly formulated in this chapter. The first part of this chapter focuses on the deterministic case and the second part is devoted to the stochastic framework. In the latter framework, the state dynamics are random and are impacted by Gaussian random noise. This extended problem called the *Linear Quadratic Gaussian* (LQG) problem is solved by leveraging the optimal control of the deterministic problem, namely the LQR. In both cases the dynamics of the state and controlled variables are known, and as such there is no learning. An extension of the LGR requiring learning is studied in Chapter 4.

In many applications emanating from physics, it is natural to consider systems in continuous time. Although, we do not treat them directly, the formulation of these problems is quite similar to the one in discrete time with essentially the same solution.

Frequently used notation. In the above and what follows, we use the following notation: we sometimes consider a vector $a = (a_1, \dots, a_\ell) \in \mathbb{R}^\ell$ as a $1 \times d$ matrix and write $a^\top$ for its transpose. For an $\ell \times \ell$ matrix $C = (C_{ij})_{i,j=1,\dots,\ell}$ we write

$$a^\top C a = Ca \cdot a = \sum_{j=1}^{\ell} (Ca)_j \, a_j = \sum_{j=1}^{\ell} \sum_{i=1}^{\ell} C_{ij} a_i a_j.$$

2.1 The deterministic problem

We start our analysis with the (discrete-time) *deterministic* Linear Quadratic problem. That is, the dynamics of the state process follows a difference equation which is not corrupted by noise.

2.1.1 Formulation

As in the general deterministic control problem in discrete time, we assume that the controlled state is summarized by a deterministic sequence of vectors, $x_0, x_1, \ldots \in \mathbb{R}^d$, called the (controlled) *state process*, and that the controllers (or the decision makers) can influence its evolution through a sequence of vectors $u_0, u_1, \ldots \in \mathbb{R}^\ell$, called the *control process*. Their goal is to minimize a given cost. The difference with the general optimal control model lies in the specification that in LQR, the dynamics of the state process are linear and the cost is quadratic.

Formulation of the Linear Quadratic problem.
Given a known initial condition x_0, the evolution of the state is governed by the linear equation,

$$x_{k+1} = Ax_k + Bu_k, \qquad k = 0, 1, \ldots, \tag{2.1}$$

where A and B are known matrices of size $d \times d$ and $d \times \ell$, respectively. For given symmetric, positive definite matrices M, $\widehat{M}$ and N of size $d \times d$, $d \times d$ and $\ell \times \ell$, respectively, the goal of the controller is to optimally choose the control sequence $\mathbf{u} := (u_0, u_1, \ldots)$ so as to minimize one of the following two costs:

(i) (in the *infinite horizon* case),

$$J_\infty(x_0, \mathbf{u}) := \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k \right), \tag{2.2}$$

where $\rho > 0$ is the *discount factor* and is usually strictly less than one.

(ii) (in the *finite horizon* case),

$$J_n(x_0, \mathbf{u}) := \sum_{k=0}^{n-1} \left(\mathbf{x}_k^\top M x_k + u_k^\top N u_k \right) + x_n^\top \widehat{M} x_n, \tag{2.3}$$

with an arbitrary *horizon* $n \geqslant 1$.

R Given an initial condition x_0, once the decision makers decide on the controls, the state can be computed recursively using equation (2.1). Importantly, the value of the state x_k depends only on the controls up to k, *i.e.* $u_0, u_1, \ldots, u_{k-1}$. Moreover, the control variable u_k has to be chosen at time k, given all the information available at time k, *i.e.* u_k may depend only on $x_0, \ldots, x_k$ and $u_0, \ldots, u_{k-1}$. Actually, in this deterministic case, the information provided by $u_0, \ldots, u_{k-1}$ is the same as the one provided by $x_0, \ldots, x_k$, but this assumption is central in the stochastic case, and imposes a natural structural restriction that the controllers are not informed about the future.

The Linear Quadratic problem can actually be seen as an appropriate approximation of more general problems, introduced in the previous chapter. Indeed, if we assume that the system is operating near one of its equilibrium points, and that the equations can be linearized around this point, we obtain a system of linear equations. The next step is to discretize time and approximate the differential equations by finite difference methods. This procedure is carried out in more details in the example of the inverted pendulum in Chapter 2.3.1 below. The assumption that the cost has to be of a quadratic form is also a simplification, but is not restrictive in many cases. Indeed, in the engineering-related applications, the cost function is often defined as a sum of the deviations of key measurements, like altitude or process temperature, from their desired values. The goal is usually to find the optimal control that minimize undesired deviations. The magnitude of the control action itself may also be included in the cost function. Therefore, since the interesting measure or cost in these applications corresponds to a deviation, it is appropriate to consider a quadratic cost.

2.1.2 A simple illustrative example

Before describing the general approach to solve this particular type of control problems, we consider the following simple *one-dimensional* example.

▪ **Example 2.1 — Infinite Horizon.** Let $d = \ell = 1$ and assume that the evolution of the state is governed by the equation (2.1) with $A = 4$ and $B = 1$, *i.e.*

$$x_{k+1} = 4x_k + u_k, \qquad k = 1, 2, \dots,$$

with initial condition $x_0 = 1$. The objective function in this infinite horizon case is given by (2.2) with $\rho = 1/2$ and $M = N = 1$, *i.e.* the problem is to minimize the following cost functional:

$$J_\infty(x_0, \mathbf{u}) := \sum_{k=0}^{\infty} 2^{-k} \left[x_k^2 + u_k^2\right], \tag{2.4}$$

over all sequences $\mathbf{u} = (u_0, u_1, \dots)$, subject to the constraints $x_0 = 1$ and that $x_{k+1} = 4x_k + u_k$.

Towards a solution, we observe that one can directly compute the values of x_k for any $k \in \mathbb{N}$ as functions of the control:

$$x_1 = 4 + u_0, \quad x_2 = 4x_1 + u_1 = 16 + 4u_0 + u_1, \quad x_3 = 4x_2 + u_2 = 64 + 16u_0 + 4u_1 + u_2, \ \dots$$

In full generality, the value of the controlled state is given by the following recursive relation (its proof follows from mathematical induction):

$$x_{k+1} = 4x_k + u_k = 16x_{k-1} + 4u_{k-1} + u_k = \dots = 4^{k+1} + \sum_{i=0}^{k} 4^{k-i} u_i.$$

Notice that without any control (i.e., $u_k = 0$ for every k), we obtain $x_k = 4^k$ for all k, and the system explodes as k goes to infinity. In many examples, this is not desirable and the control is used precisely to stabilize the state. In this setting, if we choose for example $u_0 = -4$ and then $u_k = 0$ for $k \geqslant 1$, we then obtain $x_k = 0$ for every $k \geqslant 1$. There are other choices as well: for instance, with the control $u_k = -3.5x_k$, one obtains

$$x_{k+1} = 4x_k + u_k = \frac{1}{2} x_k = 2^{-(k+1)}, \ k \in \mathbb{N}.$$

This choice of control brings the state to zero slower but it uses "less" control.

Controls that are given functions of the current state as in the above example (*i.e.* $u_k = -3.5x_k$) are called *feedback controls*. In general, they are not necessarily linear functions. In higher dimensions, the complete controllability of the above type may not be always possible and depends on the interaction between the matrices A and B. This is further discussed in Chapter 2.3. In general, the optimal choice for the control depends on the objective functional. In the current example, the goal is to minimize the cost given by (2.4).

We postulate here that the optimal solution is a linear feedback control, *i.e.* $u_k = -fx_k$ for an appropriate constant $f \in \mathbb{R}$ to be computed or optimized. A rigorous justification of this linear choice is given later when we discuss the general theory. Under this assumption on the optimal control, the evolution of the state solves the following equation:

$$x_{k+1} = 4x_k + u_k = 4x_k - fx_k = (4 - f)x_k, \quad k = 0, 1, \dots,$$

with $x_0 = 1$. Hence, one can directly show that $x_k = (4-f)^k$ and $u_k = -f(4-f)^k$ for all k, leading to the following associated cost, for all $f \in \mathbb{R}$:

$$v(f) := \sum_{k=0}^{\infty} 2^{-k}\left[x_k^2 + u_k^2\right] = \sum_{k=0}^{\infty} 2^{-k}(1+f^2)(4-f)^{2k} = (1+f^2)\sum_{k=0}^{\infty}\left(\frac{(4-f)^2}{2}\right)^k.$$

In order to have a finite cost, the parameter f must satisfy $4-\sqrt{2} < f < 4+\sqrt{2}$. Then, for these value of f,

$$v(f) = \frac{(1+f^2)}{1-\frac{(4-f)^2}{2}} = \frac{2(1+f^2)}{8f-f^2-14}.$$

Indeed, the function f is the infinite sum of a geometric series, with *coefficient* $a := (1+f^2)$ and *common ratio* $r := (4-f)^2/2$. Therefore, the sum is finite if and only if $|r| < 1$, *i.e.* $(4-f)^2/2 < 1$, and its value is given by

$$v(f) = \frac{a}{1-r} = \frac{2(1+f^2)}{-f^2+8f-14}.$$

Its graph with respect to f is given by Figure 2.1, and through a numerical minimization, we obtain $f^\star \approx 3.53305$. One can finally check that this optimal $f^\star$ is indeed in the required interval. In the next subsection, we derive a general formula for the optimal control which agrees with this value.

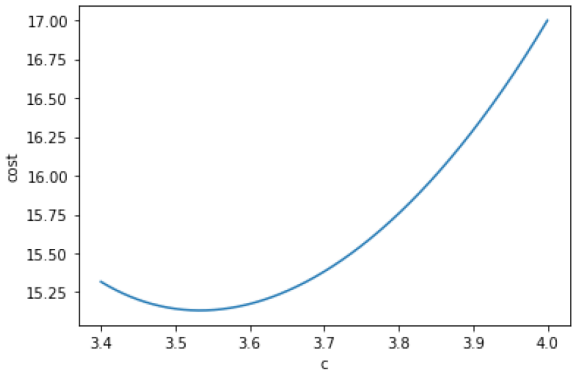


Figure 2.1: Graph of the cost function v to be minimized.

■

We next study the finite horizon analogue of the above example.

■ **Example 2.2 — Finite Horizon.** For a given horizon $n \geqslant 1$, we consider a similar problem with discount factor $\rho = 1$:

$$\text{minimize } J_n(x_0, \mathbf{u}) := \sum_{k=0}^{n-1} \left[x_k^2 + u_k^2\right],$$

subject to $x_{k+1} = 4x_k + u_k$ with initial condition $x_0 = 1$. Since the problem is time-inhomogenous, the optimal feedback control varies in time. We solve this problem inductively in n.
Case $n = 1$. This is a simple optimization problem. Since final cost is zero, the optimal control is also zero and the value function is

$$v_1(x_0) = \inf_{u_0} J_1(x_0, u_0) = x_0^2.$$

Case $n = 2$. The problem is

$$\begin{aligned}\inf_{u_0,u_1} \left\{[x_0^2+u_0^2]+[x_1^2+u_1^2]\right\} &= \inf_{u_0}\left\{[x_0^2+u_0^2]+\left(\inf_{u_1}[x_1^2+u_1^2]\right)\right\} = \inf_{u_0}\left\{[x_0^2+u_0^2]+x_1^2\right\} \\ &= \inf_{u_0}\left\{[x_0^2+u_0^2]+(4x_0+u_0)^2\right\}.\end{aligned}$$

Then, the minimizers are $u_0^* = -2x_0$ and $u_1^* = 0$. Moreover,

$$v_2(x_0) = \inf_{u_0,u_1} J_2(x_0, u_0, u_1) = 9x_0^2.$$

We postulate that $v_n(x_0) = v_n(x_0)^2$ for some constant v_n. Then,

$$v_{n+1}(x_0) = \inf_{u_0,\dots,u_n} \left\{ \sum_{k=0}^{n-1} 2^{-k} \left[x_k^2 + u_k^2\right] \right\} = \inf_{u_0} \left\{ \left[x_0^2 + u_0^2\right] + \inf_{u_1,\dots,u_n} \left(\sum_{k=1}^{n-1} 2^{-k} \left[x_k^2 + u_k^2\right] \right) \right\}$$
$$= \inf_{u_0} \left\{ \left[x_0^2 + u_0^2\right] + v_n x_1^2 \right\} = \inf_{u_0} \left\{ \left[x_0^2 + u_0^2\right] + v_n (4x_0 + u_0)^2 \right\}.$$

Hence, the minimizer is $u_0^* = -4v_n x_0/(1+v_n)$. We can also obtain a recursive equation for the constants v_n. This approach is *dynamic programming* developed in the next subsection. ∎

2.2 General approach

We now formulate a general approach for this type of problems. We only provide full details of the results in the infinite horizon case and outline the results for the finite horizon case without proof.

2.2.1 Dynamic Programming

Consider the Linear Quadratic control problem in infinite horizon. Then, the goal of the controller is to optimally choose the control process $\mathbf{u} = (u_0, u_1, \dots) \in \mathscr{U}$, *i.e.* a sequence of vectors $u_k \in \mathbb{R}^\ell$, so as the to minimize the cost (2.3):

$$J_\infty(x_0, \mathbf{u}) := \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k \right),$$

where the evolution of the state is governed by the linear equation (2.1):

$$x_{k+1} = Ax_k + Bu_k, \qquad k \in \mathbb{N}.$$

We assume here that there is no restriction on the process $\mathbf{u}$, apart from the obvious condition that the total cost $J_\infty(x_0, \mathbf{u})$ should be finite. Any such control process $\mathbf{u}$ is called *admissible* and $\mathscr{U}$ is the set of all admissible controls.

The general approach utilizes the *value function* v_∞ associated to the control problem which is the optimal value as a function of the initial state value. It is defined for all $x_0 \in \mathbb{R}^d$ as follows:

$$v_\infty(x_0) := \inf_{\mathbf{u} \in \mathscr{U}} J_\infty(x_0, \mathbf{u}) = \inf_{\mathbf{u} \in \mathscr{U}} \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k \right).$$

The *Dynamic Programming Principle* developed by Richard Bellman as stated in the below theorem is a powerful tool with an amazing range of applications.

Theorem 2.2.1 — Dynamic Programming, infinite horizon. Assume that v_∞ is finite. Then,

$$v_\infty(x_0) = x_0^\top M x_0 + \inf_{u_0 \in \mathbb{R}^\ell} \left\{ u_0^\top N u_0 + \rho v_\infty(Ax_0 + Bu_0) \right\}, \qquad \forall x_0 \in \mathbb{R}^d. \tag{2.5}$$

Proof. For a given control $\mathbf{u} \in \mathscr{U}$, set $\widehat{\mathbf{u}} := (u_1, u_2, \dots)$, i.e., $\widehat{\mathbf{u}}$ is the original control without the first element $u_0 \in \mathbb{R}^\ell$. Then,

$$J_\infty(x_0, \mathbf{u}) = x_0^\top M x_0 + u_0^\top N u_0 + \sum_{k=1}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k \right)$$
$$= x_0^\top M x_0 + u_0^\top N u_0 + \sum_{k=0}^{\infty} \rho^{k+1} \left(x_{k+1}^\top M x_{k+1} + u_{k+1}^\top N u_{k+1} \right)$$
$$= x_0^\top M x_0 + u_0^\top N u_0 + \rho J_\infty(x_1, \widehat{\mathbf{u}}).$$

Therefore, by taking the infimum over $\mathbf{u} \in \mathscr{U}$, one obtains:

$$\begin{aligned} v_\infty(x_0) &:= \inf_{\mathbf{u}\in\mathscr{U}} J_\infty(x_0,\mathbf{u}) = \inf_{\mathbf{u}\in\mathscr{U}} \left\{x_0^\top M x_0 + u_0^\top N u_0 + \rho J_\infty(x_1,\widehat{\mathbf{u}})\right\} \\ &= x_0^\top M x_0 + \inf_{\mathbf{u}\in\mathscr{U}} \left\{u_0^\top N u_0 + \rho J_\infty(x_1,\widehat{\mathbf{u}})\right\} \\ &= x_0^\top M x_0 + \inf_{u_0\in\mathbb{R}^\ell} \left\{u_0^\top N u_0 + \rho \inf_{\widehat{\mathbf{u}}\in\mathscr{U}} J_\infty(x_1,\widehat{\mathbf{u}})\right\} \\ &= x_0^\top M x_0 + \inf_{u_0\in\mathbb{R}^\ell} \left\{u_0^\top N u_0 + \rho v_\infty(x_1)\right\}, \end{aligned}$$

which concludes the proof. ■

The dynamic programming equation finds the balance between the immediate cost $u^\top N u$ of using some control $u \in \mathbb{R}^\ell$ and the future cost when starting at the new point $x_1 = Ax_0 + Bu$, summarized by $v_\infty(x_1)$. This observation also explains the reason for computing the best value v_∞ not only at the given initial point x_0 but at all other points that could be potentially visited by the state process. The value $v_\infty(x)$ gives a "quality measure" for state x and the decision maker must use this information to decide whether or not to visit this state. In fact, in modern approaches to games like Chess or Go and more generally in Q-learning, an essential ingredient of the algorithm is to compute an effective quality measure of each board configuration. The value function is the simplest analogue of this quality measure.

2.2.2 Optimal feedback control

The dynamic programming equation described above is also useful in constructing optimal controls that are in feedback form. Indeed, consider the map $u^\star : \mathbb{R}^d \mapsto \mathbb{R}^\ell$ defined by

$$u^\star(x) \in \operatorname*{arg\,min}_{u\in\mathbb{R}^\ell} \left\{u^\top N u + \rho v_\infty(Ax+Bu)\right\}, \quad x \in \mathbb{R}^d.$$

In other words, the map $u^\star$ is the unique function satisfying,

$$\min_{u\in\mathbb{R}^\ell} \left\{u^\top N u + \rho v_\infty(Ax+Bu)\right\} = \big(u^\star(x)\big)^\top N u^\star(x) + \rho v_\infty\big(Ax + Bu^\star(x)\big), \quad x \in \mathbb{R}^d.$$

Let $x^\star$ be the controlled process corresponding to this control, *i.e.*,

$$x^\star_{k+1} = Ax^\star_k + Bu^\star(x^\star_k), \quad k \in \mathbb{N},$$

with initial data $x^\star_0 = x_0$. Finally, for $k \geqslant 0$, set $u^\star_k := u^\star(x^\star_k)$ and consider the control process $\mathbf{u}^\star := (u^\star_0, u^\star_1, \ldots)$.

Theorem 2.2.2 — Optimal Feedback Controls. Assume that the value function is finite. Then, the control $\mathbf{u}^\star := (u^\star_0, u^\star_1, \ldots)$ constructed as above is optimal, i.e,

$$v_\infty(x_0) := \inf_{\mathbf{u}\in\mathscr{U}} J(x_0,\mathbf{u}) = J_\infty(x_0,\mathbf{u}^\star).$$

Proof. By the dynamic programming equation and the construction of $\mathbf{u}^\star$,

$$v_\infty(x_0) = x_0^\top M x_0 + \inf_{u_0\in\mathbb{R}^\ell} \left\{u_0^\top N u_0 + \rho v_\infty(Ax_0+Bu_0)\right\} = x_0^\top M x_0 + \big(u^\star_0\big)^\top N u^\star_0 + \rho v_\infty\big(x^\star_1\big).$$

Similarly,

$$v_\infty\big(x^\star_1\big) = \big(x^\star_1\big)^\top M x^\star_1 + \big(u^\star_1\big)^\top N u^\star_1 + \rho v_\infty\big(x^\star_2\big).$$

We substitute this into the previous identity to obtain the following,

$$v_\infty(x_0) = x_0^\top M x_0 + (u_0^\star)^\top N u_0^\star + \rho\left((x_1^\star)^\top M x_1^\star + (u_1^\star)^\top N u_1^\star\right) + \rho^2 v_\infty(x_2^\star).$$

In fact, the dynamic programming equation (2.5) with $x = x_k^\star$ together with the previous construction of $\mathbf{u}^\star$ give the following relation, for any $k \in \mathbb{N}$:

$$v_\infty(x_k^\star) = (x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star + \rho v_\infty(x_{k+1}^\star).$$

We now iterate this procedure to obtain (proof by mathematical induction), for any $n \in \mathbb{N}^\star$,

$$\begin{aligned}
v_\infty(x_0) &= \sum_{k=0}^{1} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right) + \rho^2 v_\infty(x_2^\star) \\
&= \sum_{k=0}^{1} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right) + \rho^2 \left((x_2^\star)^\top M x_2^\star + (u_2^\star)^\top N u_2^\star + \rho v_\infty(x_3^\star)\right) \\
&= \sum_{k=0}^{2} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right) + \rho^3 v_\infty(x_3^\star) = \dots \\
&= \sum_{k=0}^{n} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right) + \rho^{n+1} v_\infty(x_{n+1}^\star).
\end{aligned}$$

Using the fact that $v_\infty \geqslant 0$ (since J_∞ is a quadratic cost), we obtain for every n:

$$v_\infty(x_0) = \sum_{k=0}^{n} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right) + \rho^{n+1} v_\infty(x_{n+1}^\star) \geqslant \sum_{k=0}^{n} \rho^k \left((x_k^\star)^\top M x_k^\star + (u_k^\star)^\top N u_k^\star\right),$$

We then let n go to infinity to conclude that $v_\infty(x_0) \geqslant J_\infty(x_0, \mathbf{u}^\star)$. Since the value function is assumed to be finite, we conclude that $\mathbf{u}^\star \in \mathscr{U}$. Hence,

$$v_\infty(x_0) = \inf_{\mathbf{u} \in \mathscr{U}} J(x_0, \mathbf{u}) \leqslant J_\infty(x_0, \mathbf{u}^\star).$$

■

2.2.3 Riccati equation

When the value function is finite the value function is of the form

$$v(x) = x^\top V x, \; x \in \mathbb{R}^d, \tag{2.6}$$

for some symmetric, positive definite matrix V. This is proved in Theorem 2.3.2 below. Then, the dynamic programming equation yields an equation satisfied by this matrix.

Theorem 2.2.3 — Homogeneous Riccati Equation. Consider the following *Riccati equation*

$$V = M + \rho A^\top V A - \rho^2 A^\top V B \left(N + \rho B^\top V B\right)^{-1} B^\top V A. \tag{2.7}$$

Suppose that the above equation has a solution, then the value function is given by (2.6). Moreover, the following linear feedback control, called linear quadratic regulator (LQR),

$$u_k^\star = -F x_k^\star, \qquad \text{where} \qquad F = \rho\left(N + \rho B^\top V B\right)^{-1} B^\top V A,$$

is optimal. The $l \times d$ matrix F is called the gains matrix.

We leave the direct proof to the reader. As highlighted in this theorem, the Riccati equation (2.7) does not always have a solution, but if a solution exists, then it is unique. Moreover, a necessary and sufficient condition for existence of a solution is the finiteness of the value function v_∞. This in turn is equivalent to a controllability condition between the matrices A and B, which is further discussed in Chapter 2.3.

Exercise 2.1 In the general setting, assume that v_∞ is finite. Use the dynamic programming equation, to derive the Riccati equations. ∎

R Let V be a solution for the quadruple (A,B,M,N) with $\rho > 0$. Define $A_\rho := \sqrt{\rho}A$ and $B_\rho := \sqrt{\rho}B$. Then V is also a solution for the quadruple (A_ρ,B_ρ,M,N) with discount factor equal to one. In this case, the $d \times l$ *gains matrix* F is given by

$$F = \left(N + B_\rho^\top V B_\rho\right)^{-1} B_\rho^\top V A_\rho.$$

Exercise 2.2 Prove the above remark. ∎

Solution. Let V be a solution for the quadruple (A,B,M,N) with $\rho > 0$. Define $A_\rho := \sqrt{\rho}A$ and $B_\rho := \sqrt{\rho}B$. Then,

$$\begin{aligned} V &= M + \rho A^\top V A - \rho^2 A^\top V B\left(N + \rho B^\top V B\right)^{-1} B^\top V A \\ &= M + \rho A^\top V A - \rho A^\top V B\left(N + \rho B^\top V B\right)^{-1} \rho B^\top V A \\ &= M + A_\rho^\top V A_\rho - A_\rho^\top V B_\rho \left(N + B_\rho^\top V B_\rho\right)^{-1} B_\rho^\top V A_\rho. \end{aligned}$$

Therefore, V is also a solution for the quadruple (A_ρ,B_ρ,M,N) with discount factor equal to one. So instead of solving the Riccati equation for a given ρ, we can define A_ρ, B_ρ as above and solve the equation with discount factor equal to one. Then, the $d \times l$ *gains matrix* F is given by

$$F = \rho\left(N + \rho B^\top V B\right)^{-1} B^\top V A = \left(N + B_\rho^\top V B_\rho\right)^{-1} B_\rho^\top V A_\rho.$$

Exercise 2.3 Consider the one dimensional example Chapter 2.1 studied in Chapter 2.1.2. Assuming that the value function is of the form $v_\infty(x) = Vx^2$ for all $x \in \mathbb{R}$ and some V to be determined, use the dynamic programming equation (2.5) to derive the Riccati equation for V. Compute V and the optimal feedback control. Show that the solution coincides with Chapter 2.2.3 and with the results previously obtained in Chapter 2.1.2. ∎

Solution. Recall that the goal in Chapter 2.1 is to minimize

$$J_\infty(x_0,\mathbf{u}) := \sum_{k=0}^{\infty} 2^{-k}\left(x_k^2 + u_k^2\right), \quad \text{subject to} \quad x_{k+1} = 4x_k + u_k,\ k \in \mathbb{N},\ x_0 \in \mathbb{R}^d.$$

We postulate that the value function $v_\infty(x) := \min_{\mathbf{u}\in\mathscr{U}} J(x,\mathbf{u})$ is of the form $v_\infty(x) = Vx^2$ for all $x \in \mathbb{R}$, for some constant $V \in \mathbb{R}$ to be determined. The corresponding dynamic programming equation (2.5) in this example ($A = 4, B = 1, M = 1, N = 1, \rho = 0.5$) is given by,

$$v_\infty(x) = x^2 + \inf_{u\in\mathbb{R}}\left\{u^2 + \frac{1}{2}v_\infty(4x+u)\right\}, \qquad \forall x \in \mathbb{R}.$$

Using the ansatz $v_\infty(x) = vx^2$, one obtains:

$$\begin{aligned} Vx^2 &= x^2 + \inf_{u\in\mathbb{R}}\left\{u^2 + \frac{1}{2}V(4x+u)^2\right\} = x^2 + \inf_{u\in\mathbb{R}}\left\{u^2 + \frac{1}{2}V\left(16x^2+u^2+8xu\right)\right\} \\ &= x^2 + 8Vx^2 + \inf_{u\in\mathbb{R}}\left\{\left(1+\frac{V}{2}\right)u^2 + 4Vxu\right\} \end{aligned}$$

Computing the first and second order conditions, the infimum is attained for

$$u = -\frac{4V}{2+V}x,$$

meaning that the optimal feedback control $\mathbf{u}^\star$ is given by $u_k^\star = -Fx_k^\star$, $k\in\mathbb{N}$, with $F = \frac{4V}{2+V}$. By computing the value of the minimum, and simplifying by x^2, we obtain the following Riccati equation for the constant V:

$$V = 1 + 8V - \frac{8V^2}{2+V} = 1 + \frac{16V}{2+V}.$$

The above equations are exactly as in Theorem 2.2.3. Indeed, in this example

$$\begin{aligned} V &= M + \rho A^\top VA - \rho^2 A^\top VB\left(N+\rho B^\top VB\right)^{-1}B^\top VA = 1 + 8V - 4V^2\left(1+V/2\right)^{-1}, \\ F &= \rho\left(N+\rho B^\top VB\right)^{-1}B^\top VA = 2V\left(1+V/2\right)^{-1}, \end{aligned}$$

with $A=4, B=1, M=1, N=1, \rho=0.5$. Then, the above Riccati equation is equivalent to solving the following second-order equation:

$$V^2 - 15V - 2 = 0.$$

Since the value function is positive, we choose the positive solution:

$$V = \frac{15+\sqrt{233}}{2}, \qquad \text{and} \qquad F = -\frac{4V}{2+V} = -\frac{60+4\sqrt{233}}{19+\sqrt{233}} \approx -3.533.$$

This is exactly the value we obtained earlier.

We now consider the following example:

■ **Example 2.3** Consider the equation

$$y_{k+2} + y_k = u_k, \quad k\in\mathbb{N},$$

with given initial data $(y_0, y_1)\in\mathbb{R}^2$ and control $\mathbf{u} := (u_0, u_1, \ldots)$ with $u_k\in\mathbb{R}$ for all $k\in\mathbb{N}$. The cost function to be minimized is defined by

$$J_\infty(y_0, y_1, \mathbf{u}) = \sum_{k=0}^{\infty}\left(y_k^2 + y_{k+1}^2 + u_k^2\right),$$

■

Exercise 2.4 By using the state process $x_k = (y_k, y_{k+1})^\top$, show that the above problam can be rewritten as a standard LQ problem. Then, solve it using the dynamic programming approach developed in Chapter 2.2.3. ■

Solution. The first step is to rewrite this optimization problem as a standard Linear-Quadratic problem. We first set for all $k \in \mathbb{N}$,

$$x_k = \begin{pmatrix} y_k \\ y_{k+1} \end{pmatrix}, \qquad x_0 = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}.$$

Then, for all $k \in \mathbb{N}$,

$$x_{k+1} = \begin{pmatrix} y_{k+1} \\ y_{k+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} y_k \\ y_{k+1} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_k.$$

Hence, the state process $x_k \in \mathbb{R}^2$ solves the state equation (2.1) with matrices

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Similarly, the cost function can be written in the usual form as well. Indeed,

$$J_\infty(x_0, \mathbf{u}) = \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k \right),$$

where M is the 2×2 identity matrix, $N = 1$ and $\rho = 1$.

We can now use Chapter 2.2.3 to solve this standard LQ problem. Let the value function $v_\infty(x) := \min_{\mathbf{u} \in \mathscr{U}} J(x, \mathbf{u})$ be of the form $v_\infty(x) = x^\top V x$ for all $x \in \mathbb{R}^2$, for a given symmetric matrix V to be determined:

$$V := \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix}.$$

Then, the Riccati equation (2.7) takes the following form

$$\begin{aligned} V &= M + \rho A^\top V A - \rho^2 A^\top V B \left(N + \rho B^\top V B \right)^{-1} B^\top V A \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} V \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} - a^{-1} L^\top L \end{aligned}$$

with

$$L := B^\top V A = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} -V_{22} & V_{12} \end{pmatrix}$$

and

$$a := N + \rho B^\top V B = 1 + \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1 + V_{22}.$$

Therefore, the Riccati equation reduces to:

$$\begin{aligned} \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} - \frac{1}{1+V_{22}} \begin{pmatrix} -V_{22} \\ V_{12} \end{pmatrix} \begin{pmatrix} -V_{22} & V_{12} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} V_{22} & -V_{12} \\ -V_{12} & V_{11} \end{pmatrix} - \frac{1}{1+V_{22}} \begin{pmatrix} V_{22}^2 & -V_{12}V_{22} \\ -V_{12}V_{22} & V_{12}^2 \end{pmatrix} \end{aligned}$$

This leads to the following system of three nonlinear equations:

$$\begin{cases} V_{11} &= 1 + V_{22} - V_{22}^2 (1+V_{22})^{-1}, \\ V_{12} &= -V_{12} + V_{12} V_{22} (1+V_{22})^{-1}, \\ V_{22} &= 1 + V_{11} - V_{12}^2 (1+V_{22})^{-1}. \end{cases}$$

If $V_{12} \neq 0$, then the second equation implies $V_{22} = -2$, and the first one $V_{11} = 3$. However, the third equation yields $0 \leqslant V_{12}^2 = -6 < 0$ which is clearly not admissible. Therefore, $V_{12} = 0$, which implies $V_{22} = 1 + V_{11}$ by the last equation, and replacing in the first equation we obtain $V_{11}^2 = 3$. Hence, *i.e.* $V_{11} = \sqrt{3}$, $V_{22} = 1 + \sqrt{3}$, and

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V_{12} & V_{22} \end{pmatrix} = \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1+\sqrt{3} \end{pmatrix}.$$

Finally, 1×2 gains matrix F is given by,

$$F = \rho\left(N + \rho B^\top V B\right)^{-1} B^\top V A = a^{-1} L = \frac{1}{1+V_{22}}\begin{pmatrix} -V_{22} & V_{12} \end{pmatrix} = \begin{pmatrix} -\dfrac{1+\sqrt{3}}{2+\sqrt{3}}, & 0 \end{pmatrix}.$$

We can also solve Riccati equation using the python package *scipy.linalg*. The function is *linalg.solve_discrete_are* (here, *are* stands for *Algebraic Riccati Equations*). It takes the matrices A, B, M, N as inputs, but ρ is always taken to be one. In view of Exercise 2.2.3, other values of ρ can also be solved by this function after multiplying A, B by $\sqrt{\rho}$. Using these observations, we obtain the following solution:

$$V \approx \begin{pmatrix} 1.732 & 0 \\ 0 & 2.732 \end{pmatrix} \qquad F \approx \begin{pmatrix} -0.732 & 0 \end{pmatrix}.$$

An example of a Python code is as follows:

```
import numpy as np
from scipy import linalg as la

# Parameters
A = np.array([[0, 1], [-2, 0]])
B = np.array([[0], [1]])
M = np.array([[1, 0], [0,2]])
N = np.array([[1]])

# The line below computes the solution of the Riccati equation
V = la.solve_discrete_are(A, B, M, N)

# We use the formula from the lectures to compute the gain matrix F
N0=N+B.T.dot(V.dot(B))
F=la.inv(N0).dot(B.T.dot(V.dot(A)))
np.set_printoptions(precision=4)

print('The solution to the Riccati equation V is ', np.around(V,3))
print('The gains matrix F is ',np.around(F,7))
```

Once the solution is computed, one may also solve for the resulting optimal trajectory.

2.2.4 Finite Horizon

In this section, we consider the control of the linear system dynamics given in (2.1) before a finite but arbitrary horizon $n \in \mathbb{N}^\star$. Recall that in this setting, the general objective function (with $\rho = 1$) is defined by the following finite sum:

$$J_n(x_0, \mathbf{u}) := \sum_{k=0}^{n-1} \left(x_k^\top M x_k + u_k^\top N u_k \right) + x_n^\top \widehat{M} x_n.$$

Recall that M, N are as before and $\widehat{M}$ is another positive symmetric matrix. With a slight abuse of notation, we also define $J_0(x, \mathbf{u}) = x^\top \widehat{M} x$ for any control $\mathbf{u}$ and $x \in \mathbb{R}^d$. Note that only the control values before n, *i.e.* $(u_0, u_1, \dots, u_{n-1})$, are relevant for the problem with finite horizon n. The value function at any point $x \in \mathbb{R}^d$ is then defined by

$$v_n(x) := \inf_{\mathbf{u} \in \mathscr{U}} J_n(x, \mathbf{u}).$$

As in the infinite horizon problem, the following dynamic programming equation holds.

Theorem 2.2.4 — Dynamic Programming, finite horizon. For all $n \in \mathbb{N}$, we have:

$$v_{n+1}(x) = x^\top M x + \inf_{u \in \mathbb{R}^\ell} \left\{ u^\top N u + v_n(Ax + Bu) \right\}, \qquad \forall x \in \mathbb{R}^d. \tag{2.8}$$

Moreover, $v_0(x) = x^\top \widehat{M} x$.

In this setting, the optimal feedback control is now a function of state and time:

$$u^\star(k, x) \in \arg\min_{u \in \mathbb{R}^\ell} \left\{ u^\top N u + v_k(Ax + Bu) \right\}, \qquad x \in \mathbb{R}^d,\ k = 0, 1, 2, \dots$$

Therefore, given a fix horizon $n \in \mathbb{N}$, we need to construct the optimal state process $x^\star$ recursively, starting at $x_0^\star = x_0$. We claim that the optimal control at step k and point $x_k^\star$ is given by:

$$u_k^\star := u^\star(n - k - 1, x_k^\star), \quad k = 0, 1, \dots, n-1.$$

The iterative process to determine te evolution of the optimal state $x^\star$ is then given by:

$$x_{k+1}^\star = A x_k^\star + B u_k^\star, \qquad k = 0, 1, \dots, n-1.$$

Note that $u^\star$ depends on n but this dependence is suppressed in the notation.

Theorem 2.2.5 — Optimal Feedback Controls. For any horizon n, the control processes $\mathbf{u}^\star := (u_0^\star, u_1^\star, \dots, u_{n-1}^\star)$ constructed as above is optimal, i.e,

$$v_n(x) := \inf_{\mathbf{u} \in \mathscr{U}} J_n(x, \mathbf{u}) = J_n(x, \mathbf{u}^\star).$$

We again postulate that

$$v_n(x) = x^\top V_n x, \qquad x \in \mathbb{R}^d.$$

We then use dynamic programming to compute the $d \times d$ positive definite, symmetric matrix V_n. A tedious but otherwise routine calculation yields the following.

Theorem 2.2.6 — Inhomogeneous Riccati equation. For a fixed horizon n, the optimal solution of the linear quadratic regulator in finite horizon is given by

$$u_k^\star = -F_{n-k-1} x_k^\star, \qquad \text{where} \qquad F_k = \left(N + B^\top V_k B\right)^{-1} B^\top V_k A,$$

for all $k = 0, 1, \dots, n-1$. Starting with the initial data $V_0 = \widehat{M}$, $d \times d$ symmetric matrices V_k are computed recursively, as solution of the inhomogeneous Riccati equation

$$V_{k+1} = M + A^\top V_k A - A^\top V_k B \left(N + B^\top V_k B\right)^{-1} B^\top V_k A, \qquad k = 0, 1, \dots, n-1. \tag{2.9}$$

As opposed to the homogenous Riccati equation (2.7), the above inhomogeneous equation (2.9) always have a unique solution.

Exercise 2.5 Use the dynamic programming equation to derive the Riccati equations. ■

Observe that the Riccati matrices V_k are independent of the horizon and can be computed once the matrices $A, B, M, N, \widehat{M}$ are given. Therefore, the process for solving a LQ problem with finite horizon $n \in \mathbb{N}$ is the following:

(i) Compute first the $d \times d$ symmetric matrices V_k for $k = 1, \dots, n$ starting from the initial condition $V_0 = \widehat{M}$ and using the iteration given by the Ricatti equation (2.9):

$$V_k = M + A^\top V_{k-1} A - A^\top V_{k-1} B \big(N + B^\top V_{k-1} B\big)^{-1} B^\top V_{k-1} A, \qquad k = 1, \dots, n.$$

(ii) Then, using the value V_{n-1}, one can compute the gains matrix F_{n-1} through the formula:

$$F_{n-1} = \big(N + B^\top V_{n-1} B\big)^{-1} B^\top V_{n-1} A,$$

and use this matrix together with the initial condition $x_0^\star = x_0$ to compute the first optimal control $u_0^\star = -F_{n-1} x_0^\star$, and therefore the next point $x_1^\star = A x_0^\star + B u_0^\star$.

(iii) We can repeat the previous point for any k from 1 to $n-1$: using the value V_{n-1-k}, one can compute the gains matrix F_{n-1-k} through the formula:

$$F_{n-1-k} = \big(N + B^\top V_{n-1-k} B\big)^{-1} B^\top V_{n-1-k} A,$$

and then use this matrix together with the current state $x_k^\star$ to compute the optimal control at step k, *i.e.* $u_k^\star = -F_{n-1-k} x_k^\star$, as well as the next point $x_{k+1}^\star = A x_k^\star + B u_k^\star$.

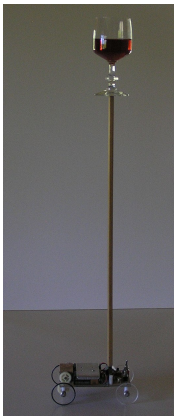
Note that after the first step, we can compute the optimal cost when starting at an arbitrary point $x \in \mathbb{R}^d$. It is indeed given by $v_n(x) = x^\top V_n x$. However, we do not yet know what the optimal control is to achieve this minimum cost. This is precisely the goal of steps (ii) and (iii).

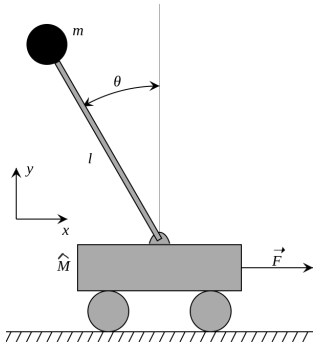
2.3 Controllability

The *controllability* of a dynamics system is its ability to move from any initial state to any final state with a judicious choice of a control process. It is central in many engineering applications. We will first discuss the classical problem of an inverted pendulum to motivate the problem and then give its general definition and resolution.

2.3.1 The inverted pendulum

This is a famous problem in optimal control that has served as a benchmark problem in robotics. The goal is to keep an inverted pendulum in the horizontal position by moving the cart on which it is located. Clearly this position is unstable and any disturbance would kick it to its stable downward position without any interference from the controller. The control here is the acceleration of the cart on which the pendulum is placed. Through some relevant approximations, the problem can be modeled as a Linear Quadratic problem, and its solution is provided by the Linear Quadratic Regulator. The critical question is whether the relevant linear dynamical system can be stabilized by an appropriate choice of the controls or equivalently, whether this system is controllable.





We continue by quickly deriving the linear dynamical system. We first describe the constant specifications of the system: L is the length of the pendulum, m is the mass of the object at the top of the pendulum, and $\widehat{M}$ is the mass of the cart. We now assume that the cart is moving on a one-dimensional *frictionless* track. Therefore, at some time $t \geqslant 0$, we denote by $p(t)$ the location of the cart on the track, by $\theta(t)$ be the angle between pendulum and the vertical axis, and finally by $F(t)$ the action applied to move the cart. The two equations below follow from classical mechanics and we do not report the details:

$$F(t) = (\widehat{M}+m)p''(t) - mL\theta''(t)\cos(\theta(t)) + mL\left(\theta'(t)\right)^2\sin(\theta(t)),$$
$$L\theta''(t) = g\sin(\theta(t)) + p''(t)\cos(\theta(t)),$$

where g is the gravitational constant.

To model this as a Linear Quadratic problem, we first linearize around the target position $\theta = 0$. Hence, we can make the following approximations, $\cos(\theta) \approx 1-\theta^2 \approx 1$ and $\sin(\theta) \approx \theta$. This yields to the following system of linear differential equations,

$$F(t) = (\widehat{M}+m)p''(t) - mL\theta''(t), \qquad L\theta''(t) = g\theta(t) + p''(t).$$

Therefore, the state (p,θ) of the system satisfy the following second-order equations:

$$p''(t) = \frac{mg}{\widehat{M}}\theta(t) + \frac{F(t)}{\widehat{M}}, \qquad \theta''(t) = \frac{g(m+\widehat{M})}{L\widehat{M}}\theta(t) + \frac{F(t)}{L\widehat{M}}.$$

where F is the control. The next step is to rewrite the previous equations as a system of first-order equations. We let

$$x(t) := (p(t), p'(t), \theta(t), \theta'(t))^\top, \quad \text{and} \quad u(t) := \frac{F(t)}{L\widehat{M}}.$$

As usual, x denotes the controlled state and $\mathbf{u}$ the control process. Hence, we can write a linear dynamics for the state x, *i.e.* $x'(t) = A_c x(t) + B_c u(t)$, where

$$A_c = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & b & 0 \end{pmatrix} \qquad B_c = \begin{pmatrix} 0 \\ L \\ 0 \\ 1 \end{pmatrix}, \quad a = \frac{mg}{\widehat{M}}, \quad b = \frac{(m+\widehat{M})g}{L\widehat{M}}.$$

Finally, we discretize time with $\Delta t =: h$ very small, and use the following finite-difference approximation:

$$x'(kh) \approx \frac{1}{h}(x((k+1)h) - x(kh)), \quad k = 0,1,\dots$$

Therefore, using the notations $x_k := x(kh)$ and $u_k := u(kh)$ for $k = 0,1,\dots$, we have:

$$x_{k+1} \approx x_k + hx'_k = x_k + hA_c x_k + hB_c u_k = (I_4 + hA_c)x_k + hB_c u_k.$$

We then obtain that the evolution of the state x_k is described by the equation (2.1) with

$$A = I_4 + hA_c = \begin{pmatrix} 1 & h & 0 & 0 \\ 0 & 1 & ah & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & bh & 1 \end{pmatrix}, \qquad B = hB_c = \begin{pmatrix} 0 \\ Lh \\ 0 \\ h \end{pmatrix}.$$

The goal of the next subsection is to show that this system is controllable, and that the corresponding Riccati equation has a unique solution.

2.3.2 Definition and the controllability matrix

As in the problem of inverted pendulum, the question of controllability of the system (2.1) is central in many engineering applications. A dynamical system is called *controllable* if it can move from any initial state to any final state with an appropriate control process. Mathematically, the definition is as follows.

Definition 2.3.1 — Controllability. The controlled dynamical system (2.1) is said to be controllable if for any pair $(x_{\text{in}}, x_{\text{out}}) \in \mathbb{R}^d \times \mathbb{R}^d$, there exists a control process $\mathbf{u} \in \mathscr{U}$ so that the solution of (2.1) with initial data $x_0 = x_{\text{in}}$ moves to $x_n = x_{\text{out}}$ at some time point $n \in \mathbb{N}$.

Note that for $d = \ell = 1$ with $B \neq 0$, the system is always controllable. Similarly, if $d = \ell$ and B is invertible, then for any pair $(x_{\text{in}}, x_{\text{out}}) \in \mathbb{R}^d \times \mathbb{R}^d$, one can choose the control

$$u_0 = B^{-1}\left(x_{\text{out}} - Ax_{\text{in}}\right)$$

so that

$$x_1 = Ax_0 + Bu_0 = Ax_{\text{in}} + BB^{-1}\left(x_{\text{out}} - Ax_{\text{in}}\right) = x_{\text{out}},$$

which shows controllability. However, the question is a subtle one in higher dimensions and it depends on the interaction between the matrices A and B.

Theorem 2.3.1 — Controllability. The dynamic system (2.1) is controllable if and only if the $d \times \ell d$ controllability matrix

$$\mathscr{C} := \begin{pmatrix} B & AB & \cdots & A^{d-1}B \end{pmatrix}$$

has (full) rank d. Moreover, when the system is controllable, its corresponding Riccati equation (2.7) has a solution.

Proof of Theorem 2.3.1. For any $\mathbf{u} \in \mathscr{U}$, recall that by mathematical induction we have:

$$\begin{aligned} x_d &= Ax_{d-1} + Bu_{d-1} = A\left(Ax_{d-2} + Bu_{d-2}\right) + Bu_{d-1} \\ &= A^2 x_{d-2} + ABu_{d-2} + Bu_{d-1} \\ &= \dots \\ &= A^d x_0 + \sum_{k=0}^{d-1} A^k B u_{d-k-1}. \end{aligned}$$

Using the controllability matrix defined in Chapter 2.3.1, one can write:

$$x_d = A^d x_0 + \mathscr{C}\mathbf{u}^{(d)}, \qquad \text{where} \qquad \mathbf{u}^{(d)} := \begin{pmatrix} u_{d-1} \\ \cdots \\ u_0 \end{pmatrix}.$$

Hence, the linear map

$$\mathscr{C} : \mathbf{u}^{(d)} \in \mathbb{R}^{\ell d} \mapsto x_d - A^d x_0 \in \mathbb{R}^d$$

is surjective (or *onto*) if and only if $\mathscr{C}$ has full (row) rank. Note that this is equivalent to controllability, as by a judicious choice of $\mathbf{u}^{(d)}$, the vector $\mathscr{C}\mathbf{u}^{(d)}$ could be equal to $x_{\text{out}} - A^d x_{\text{in}}$. The final statement about the Riccati equation is proved in Theorem 2.3.2, below. ■

■ **Example 2.4** Consider the example of the inverted pendulum. Then,

$$\mathscr{C} = \begin{bmatrix} 0 & h^2 L & 2h^2 L & 3h^2 L + ah^3 \\ hl & hL & hL + ah^3 & hlL + 3ah^3 \\ 0 & h^2 & 2h^2 & 3h^2 + bh^4 \\ h & h & h + bh^3 & h + 3bh^3 \end{bmatrix},$$

and it has full full rank. In fact, the rank of $\mathscr{C}$ is equal to the rank of $\mathscr{C}_c$ where

$$\mathscr{C}_c = \begin{bmatrix} B_c & A_cB_c & \cdots A_c^{d-1}B_c \end{bmatrix} = \begin{bmatrix} 0 & L & 0 & a \\ L & 0 & a & 0 \\ 0 & 1 & 0 & b \\ 1 & 0 & b & 0 \end{bmatrix}.$$

As $a/(Lb) = m/(m+M)$, we see that the above matrix has full rank of 4. Therefore, the linearized inverted pendulum is controllable. ∎

2.3.3 Link with the Riccati equation

As already mentioned, in the infinite horizon, it is possible that the value function is infinite. In such a problem, the corresponding Riccati equation does not have a solution, and the system is not controllable. This result is stated in the following theorem.

Theorem 2.3.2 — Solution to Riccati. Suppose that the system characterised by (A,B) is controllable. Then, the value function is finite and the value matrix V is the unique solution of the Riccati equation (2.7).

Proof. The definition of controllability implies, in particular, that for any initial condition $x \in \mathbb{R}^d$, there exists an admissible control $\mathbf{u}$ such that the system can be moved to the origin in finite time, say n, *i.e.* $x_n = 0$. After time n, we set the control to zero, *i.e.* $u_k = 0$ for $k = n+1, \ldots$. Let $\bar{\mathbf{u}}$ be this control sequence. Since both the control and the state are equal to zero after time n,

$$\begin{aligned} v_\infty(x) &:= \inf_{\mathbf{u}\in\mathscr{U}} J_\infty(x,\mathbf{u}) = \inf_{\mathbf{u}\in\mathscr{U}} \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k\right) \\ &\leqslant \sum_{k=0}^{\infty} \rho^k \left(x_k^\top M x_k + \bar{u}_k^\top N \bar{u}_k\right) = \sum_{k=0}^{n} \rho^k \left(x_k^\top M x_k + \bar{u}_k^\top N \bar{u}_k\right) \\ &= J_\infty(\bar{\mathbf{u}}) < \infty. \end{aligned}$$

Hence, the value function is finite.

It remains to show that $v(x) = x^\top V x$ where V is a solution of the Riccati equation (2.7). With this in mind, we introduce the value function $v_n(x)$ of the corresponding system but with finite horizon n, *i.e.*

$$v_n(x) = \sum_{k=0}^{n-1} \rho^k \left(x_k^\top M x_k + u_k^\top N u_k\right) + x_n^\top M x_n,$$

with, $v_0(x) = x^\top M x$. By induction and using the inhomogeneous Riccati equation (2.9), we can show that the value function is quadratic. That is, there exits a positive semi-definite matrix V_n such that $v_n(x) = x^\top V_n x$. As $v_n(x)$ is non-decreasing in n, it does have a limit, and we naturally have,

$$\lim_{n\to\infty} v_n(x) = v(x).$$

Since each v_n is quadratic and since $v(x)$ is finite for every x, we conclude that there exists a positive semi-definite matrix V such that $v(x) = x^\top V x$. We then use the dynamic programming to conclude that V is a solution of the Riccati equation (2.7). ■

Theorem 2.3.1 states that if the system is controllable then the value function is finite. However, this is not an equivalence, as illustrated by the example below. More precisely, the fact that the value function is finite does not imply that the system is controllable, or similarly the fact that the system is not controllable does not necessarily imply that the value function is infinite.

■ **Example 2.5** Suppose that $d = 2$, $\ell = 1$, $\rho = 1$ and

$$A := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad x_0 := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Let $x_k = (y_k, z_k)^\top$. Then, the equation $x_{k+1} = Ax_k + Bu_k$ is equivalent to the following two decoupled equations,

$$y_{k+1} = y_k, \text{ and } z_{k+1} = z_k + u_k \qquad k = 0, 1, \ldots$$

We observe that the sequence y is not controlled and therefore constant, *i.e.* $y_k = y_0 = 1$ for every k. Naturally, the system is not controllable since there is no way of driving y to an arbitrary position $y_{\text{out}} \neq 1$. Indeed, the controllability matrix of the above system,

$$\mathscr{C} := \begin{pmatrix} B & AB \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

The matrix $\mathscr{C}$ isof rank one instead of two, implying that the system is not controllable.

1. Let us consider the value function for

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad N = 1.$$

Notice that, for any action sequence $\mathbf{u}$,

$$J_\infty(x_0, \mathbf{u}) = \sum_{k=0}^{\infty} \left(y_k^2 + z_k^2 + u_k^2\right) = \sum_{k=0}^{\infty} \left(1 + z_k^2 + u_k^2\right) = \infty.$$

Hence, in this case, the associated value function $v_\infty(x_0)$ in infinite.

2. Consider now the same problem but with

$$M = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Then, for any control sequence $\mathbf{u}$, the cost is given by

$$J_\infty(x_0, \mathbf{u}) = \sum_{k=0}^{\infty} \left(z_k^2 + u_k^2\right), \text{ with } z_{k+1} = z_k + u_k, \; z_0 = 1.$$

Remark that by taking $\bar{\mathbf{u}} := (-1, 0, 0, \ldots)$, we have

$$z_0 = 1, \; z_1 = z_0 + u_0 = 1 - 1 = 0, \; \ldots, \; z_{k+1} = z_k + u_k = 0, \; \ldots,$$

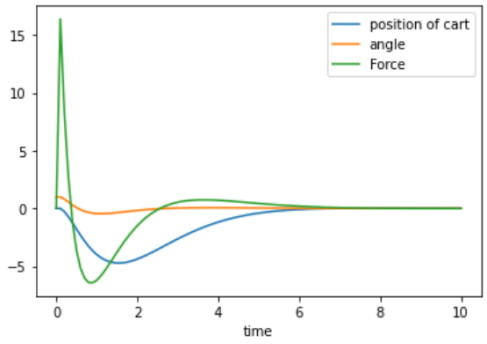
and therefore

$$v_\infty(x_0) := \inf_{\mathbf{u} \in \mathscr{U}} J_\infty(\mathbf{u}) \leqslant J_\infty(\bar{\mathbf{u}}) = 1^2 + (-1)^2 = 2 < \infty.$$

In this case, the value function is finite even if the system is not controllable. ■

■ **Example 2.6** We again consider the example of the inverted pendulum. With $\rho = 1$, $M = I_4$ and $N = 1$, the gains matrix F by using a python package, value matrix V and the gains matrix F are given by,

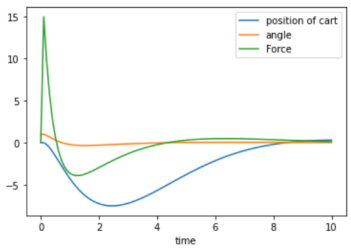
```
V=
[[  25.7561   26.881  -131.0256  -41.3015]
 [  26.881    53.4442 -283.0665  -89.1439]
 [-131.0256 -283.0665 2347.5913  736.0058]
 [ -41.3015  -89.1439  736.0058  232.7963]]

F= [[-0.693 -1.786 28.741  9.086]]

A-BF=
[[ 1.       0.1      0.       0.     ]
 [ 0.06935  1.17861 -2.86407 -0.9086 ]
 [ 0.       0.       1.       0.1    ]
 [ 0.06935  0.17861 -1.86407  0.0914 ]]

eigenvalues of A-BF are

 1 : (0.67+0j)
 2 : (0.77+0j)
 3 : (0.92+0.05j)
 4 : (0.92-0.05j)

eigenvectors of A-BF are

 1 : [0.03+0.j 0.27+0.j 0.72+0.j 0.72-0.j]
 2 : [-0.1 +0.j   -0.62+0.j   -0.6 +0.33j -0.6 -0.33j]
 3 : [ 0.28+0.j   -0.3 +0.j   -0.03+0.06j -0.03-0.06j]
 4 : [-0.95+0.j    0.68+0.j   -0.  -0.07j -0.  +0.07j]
```



Note that the resulting controlled dynamical system $x_{k+1} = Ax_k + Bu_k^* = (A - BF)x_k$, is stable. Equivalently, as calculated above the absolute value of the real parts of the eigenvalues of the matrix $A - BF$ are strictly less one. This implies that the solutions converge to zero exponentially fast, as seen in the numerical solutions.

We change M slightly we get a similar qualitative behavior.

```
V=
[[   3.3624    5.4846  -31.0595   -9.8214]
 [   5.4846   14.7868  -91.5057  -28.935 ]
 [ -31.0595  -91.5057 1322.6335  416.2149]
 [  -9.8214  -28.935   416.2149  131.1637]]

F= [[-0.231 -0.775 22.747  7.162]]

M=
[[0.1 0.  0.  0. ]
 [0.  0.  0.  0. ]
 [0.  0.  1.  0. ]
 [0.  0.  0.  0. ]]
```



■

R The Linear Quadratic problem can also be formulated in continuous-time. One may find this formulation on the Wikipedia page *Linear–quadratic regulator*.

2.4 Stochastic Dynamics

In the Linear Quadratic Gaussian problem, or Stochastic Linear Quadratic problem, the system dynamics is given by the addition of a Gaussian noise to the deterministic equation (2.1). In the auto-pilot example or in the inverted pendulum, this noise or randomness corresponds to disturbances due to wind or other inaccuracies.

More precisely, the random dynamics is given by,

$$x_k = Ax_{k-1} + Bu_{k-1} + \omega_k, \qquad k = 1,2,\dots, \tag{2.10}$$

where A, B are as before, and ω_k is a sequence of *independent*, Gaussian random process with *mean* 0 and a known $d \times d$ *covariance matrix* Q_k.

In this stochastic model, the control process u can also be random. Indeed, when choosing the optimal control at time k, the decision makers should use all the information available up to time k, and in particular the values $\omega_1, \dots, \omega_k$ of the random noise may influence his decision. Note that here, contrary to the deterministic model, it is crucially important that the decision makers use *only* the information available at time k to choose u_k. Namely the decision u_k made at time k is only a function of $\omega_1, \dots, \omega_k$ and other known and given quantities such as A, B, x_0. Such random control processes $\mathbf{u}$ are called *adapted* to the information obtained by observing the noise. As usual, we let $\mathscr{U}$ to be set of adapted control processes.

2.4.1 Infinite Horizon

We assume here that the covariance matrices Q_k are independent of k, and we thus denote it simply Q. In the infinite horizon case, for an initial condition $x_0 = x$ and a control process $\mathbf{u} \in \mathscr{U}$, the objective function is defined almost as in (2.2). However, note that all processes here are random, and we therefore need to average them in order to obtain a real-valued objective function (not a random one). With this in mind, the cost to minimize is defined by:

$$J_\infty(x, \mathbf{u}) := \sum_{k=0}^{\infty} \rho^k \mathbb{E}\left[x_k^\top M x_k + u_k^\top N u_k\right],$$

for a fixed discount factor $\rho \in (0,1)$. Then, the value function corresponding to this minimisation problem is naturally defined by:

$$v_\infty(x) := \inf_{\mathbf{u} \in \mathscr{U}} J_\infty(x, \mathbf{u}).$$

■ **Example 2.7** Let us consider again the simple one-dimensional Chapter 2.1 studied in Chapter 2.1.2. To make it stochastic, we just add a Gaussian noise to the system dynamics as follows:

$$x_{k+1} = 4x_k + u_k + \omega_{k+1}, \qquad k = 0, 1, \dots$$

More precisely, the noise process ω_k is an i.i.d. sequence of Gaussian random variables with mean zero and variance $\sigma^2 > 0$.

Suppose that we use the Linear Quadratic Regulator found before, *i.e.* the linear feedback control $u_k = -fx_k$ for some constant $f \in \mathbb{R}$ to be determined. We compute that

$$x_1 = (4-f)x_0 + \omega_1, \quad x_2 = (4-f)^2 x_0 + \omega_2 + (4-f)\omega_1, \dots$$

Hence, by mathematical induction, we obtain that for any $n \geqslant 1$,

$$x_n = (4-f)^n x_0 + y_n, \qquad \text{where} \qquad y_n = \sum_{k=1}^{n} (4-f)^{n-k} \omega_k.$$

Suppose that $|4-f|<1$. Then, $\mathbb{E}[x_n]=(4-f)^n x_0$ converges to zero as n tends to infinity. However,

$$\mathbb{V}\mathrm{ar}[x_n]=\mathbb{V}\mathrm{ar}\left[(4-f)^n x_0+y_n\right]=\mathbb{V}\mathrm{ar}[y_n]=\sum_{k=1}^{n}\mathbb{V}\mathrm{ar}\left[(4-f)^{n-k}\omega_k\right]=\sum_{k=1}^{n}(4-f)^{2(n-k)}\sigma^2,$$

since the ω_k are *i.i.d* with variance σ^2, and therefore

$$\lim_{n\to\infty}\mathbb{V}\mathrm{ar}[x_n]=\lim_{n\to\infty}\sigma^2\sum_{i=0}^{n-1}(4-f)^{2i}=\sigma^2\sum_{i=0}^{\infty}\left((4-f)^2\right)^i=\frac{\sigma^2}{1-(4-f)^2}>0.$$

We see in this simple example that, in the presence of a Gaussian noise, it is not always possible to derive the state process to zero. Nevertheless, the mean of the state converges to zero, and its variance is finite. ∎

We can proceed exactly as in the (deterministic) LQ case to derive the following dynamic programming equation:

$$v_\infty(x)=x^\top Mx+\inf_{u\in\mathbb{R}^\ell}\left\{u^\top Nu+\rho\mathbb{E}\left[v_\infty(Ax+Bu+\omega_1)\right]\right\},\qquad\forall\, x\in\mathbb{R}^d. \tag{2.11}$$

This equation is slightly different than the dynamic programming equation because of the expectation. Nevertheless, the optimal solution does not change, in the sense that the optimal control is given by the Linear Quadratic Regulator, *i.e.* the feedback control described in Chapter 2.2.3. This is precisely stated in the following theorem.

Theorem 2.4.1 Given the matrices (A,B,M,N) and a discount factor ρ, let V be the solution of the Riccati equation (2.7), and F be the corresponding gains matrix. Then, the value function of the LQG problem is given by

$$v_\infty(x)=x^\top Vx+\frac{\rho}{1-\rho}\mathrm{Tr}(VQ).$$

Moreover, the linear feedback control $u_k^\star=-Fx_k^\star$ is optimal.

Proof. Let V,F be as defined in the theorem. Set

$$\varphi(x):=x^\top Vx+\frac{\rho}{1-\rho}\mathrm{Tr}(VQ).$$

The goal is to show that φ solves the dynamic programming equation (2.11), and therefore coincides with the value function. Set

$$\mathscr{I}(x):=x^\top Mx+\inf_{u\in\mathbb{R}^\ell}\left\{u^\top Nu+\rho\mathbb{E}\left[\varphi(Ax+Bu+\omega_1)\right]\right\}.$$

We need to show that $\mathscr{I}=\varphi$. Using the definition of φ, we first compute the following expectation:

$$\begin{aligned}\mathbb{E}\left[\varphi(Ax+Bu+\omega_1)\right]&=\mathbb{E}\left[(Ax+Bu+\omega_1)^\top V(Ax+Bu+\omega_1)+\frac{\rho}{1-\rho}\mathrm{Tr}(VQ)\right]\\&=(Ax+Bu)^\top V(Ax+Bu)+\frac{\rho}{1-\rho}\mathrm{Tr}(VQ)+\mathbb{E}\left[\omega_1^\top V\omega_1\right]\\&=(Ax+Bu)^\top V(Ax+Bu)+\frac{\rho}{1-\rho}\mathrm{Tr}(VQ)+\mathrm{Tr}(VQ)\\&=(Ax+Bu)^\top V(Ax+Bu)+\frac{1}{1-\rho}\mathrm{Tr}(VQ).\end{aligned}$$

In the above we used that since ω_k has zero mean,

$$\mathbb{E}\big[\omega_1^\top V \omega_1\big] = \mathbb{E}\Big[\sum_{k=1}^{d}\sum_{i=1}^{d}\omega_1^k \omega_1^i V_{ik}\Big] = \sum_{k=1}^{d}\sum_{i=1}^{d} V_{ik}\mathbb{E}\big[\omega_1^k\omega_1^i\big] = \sum_{i=1}^{d}\sum_{k=1}^{d} V_{ik}Q_{ki} = \mathrm{Tr}(VQ).$$

We use the above calculation in the definition of $\mathscr{I}$ to arrive at

$$\mathscr{I}(x) = x^\top Mx + \inf_{u\in\mathbb{R}^\ell}\big\{u^\top Nu + \rho(Ax+Bu)^\top V(Ax+Bu)\big\} + \frac{\rho}{1-\rho}\mathrm{Tr}(VQ).$$

Notice that, since V satisfies the Riccati equation (2.7), we can apply the results of the deterministic case. Namely, Theorem 2.2.3 states that:

$$x^\top Vx = x^\top Mx + \inf_{u\in\mathbb{R}^\ell}\big\{u^\top Nu + \rho(Ax+Bu)^\top V(Ax+Bu)\big\}.$$

Using this identity conclude that

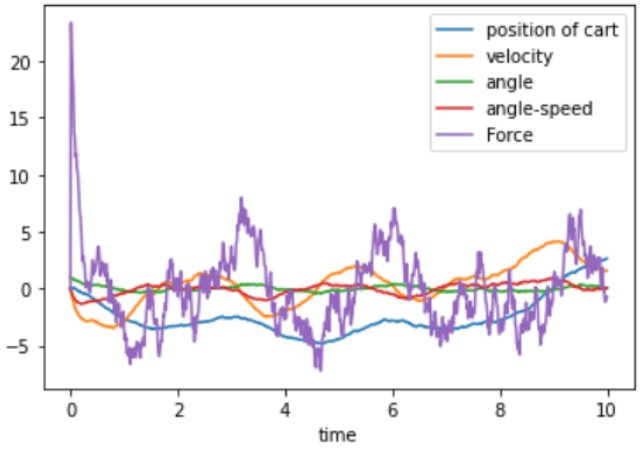
$$\mathscr{I}(x) = x^\top Vx + \frac{\rho}{1-\rho}\mathrm{Tr}(VQ) = \varphi(x).$$

Hence, φ solve the dynamic programming equation (2.11) and is equal to the value function. Moreover, by previous computations and Theorem 2.2.3,

$$\begin{aligned}
&\arg\min_{u\in\mathbb{R}^\ell}\big\{u^\top Nu + \rho\mathbb{E}\big[v_\infty(Ax+Bu+\omega_1)\big]\big\}\\
&\quad= \arg\min_{u\in\mathbb{R}^\ell}\Big\{u^\top Nu + \rho(Ax+Bu)^\top V(Ax+Bu) + \frac{\rho}{1-\rho}\mathrm{Tr}(VQ)\Big\}\\
&\quad= \arg\min_{u\in\mathbb{R}^\ell}\big\{u^\top Nu + \rho(Ax+Bu)^\top V(Ax+Bu)\big\} = -Fx,
\end{aligned}$$

where F is as in the deterministic case. Therefore, $u^\star(x) = -Fx$ the optimal feedback control. ■

Below is a simulation with A and B from the inverted pendulum problem with $h = 0.01$, $N = 1$, $\rho = 1$, and $M = \mathrm{diag}(0.1, 0, 1, 1)$ and covariance matrix $Q = 0.25\,hI$. The python code is provided in the course webpage. The states processes, although not converging to zero, oscillate around the origin preventing the pendulum from tipping over.



2.4.2 Finite Horizon

We now consider the Linear Quadratic Gaussian problem with an arbitrary but finite horizon $n \in \mathbb{N}$. In this case, the objective function is naturally defined for all starting point $x \in \mathbb{R}$ and all admissible control $\mathbf{u} \in \mathscr{U}$ by

$$J_n(x,\mathbf{u}) := \sum_{k=0}^{n-1} \mathbb{E}\big[x_k^\top M x_k + u_k^\top N u_k\big] + \mathbb{E}\big[x_n^\top \widehat{M} x_n\big],$$

and the value function is defined by,

$$v_n(x) := \inf_{\mathbf{u}\in\mathscr{U}} J_n(x,\mathbf{u}).$$

Using the standard approach, we can prove the following dynamic programming equation:

$$v_{n+1}(x) = x^\top M x + \inf_{u\in\mathbb{R}^\ell} \Big\{u^\top N u + \mathbb{E}\big[v_n(Ax+Bu+\omega_1)\big]\Big\}, \qquad \forall\, x\in\mathbb{R}^d,\ n\in\mathbb{N}.$$

The expected value simply averages over the ω_1:

$$\mathbb{E}[v(Ax+Bu+\omega_1)] = \int_{\mathbb{R}^d} v(Ax+Bu+z)\, f_z(z)\, \mathrm{d}z,$$

where f_z is the Gaussian probability density of $\omega_k 1 \in \mathbb{R}^d$,

$$f_z(z) = \frac{1}{\sqrt{(2\pi)^d \det(Q_1)}} \exp(-\frac{1}{2} Q_1^{-1} z\cdot z), \qquad z\in\mathbb{R}^d.$$

Let V_k with $V_0 = \widehat{M}$ be the solution of the Riccati equation (2.9). In this stochastic case, it is clear that we still have $v_0(x) = x^\top \widehat{M} x = x^\top V_0 x$. However, the simple quadratic assumption is no longer true for other n values, as we have seen already in the infinite horizon case. Indeed, by dynamic programming,

$$\begin{aligned} v_1(x) &= x^\top M x + \inf_{u\in\mathbb{R}^\ell} \Big\{u^\top N u + \mathbb{E}\big[v_0(Ax+Bu+\omega_1)\big]\Big\} \\ &= x^\top M x + \inf_{u\in\mathbb{R}^\ell} \Big\{u^\top N u + \mathbb{E}\big[(Ax+Bu+\omega_1)^\top \widehat{M}(Ax+Bu+\omega_1)\big]\Big\} \\ &= x^\top M x + \inf_{u\in\mathbb{R}^\ell} \Big\{u^\top N u + (Ax+Bu)^\top \widehat{M}(Ax+Bu)\Big\} + \mathbb{E}\big[\omega_1^\top \widehat{M}\omega_1\big]. \end{aligned}$$

We observe that the equation is the same as in the deterministic case, except for the last additional term. As in the previous infinite horizon case, observe that

$$\mathbb{E}\big[\omega_1^\top \widehat{M}\omega_1\big] = \mathrm{Tr}(\widehat{M}Q_1) = \mathrm{Tr}(V_0 Q_1),$$

which implies

$$v_1(x) = x^\top M x + \inf_{u\in\mathbb{R}^\ell} \Big\{u^\top N u + (Ax+Bu)^\top V_0 (Ax+Bu)\Big\} + \mathrm{Tr}(V_0 Q_1).$$

As the sequence V solves the Riccati equation (2.9), we have in particular by Chapter 2.2.6 that

$$x^\top V_1 x = x^\top M x + \inf_{u\in\mathbb{R}^\ell} \big\{u^\top N u + (Ax+Bu)^\top V_0 (Ax+Bu)\big\}, \qquad \forall x\in\mathbb{R}^d.$$

Consequently,

$$v_1(x) = x^\top V_1 x + \mathrm{Tr}(V_0 Q_1).$$

By a direct induction argument, we can show that this holds for all n. Indeed, for all n, we have:

$$v_{n+1}(x) = x^\top M x + \inf_{u\in\mathbb{R}^\ell} \left\{ u^\top N u + (Ax+Bu)^\top V_n (Ax+Bu) \right\} + \mathrm{Tr}(V_n Q_n),$$

and using that V_n solves the Riccati equation (2.9), we have by Chapter 2.2.6 that

$$x^\top V_n x = x^\top M x + \inf_{u\in\mathbb{R}^\ell} \left\{ u^\top N u + (Ax+Bu)^\top V_{n-1} (Ax+Bu) \right\}, \qquad \forall x \in \mathbb{R}^d.$$

Replacing in the previous equation, we obtain the general formula:

$$v_n(x) = x^\top V_n x + \mathrm{Tr}(V_{n-1} Q_{n-1}).$$

This is summarised in the following theorem, which states in particular that there is no essential difference between the deterministic problem and its stochastic equivalent, especially in terms of the optimal control, still given by the Linear Quadratic Regulator. The only difference lies in the formula of the value function.

Theorem 2.4.2 Let the sequence $V := (V_0, \dots, V_n)$ be the solution of the inhomogeneous Riccati equation (2.9). Then, the value function of the Linear Quadratic Gaussian problem is given by

$$v_n(x) = x^\top V_n x + \mathrm{Tr}(V_{n-1} Q_{n-1}), \qquad \forall x \in \mathbb{R}^d,\ n \in \mathbb{N}.$$

Moreover, the linear feedback control $u_k^\star = -F_{n-k-1} x_k^\star$ constructed in Chapter 2.2.3 is also optimal for the LQG problem.

2.5 Exercises

Problem 2.1 Consider the controlled system

$$y_{k+2} + y_k = u_k, \qquad k = 0, 1, n,$$

with $y_0 = y_1 = 1$ and the cost functional,

$$J = \sum_{k=0}^{n-1} [y_{k+1}^2 + y_k^2 + u_k^2] + [y_{n+1}^2 + y_n^2].$$

where u_k is the control.

a. Express the above system in the standard form.

b. Write down the Riccati equations.

c. Solve for V_0, V_1, V_2.

d. Compute the optimal controls u_0^*, u_1^* with horizon $n = 2$.

Problem 2.2 Consider the LQR with

$$A = \begin{bmatrix} 0.5 & 0 \\ 0 & a \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad N = 1,$$

with $0 < \rho \leqslant 1$ and the constant a are arbitrary.

a. Is (A, B) controllable?

b. For which values of $x_0 = x$, ρ and a, is the value function $v(x)$ finite?

c. For $a = \rho = 1$, write down the Riccati equation with infinite horizon.
d. Solve the above Riccati equation for $a = \rho = 1$.

Problem 2.3 Consider the LQR with

$$A = \begin{bmatrix} 1 & 0 \\ 0 & a \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad N = 1,$$

with $0 < \rho \leqslant 1$ and the constant a are arbitrary.
a. For which values of a, the system (A,B) controllable?
b. For which values of x, ρ and a, is the value function $v(x)$ finite?

Problem 2.4 Consider the control problem (which is not a LQR):

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

with a cost function for $\mathbf{u} = (u_0, u_1, \ldots)$,

$$J(x, \mathbf{u}) = \sum_{k=0}^{\infty} 2^{-k} [x_k^2 + u_k^2].$$

Let $v(x) = \min_{\mathbf{u}} J(x, \mathbf{u})$ be the value function.
a. Write down the dynamic programming equation for v (do not solve it).
b. Show that $v(0) < \infty$.
c. Show that $v(x) < \infty$ for every x

Problem 2.5 Consider an infinite horizon LQR, i.e., with $x = x_0$,

$$x_{k+1} = A x_k + B u_k, \qquad v(x) = \min_{\mathbf{u}} J(x, \mathbf{u}) := \sum_{k=0}^{\infty} \rho^k [M x_k \cdot x_k + N u_k \cdot u_k]$$

where $\mathbf{u} = (u_0, u_1, \ldots)$. Suppose that

$$A = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad N = 1,$$

where the constant a, the discount factor $\rho > 0$ and the matrix M are arbitrary.
a. (5pts) For which values of a, is the system (A,B) controllable?
b. (8pts) Suppose that

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

For which values of x, ρ and a, is the value function $v(x)$ finite?
c. (7pts) Suppose $a = 0$, $\rho = 1$, and

$$M = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \text{(note this is different than the one above).}$$

Compute the value function $v(x) = \min_{\mathbf{u}} J(\mathbf{u})$.

Problem 2.6 Consider the control problem (which is not a LQR):

$$x_{k+1} = x_k + u_k [z_k + 1],$$

where $z_0, z_1, \ldots$ is an i.i.d. sequence of random variables taking values ± 1 with equal probability. For a finite horizon n and control $\mathbf{u} = (u_0, u_1, \ldots)$,

$$J(n, x, \mathbf{u}) := \mathbb{E}\left[x_n - \frac{1}{2}\sum_{k=0}^{n-1} u_k^2\right].$$

Let $v(n,x) = \max_{\mathbf{u}} J(n,x,\mathbf{u})$ be the value function.
a. Write down the dynamic programming equation for v.
b. Show that $v(n,x) = x + \frac{n}{2}$.
c. Show that the optimal control is $u_k^* = 1$ for every k.

Problem 2.7 Consider the continuous time system

$$y''(t) = \sin(y(t)) + (1 + y(t))\, u(t),$$

where $u(t)$ is the control.
a. Rewrite the above equations as a system of first order equations.
b. Linearize the resulting system around $y(t) \equiv 0$.
c. Discretize (in time) the resulting linear equation.
d. Is the resulting system controllable?

Problem 2.8 Consider the continuous time system

$$y'''(t) = (y'(t))^3 + u(t),$$

where $u(t)$ is the control.
a. Rewrite the above equations as a system of first order equations.
b. Linearize the resulting system around $y(t) \equiv 0$.
c. Discretize (in time) the resulting linear equation.

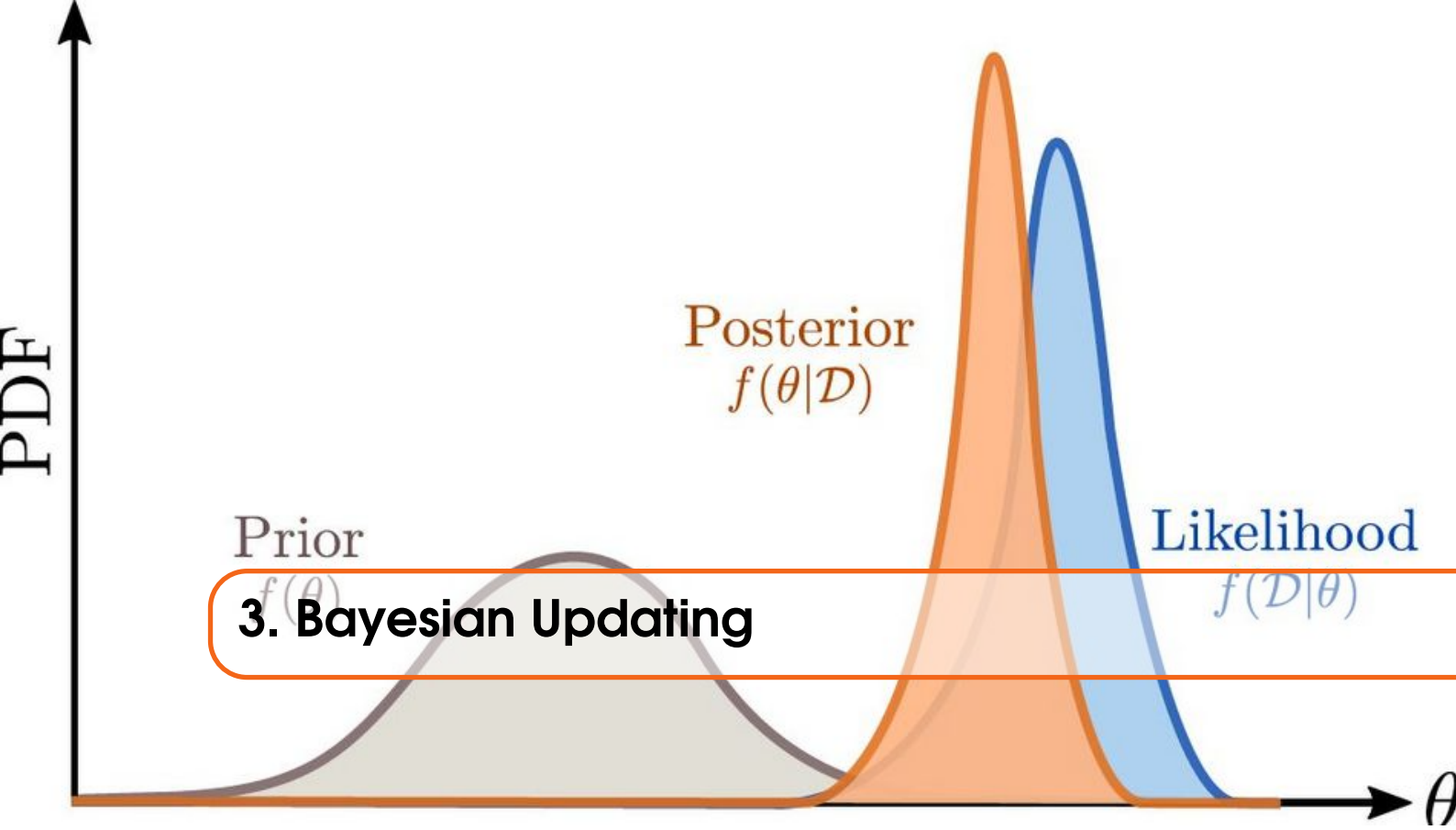
Problem 2.9 Consider the inverted pendulum problem of Section 2.5.1 in infinite horizon. Suppose that

$$h = 0.01, \quad a = 0.1, \quad b = 5.1, \quad L = 2, \quad \rho = 0.81.$$

a. Compute the A and B matrices.
b. Suppose that $N = 1$ and

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Write down the Riccati equation.
c. With the use of a python package compute the solution V of the Riccati equation and use it to compute the gains matrix F.
d. Show that the dynamics given by (A, B) is controllable.



3. Bayesian Updating

The famous Bayes formula can be seen as the first simple example of unsupervised learning. Indeed, in this structure, we have a certain *prior* distribution of a certain probability or a random variable. Then, based on an observation, we update and obtain the *posterior* distribution. Through this process we have a better estimate and may consider this as Bayesian learning. In this chapter, we recall the basics of the Bayes formula which was also covered in ORF 309 and discuss several examples to emphasize the learning.

Notation.

In what follows, we let Ω be a probability space and $\mathbb{P}$ be a probability measure on Ω. For given random variables $X \in \mathbb{R}^k, Y \in \mathbb{R}^d$, $\mathbb{C}(X,Y)$ is the $k \times d$ covariance matrix between X, Y:

$$\mathbb{C}(X,Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^\top].$$

We write $\mathbb{C}(X) := \mathbb{C}(X,X)$ for the $k \times k$ covariance matrix of X, $\mathbb{C}(Y) := \mathbb{C}(Y,Y)$ is the $d \times d$ covariance matrix of Y, and $\mathbb{C}((X,Y),(X,Y))$ is the $(k+d) \times (k+d)$ covariance matrix of the extended random variable (X,Y). Then,

$$\begin{aligned}
\mathbb{C}((X,Y),(X,Y)) &= \begin{bmatrix} \mathbb{C}(X) & \mathbb{C}(X,Y) \\ \mathbb{C}(Y,X) & \mathbb{C}(Y) \end{bmatrix}, \\
\mathbb{C}(X)_{i,n} &= \mathbb{E}\left[(X - \mathbb{E}[X])_i (X - \mathbb{E}[X])_n\right], \qquad i,n = 1,\dots,k, \\
\mathbb{C}(Y)_{l,j} &= \mathbb{E}\left[(Y - \mathbb{E}[Y])_l (Y - \mathbb{E}[Y])_j\right], \qquad l,j = 1,\dots,d, \\
\mathbb{C}(X,Y)_{i,j} &= \mathbb{E}\left[(X - \mathbb{E}[X])_i (Y - \mathbb{E}[Y])_j\right], \qquad i = 1,\dots,k, \;\; j = 1,\dots,d,
\end{aligned}$$

and $\mathbb{C}(X,Y) = \mathbb{C}(Y,X)^\top$. We write $\mathbb{C}(X|Y)$ for the conditional covariance of X given Y:

$$\mathbb{C}(X|Y) := \mathbb{E}[(X - \mathbb{E}[X|Y])^\top (X - \mathbb{E}[X|Y]) \,|\, Y].$$

When $k = 1$, we write $\mathrm{var}(X)$ for the scalar $\mathbb{C}(X)$.

3.1 Review of the Bayes formula

In the following subsubsections, we study the Bayes formula in three different settings, despite the fact that they all are highly similar to each other. We also derive several formulae in the Gaussian setting which will be extensively used in the derivation of the Kalman filter.

3.1.1 Bayes formula for probability events

An *event* A is a subset of the probability space Ω, and its conditional probability given another event B, also a subset of Ω, is defined by:

$$\mathbb{P}(A|B) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)},$$

assuming that $\mathbb{P}(B) \neq 0$. Similarly, when $\mathbb{P}(A) \neq 0$, we have:

$$\mathbb{P}(B|A) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}.$$

This definition implies in particular:

$$\mathbb{P}(A \cap B) = \mathbb{P}(A|B)\mathbb{P}(B) = \mathbb{P}(B|A)\mathbb{P}(A).$$

First Bayes formula. Let A, B be two subsets of the probability space Ω with $\mathbb{P}(B) \neq 0$ and $\mathbb{P}(A) \neq 0$, then

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)} \quad \text{and similarly} \quad \mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)}.$$

The following also is a direct consequence of the definitions:

$$\mathbb{P}(B) = \mathbb{P}(B \cap A) + \mathbb{P}(B \cap A^c) = \mathbb{P}(B|A)\mathbb{P}(A) + \mathbb{P}(B|A^c)\mathbb{P}(A^c),$$

where A^c is the complement of A, and thus $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$. Substituting this in the Bayes formula yields:

Second Bayes formula.

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B|A)\mathbb{P}(A) + \mathbb{P}(B|A^c)(1 - \mathbb{P}(A))}.$$

Hence, if we know the probability $\mathbb{P}(A)$ and the conditional probabilities $\mathbb{P}(B|A)$ and $\mathbb{P}(B|A^c)$, we can then compute the conditional probability of A given B.

We can generalise the previous formula to a given partition of Ω. More precisely, let $\{A_i\}_{i=1,\dots,m}$ be a *partition* of Ω, *i.e.* such that

$$\bigcup_{j=1}^{m} A_j = \Omega, \qquad \text{with} \qquad A_i \cap A_j = \varnothing, \quad \forall\, i \neq j.$$

Then, as $B = \bigcup_{j=1}^{m} (B \cap A_j)$, using the previous definition of conditional probability, we have:

$$\mathbb{P}(B) = \sum_{j=1}^{m} \mathbb{P}(B \cap A_j) = \sum_{j=1}^{m} \mathbb{P}(B|A_j)\,\mathbb{P}(A_j).$$

Consequently, substituting this in to the first Bayes formula yields:

Bayes formula with partition. For $i = 1, \dots, m$,

$$\mathbb{P}(A_i|B) := \frac{\mathbb{P}(A_i \cap B)}{\mathbb{P}(B)} = \frac{\mathbb{P}(B|A_i)\mathbb{P}(A_i)}{\mathbb{P}(B)} = \frac{\mathbb{P}(B|A_i)\mathbb{P}(A_i)}{\sum_{j=1}^{m} \mathbb{P}(B|A_j)\,\mathbb{P}(A_j)}. \tag{3.1}$$

The idea for the use of the Bayes formula in a learning framework is the following: starting from a prior "distribution" $\mathbb{P}(A_i)$, one can update it by observing the event B, *i.e.* construct a posterior "distribution" $\mathbb{P}(A_i|B)$ using the previous Bayes' formula. However, note that in order to do this updating, one also need to know the conditional probabilities $\mathbb{P}(B|A_i)$, in addition to the prior distribution $\mathbb{P}(A_i)$.

■ **Example 3.1 — Medical testing.**

Medical testing is one of the classical example for the use of the Bayes formula. Let p be the probability of a randomly chosen person to be infected by a disease/virus, *e.g.* Covid. This will correspond to the prior distribution. Then, we can test the person in order to determine if the he/she is infected or not. However, the test is not necessarily completely accurate: there may be false positives or false negatives.

Let p_+ be the probability that the Covid test for an infected person is indeed positive, and p_f be the probability of a false positive, *i.e.* the test is positive while in fact, the person is not infected.

Suppose a person is randomly chosen and tested for Covid. Let

$$A = \text{the person is infected}, \qquad B = \text{the test is positive}.$$

We know that

$$\mathbb{P}(A) = p, \qquad \mathbb{P}(B|A) = p_+, \qquad \mathbb{P}(B|A^c) = p_f.$$

These imply that

$$\mathbb{P}(B^c|A) = 1 - \mathbb{P}(B|A) = (1 - p_+), \qquad \mathbb{P}(B^c|A^c) = 1 - \mathbb{P}(B^c|A^c) = (1 - p_f).$$

We now use the Bayes' formula with the partition $\{A, A^c\}$ to compute the probabilities that this person is infected given that the test is positive and negative. The results are

$$\mathbb{P}(A|B) = \frac{p_+ p}{p_+ p + p_f (1-p)}, \qquad \mathbb{P}(A|B^c) = \frac{(1-p_+)\, p}{(1-p_+)\, p + (1-p_f)\,(1-p)}.$$

We tabulate some examples in the table below:

$p = \mathbb{P}(A)$	$p_+ = \mathbb{P}(B\|A)$	$p_f = \mathbb{P}(B\|A^c)$	$\mathbb{P}(A\|B)$	$\mathbb{P}(A\|B^c)$
3%	99%	10%	23%	0.034%
3%	99%	1%	75 %	0.031%
3%	99%	50%	5.7%	0.11%
3%	100%	50%	5.8%	0%
10%	99%	10%	52%	0.12%
10%	90%	1%	73%	0.31%

It is clear that one wants a high p_+ and a low p_f value. However, it may be very costly to achieve both. So one can find the most effective p_+, p_f values by studying the table above and the costs associated with them.

■

■ **Example 3.2 — Repeated testing.**

Suppose that after the first positive test, the patient is tested again and the result is again positive. We want to compute the probability that this person is infected given the two positive tests. We assume that $p = P(A) = 3\%$, $p_+ = \mathbb{P}(B_i\,|\,A) = 90\%$ and $p_f = \mathbb{P}(B_i\,|\,A^c) = 10\%$ for each $i = 1,2$. For each $i = 1,2$, we have

$$\mathbb{P}(B_i) = \mathbb{P}(B_i \cap A) + \mathbb{P}(B_i \cap A^c) = P(B_i\,|\,A)\mathbb{P}(A) + P(B_i\,|\,A^c)\mathbb{P}(A^c) = 0.1267.$$

We slightly change the notation of the previous example and set

$$B_1 = \text{first test is positive}, \qquad B_2 = \text{second test is positive}.$$

We define the *posterior* probability conditioned on the event B_1 by,

$$\mathbb{Q}(E) := \mathbb{P}(E\,|\,B_1),$$

where E is any event. By previous example, $p_1 := \mathbb{Q}(A) = \mathbb{P}(A\,|\,B_1) = 0.234412$. Our goal is to compute $p_2 := \mathbb{P}(A\,|\,B_1 \cap B_2)$. We observe that

$$\begin{aligned}
\mathbb{P}(A \cap B_1 \cap B_2) &= \mathbb{P}(A \cap B_2\,|\,B_1)\ \mathbb{P}(B_1) = \mathbb{Q}(A \cap B_2\,|\,B_1)\ \mathbb{P}(B_1),\\
\mathbb{P}(B_1 \cap B_2) &= \mathbb{P}(B_2\,|\,B_1)\ \mathbb{P}(B_1) = \mathbb{Q}(B_2)\ \mathbb{P}(B_1),
\end{aligned}$$

and therefore,

$$p_2 := \mathbb{P}(A\,|\,B_1 \cap B_2) = \frac{\mathbb{P}(A \cap B_1 \cap B_2)}{\mathbb{P}(B_1 \cap B_2)} = \frac{\mathbb{P}(A \cap B_2\,|\,B_1)\ \mathbb{P}(B_1)}{\mathbb{P}(B_1)\,\mathbb{Q}(B_2)} = \frac{\mathbb{Q}(A \cap B_2)}{\mathbb{Q}(B_2)} = \mathbb{Q}(A\,|\,B_2).$$

We further develop this formula by using the Bayes formula for the probability $\mathbb{Q}$:

$$p_2 = \mathbb{Q}(A\,|\,B_2) = \frac{\mathbb{Q}(B_2\,|\,A)\ \mathbb{Q}(A)}{\mathbb{Q}(B_2\,|\,A)\ \mathbb{Q}(A) + \mathbb{Q}(B_2\,|\,A^c)\ \mathbb{Q}(A^c)}.$$

To compute p_2 we need to know $\mathbb{Q}(B_2\,|\,A)$ and $\mathbb{Q}(B_2\,|\,A^c)$. By the definition of $\mathbb{Q}$:

$$\mathbb{Q}(B_2\,|\,A) = \mathbb{P}(B_2\,|\,A \cap B_1), \qquad \text{and} \qquad \mathbb{Q}(B_2\,|\,A^c) = \mathbb{P}(B_2\,|\,A^c \cap B_1).$$

We now make the reasonable assumption that B_1, B_2 are independent conditioned on A and A^c as well. Then, by the definition of conditional independence, we have

$$\begin{aligned}
\mathbb{Q}(B_2\,|\,A) &= \mathbb{P}(B_2\,|\,A \cap B_1) = \mathbb{P}(B_2\,|\,A) = p_+ = 0.99,\\
\mathbb{Q}(B_2\,|\,A^c) &= \mathbb{P}(B_2\,|\,A^c \cap B_1) = \mathbb{P}(B_2\,|\,A^c) = p_f = 0.10.
\end{aligned}$$

Hence,

$$p_2 = \mathbb{Q}(A\,|\,B_2) = \frac{\mathbb{Q}(B_2\,|\,A)\ \mathbb{Q}(A)}{\mathbb{Q}(B_2\,|\,A)\ \mathbb{Q}(A) + \mathbb{Q}(B_2\,|\,A^c)\ \mathbb{Q}(A^c)} = \frac{p_+\ p_1}{p_+\ p_1 + p_f\ (1-p_1)} = 0.7519.$$

It is interesting to note that

$$\begin{aligned}
\mathbb{Q}(B_2) &= \mathbb{Q}(B_2\,|\,A)\ \mathbb{Q}(A) + \mathbb{Q}(B_2\,|\,A^c)\ \mathbb{Q}(A^c) = p_+\ p_1 + p_f\ (1-p_1) = 0.30862\\
&> \mathbb{P}(B_2) = \mathbb{P}(B_1) = p_+\ p + p_f\ (1-p) = 0.1267.
\end{aligned}$$

Hence, B_2 is *not independent of* B_1. ■

Exercise 3.1 Suppose that a randomly selected person is tested positive three times. Compute the probability that this person is infected with $p = 3\%$, $p_+ = 90\%$ and $p_f = 10\%$. ■

■ **Example 3.3 — Application to a court case: People vs. Collins 1968.**

A couple with a certain ethnic characteristics in a yellow convertible was involved in a purse snatching. This combination of ethic characteristics and the type of the car was taken to be rare and a few days later a couple with the same ethnic characteristics in a yellow convertible was arrested based solely on their ethic characteristics and the type of their car. A statistician estimated that the probability that a randomly selected couple would posses these characteristics and a yellow convertible is about 8.3×10^{-8} or one in 12 million. Although it is not clear how this probability was estimated, based on this assumed very small probability they were convicted. The decision was appealed and was reversed by the Supreme Court of the United States, again based on statistical estimation.

The following calculation was presented to the Supreme Court. Let p be the probability that a randomly selected couple has these characteristics and own a yellow convertible. Suppose the population living in the relevant geographical area has n couples denoted by $c_1, c_2, \ldots, c_n$. Define then the following events:

- C_i be the event that the couple c_i has these characteristics,
- B be the event that there are at least one couple in this area with this these characteristics,
- A be the event that there are at least two,
- C is the event there is exactly one.

With this notation, we have $\mathbb{P}(C_i) = p \approx 8.3 \times 10^{-8}$. The conviction was based on the false impression that *since p is very small, there can only be one couple with these characteristics*. However, the relevant quantity to consider is the *conditional probability that, after observing one couple with these characteristics, there is no other couple with these characteristics*. Given the definitions of the events, this is exactly the conditional probability $\mathbb{P}(A^c|B)$. Then, the goal is to compute this conditional probability using the Bayes formula

$$\mathbb{P}(A^c|B) = 1 - \mathbb{P}(A|B) = 1 - \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Since A is a subset of B, $\mathbb{P}(A \cap B) = \mathbb{P}(A)$. We thus have to compute $\mathbb{P}(A)$ and $\mathbb{P}(B)$.

To compute $\mathbb{P}(A)$, notice first that $B = A \cup C$ and $A \cap C = \varnothing$. We thus have $\mathbb{P}(B) = \mathbb{P}(A) + \mathbb{P}(C)$, which implies

$$\mathbb{P}(A) = \mathbb{P}(B) - \mathbb{P}(C) = 1 - (1-p)^n - \mathbb{P}(C).$$

It therefore remains to compute $\mathbb{P}(C)$. For this, remark that:

$$C = \bigcup_{i=1}^{n} \left[C_i \cap \left(\bigcap_{j \neq i} C_j^c \right) \right].$$

Since $\mathbb{P}(C_i) = p$ and assuming that these events are independent (an assumption which could also be challenged), we now directly calculate that

$$\mathbb{P}\left(C_i \cap \left(\bigcap_{j \neq i} C_j^c \right) \right) = p(1-p)^{n-1},$$

and therefore $\mathbb{P}(C) = np(1-p)^{n-1}$.

Hence, we finally obtain

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} = \frac{\mathbb{P}(A)}{\mathbb{P}(B)} = \frac{1 - (1-p)^n - np(1-p)^{n-1}}{1 - (1-p)^n}$$

The population of the area of the purse snatching has about 8 million couples. First suppose that we accept the p-estimate used by the lower court; that is p is one in 12 million. With these numbers,

$$\mathbb{P}(C) = 0.3423, \quad \mathbb{P}(B) = 0.4866, \qquad \mathbb{P}(A) = 0.1443, \qquad \mathbb{P}(A|B) = 0.2966.$$

Therefore, the probability that there exists more than one couple with the required characteristics, knowing that there exists at least one, is approximately 30%, whereas the conditional probability of having at least is 14%. In other words, there is 30% chance that the selected couple is not the only one with these characteristics, and therefore not necessarily the guilty couple. Hence, the statistics shows that the convicted couple is guilty with only 70% probability *even if we accept the arbitrary estimate of* p. This is certainly not large enough for conviction solely based on a probability being small.

Additionally, with a p value of one in 5 million, $\mathbb{P}(A^c|B)$ is 40% and with p equal to one in a million, $\mathbb{P}(A^c|B)$ is 0.26%. This very strong dependence on the subjective and difficult to estimate p value is another convincing argument for the reversal. ∎

Exercise 3.2 For p values one in 10 million, one in 4 million and one in a million, compute $\mathbb{P}(A)$, $\mathbb{P}(B)$, $\mathbb{P}(A|B)$ and the "learning ratio" $\mathbb{P}(A|B)/\mathbb{P}(A)$. ∎

∎ Example 3.4 — Dependence on the prior.

Suppose that a coin is flipped 100 times with heads in 52 of them. Let P be the probability of heads. We are not sure that the coin is fair. But we are certain that P can only take values $k/10$ and our prior is

$$\mathbb{P}(P = \frac{k}{10}) = p_k, \qquad \text{for } k = 0, 1, \ldots, 10.$$

Clearly we must have each $p_k \in [0,1]$ and $\sum_k p_k = 1$. Let B be the observed event; namely, 52 heads in 100 trials. Let $A_k = \{P = k/10\}$ so that $\mathbb{P}(A_k) = p_k$. We also calculate that

$$\mathbb{P}(B|A_k) = \binom{100}{52} \left(\frac{k}{10}\right)^{52} \left(\frac{10-k}{10}\right)^{48}.$$

As we are certain that P can only take values $k/10$, the events $\{A_k\}_{k=0,\ldots,10}$ is a partition and we can use the Bayes' formula:

$$\widehat{p_k} = \mathbb{P}(A_k|B) = \frac{\mathbb{P}(B|A_k)\,\mathbb{P}(A_k)}{\sum_{i=0}^{10} \mathbb{P}(B|A_i)\,\mathbb{P}(A_i)}, \qquad k = 0, 1, \ldots, 10. \tag{3.2}$$

As an example, suppose that $p_k = 1/3$ for $k = 4, 5, 6$ and zero otherwise, Then,

$$\widehat{p_k} = \begin{cases} 0.0427, & k = 4 \\ 0.7407, & k = 5 \\ 0.216, & k = 6 \\ 0. & \text{otherwise} \end{cases}$$

If we had the prior $p_k = 1/2$ for $k = 4, 6$ and p_k is zero otherwise, then,

$$\widehat{p_k} = \begin{cases} 0.164, & k = 4 \\ 0.835, & k = 6 \\ 0. & \text{otherwise} \end{cases}$$

This shows a strong dependence on the prior. In particular, if a certain event is assumed to have zero probability then it can not be changed. In the next section, we remove the assumption that p can only take certain values and we will have a very different result. ∎

Exercise 3.3 Consider the above example with a general prior p_k. For $t = 1,\dots,100$, let x_t be the number of heads in the first t trials. We know that $p_{100} = 52$. An impatient mathematician instead of waiting until all trials are completed, computes the Bayesian posterior after every trail. Let

$$\widehat{p}_k(t,x) := \mathbb{P}(A_k \mid x_t = x).$$

Does $\widehat{p}_k(100,52)$ agree with the formula (3.2)?

Hint. Show that for any $t \geqslant 0$ and $x \in \{0,1,\dots,t\}$

$$\widehat{p}_k(t+1,x+\zeta) = \frac{\widehat{p}_k(t,x)}{c(t+1,x+\zeta)} \left(\frac{k}{10}\right)^{\zeta} \left(\frac{10-k}{10}\right)^{1-\zeta}, \qquad \text{for } \zeta = 0,1,$$

for some constant $c(t+1,x+\zeta)$. There is no need to compute this constant. ∎

Exercise 3.4 Suppose that X,Y are discrete-valued random variables with X taking values $\{x_1,\dots,x_N\}$ and Y taking values $\{y_1,\dots,y_M\}$. Show the following formula which can be interpreted as the Bayes formula for discrete-valued random variables:

$$\mathbb{P}(X = x_i | Y = y_k) = \frac{\mathbb{P}(Y = y_k | X = x_i)\ \mathbb{P}(X = x_i)}{\sum_{j=1}^{N} \mathbb{P}(Y = y_k | X = x_j)\ \mathbb{P}(X = x_j)}, \qquad i = 1,\dots,N,\ k = 1\dots,M.$$

Hint. The events $A_j := \{X = x_j\}$ forms a partition. One can then use the formula (3.1). ∎

3.1.2 Bayes for the continuous case

Suppose that X and Y are two random variables with probability density functions f_X, f_Y and a joint density $f_{X,Y}$. Then, formally

$$f_X(x)\,\mathrm{d}x \approx \mathbb{P}(X \in (x,x+\mathrm{d}x)), \quad f_Y(y)\,\mathrm{d}y \approx \mathbb{P}(Y \in (y,y+\mathrm{d}y)),$$

$$f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y \approx \mathbb{P}(X \in (x,x+\mathrm{d}x) \text{ and } Y \in (y,y+\mathrm{d}y)).$$

We use these to derive a formula for the probability density of X given Y is,

$$\begin{aligned} f_{X|Y}(x|y)\,\mathrm{d}x &\approx \mathbb{P}(X \in (x,x+\mathrm{d}x) \mid Y \in (y,y+\mathrm{d}y)) \\ &\approx \frac{\mathbb{P}(X \in (x,x+\mathrm{d}x) \text{ and } Y \in (y,y+\mathrm{d}y))}{\mathbb{P}(Y \in (y,y+\mathrm{d}y))} \approx \frac{f_{X,Y}(x,y)\mathrm{d}x\,\mathrm{d}y}{f_Y(y)\mathrm{d}y} \\ &= \frac{f_{X,Y}(x,y)}{f_Y(y)}\,\mathrm{d}x. \end{aligned}$$

Hence,

Conditional density formula.

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}, \qquad x,y \in \mathbb{R}.$$

As $f_{X,Y}(x,y) = f_{Y|X}(y|x)\,f_X(x)$ and

$$f_Y(y) = \int f_{X,Y}(x',y)\,\mathrm{d}x' = \int f_{Y|X}(y|x')\,f_X(x')\,\mathrm{d}x',$$

we have the useful continuous Bayes' formula,

Continuous Bayes formula.

$$f_{X|Y}(x|y) = \frac{f_{Y|X}(y|x) f_X(x)}{f_Y(y)} = \frac{f_{Y|X}(y|x) f_X(x)}{\int f_{Y|X}(y|x') \, f_X(x') \, \mathrm{d}x'}, \qquad x, y \in \mathbb{R}. \tag{3.3}$$

3.1.3 Bayes for the mixed case

For now, we developed the Bayes formula in the case where X and Y are both discrete or both continuous. However, it is possible to have some cases where X is continuous but not Y, or the converse. For example, in the previous situation with the coin flip, if our prior distribution on p, the probability of a head, is uniform on $[0,1]$, it is a continuous random variable. On the contrary, the observation we make and that we want to use to compute the posterior probability, namely the number of heads in the 100 coin-flips, is not a continuous random variable. We study these cases next.

a. Continuous X, discrete Y.

Suppose that X is a random variable with a probability density function $f_X(x)$ and suppose that Y takes values in $\{y_1, \dots, y_n, \dots\}$ and the probabilities $\mathbb{P}(Y = y_k)$ are known for each k. Additionally, we known the conditional probabilities $\mathbb{P}(Y = y_k \,|\, X = x)$ are known for each k and $x \in \mathbb{R}$. Then,

$$\mathbb{P}(Y = y_k) = \int \mathbb{P}(Y = y_k \,|\, X = x') \, f_X(x') \, \mathrm{d}x',$$

and the Bayes' formula takes the form:

Continuous X, Discrete Y.

$$f_{X|Y}(x|y_k) = \frac{\mathbb{P}(Y = y_k \,|\, X = x) \, f_X(x)}{\mathbb{P}(Y = y_k)} = \frac{\mathbb{P}(Y = y_k \,|\, X = x) \, f_X(x)}{\int \mathbb{P}(Y = y_k \,|\, X = x') \, f_X(x') \, \mathrm{d}x'}, \qquad x \in \mathbb{R}, \; k = 1, \dots$$

We continue with the coin-flip Example 3.4.

▪ Example 3.5

As in the previous section, let P be the probability of a head. Further let Y be the number of heads in 100 coin flips and $B = \{Y = 52\}$. We now assume that the prior distribution of P is uniform over $[0,1]$ and we compute the posterior distribution of P given that the event B is observed. Let A be a subset of $[0,1]$. Then,

$$\mathbb{P}(\{P \in A\} \,|\, B) = \frac{\mathbb{P}(\{P \in A\} \text{ and } Y = 52)}{\mathbb{P}(Y = 52)} = \frac{\mathbb{P}(Y = 52 \,|\, \{P \in A\}) \, \mathbb{P}(\{P \in A\})}{\mathbb{P}(Y = 52)}.$$

Since the prior distribution is uniform of P is uniform, i.e, $f_P \equiv 1$,

$$\mathbb{P}(Y = 52) = \int_0^1 \mathbb{P}(Y = 52 \,|\, P = p') \, f_P(p') \, \mathrm{d}p' = \int_0^1 \binom{100}{52} (p')^{52} \, (1 - p')^{48} \mathrm{d}p' =: c_{52}.$$

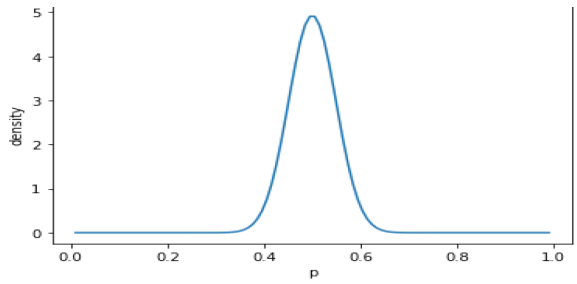
We proceed formally and take $A = (p, p + \mathrm{d}p)$ for some $p \in (0,1)$. Then,

$$\mathbb{P}(Y = 52 \,|\, \{P \in A\}) = \mathbb{P}(Y = 52 \,|\, \{P = p\}) = \binom{100}{52} p^{52} \, (1 - p)^{48},$$

and hence,

$$f_{P|B}(p) := \mathbb{P}(P \in (p, p + \mathrm{d}p) \,|\, B) = C \, p^{52} \, (1 - p)^{48}, \quad \text{where} \quad C = \binom{100}{52} \frac{1}{c_{52}}.$$

This is the Beta distribution with parameters $a = 53$, $b = 49$. Its mean is $a/(a+b) = 53/102 \approx 0.5196$ and its graph is:



Note that even with an agnostic prior such as the uniform, the posterior distribution concentrates around the estimate $\widehat{p} = 52/100$. However, the mean 0.5196, which can be interpreted as the Bayesian estimate of P, is slightly different that the classical estimate $\widehat{p} = 0.52$.

■

The followings are two more examples of this case.

Exercise 3.5 Suppose that $Y_1, \ldots, Y_n$ are i.i.d. from the Poisson distribution with mean $\theta > 0$, i.e., for any $k = 1, \ldots$, conditioned on the value of θ,

$$\mathbb{P}(Y_k = y \,|\, \theta) = \frac{\theta^y}{y!} e^{-\theta}, \qquad y = 0, 1, \ldots$$

The parameter θ is not known and hence, is a random variable. Suppose that the prior distribution of θ is the gamma distribution with parameters $a, b > 0$, i.e., the p.d.f. is given by,

$$f_\theta(\theta) = \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta}, \qquad \theta > 0,$$

where Γ is the *gamma function*:

$$\Gamma(a) := \int_0^\infty z^{a-1} e^{-z} dz.$$

In this exercise you do not need the properties of the Γ function.

a. Show that the posterior distribution of θ based on the observation that $Y_1 = y_1$ is again gamma with parameters,

$$a + y_1, \qquad b + 1.$$

b. More generally, show that the posterior distribution of θ based on the observations $Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n$ is again gamma with parameters,

$$a + \sum_{i=1}^n y_i, \qquad b + n.$$

■

Exercise 3.6 This is related to a court case *Castaneda vs. Partida* 430 U.S. 482 (1977) deciding whether there was bias in juror selection in a certain district. The local population of that district was 79.1% Mexican-American. During a 2.5-year period, there were 220 persons called on serve grand juries, but only 100 were Mexican-American.

Court assumed that the selection of jurors were random with P being the probability of any selected juror being Mexican-American. If X denotes the number of Mexican-American jurors, then X is a Binomial random $n = 220$ and success probability P. That is what the Court assumes based on common-sense. Namely, the conditional distribution of X given $P = p$ is Binomial:

$$\mathbb{P}(X = x \,|\, P = p) = \binom{220}{x} p^x (1-p)^{220-x}, \qquad \text{for } x = 0, 1, \ldots, 220.$$

Based of the population percentage, if one uses a success probability of 0.791 (we do not mean that P is in fact equal to this value!), then $\mathbb{E}[X] = 174.02$. A value that is substantially smaller than the actually observed value of 100. The question is, based on the observation that $X = 100$, whether this is overwhelming evidence of bias in the jury selection procedure.

Mathematically, the bias would mean the the random variable P defined above is substantially smaller than 0.791. Say we define the bias to be P being 80% or less than the population percentage of 79.1%. Equivalently, the event $P \leqslant 0.8 \times 0.791 = 0.6328$ means bias. Therefore, we have to compute the conditional probability of this event based on the observation that $X = 100$.

Estimating a parameter like P is very similar to election predictions and there is a lot experience. Based on this scientific experience, the Court assumed that the distribution of P prior to the observation was beta with parameters a and b, i.e.,

$$f_P(p) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1} (1-p)^{b-1}, \qquad p \in (0,1).$$

As the prior must be consistent with the population, the parameters must satisfy $\mathbb{E}[P] = a/(a+b) = 0.791$, or $a = 3.785\, b$.

a. Show that the conditional p.d.f. of P given that $X = 100$ is again a beta distribution with parameters $a + 100$ and $b + 120$.

b. As $a = 3.785\, b$, the conditional probability $\mathbb{P}(\, P \leqslant 0.6328 \mid X = 100\,)$ is only a function b. Compute and plot this probability as a function of $b \in [0, 100]$.

c. Find the smallest value of b for which this probability is less than 0.5.

d. Compute the unconditional $\mathbb{P}(\, P \leqslant 0.6328\,)$ using the prior with the b value computed in above part.

e. Try to justify the following statement with precise numeric values. The above calculations prove that if initially one accepts even with a small probability that there is bias, then after the observation of $X = 100$ the probability of bias is overwhelming. Stating the opposite direction: in order to believe that there is no-bias, initially one must give almost zero probability to the possibility that there is bias.

■

b. Discrete X, continuous Y.

This is the less common case. Here, X takes discrete values in $\{x_1, \ldots, x_n, \ldots\}$ and the probabilities $\mathbb{P}(X = x_k)$ are known for each k, while Y is a random variable with a probability density function $f_Y(y)$. Additionally, we known the conditional probability density function $f_{Y|X}(y|x_k)$ is known for each k and $y \in \mathbb{R}$. Then,

$$f_Y(y) = \sum_j f_{Y|X}(y|x_j)\mathbb{P}(X = x_j),$$

and the Bayes' formula takes the form:

Discrete X, Continuous Y.

$$\mathbb{P}(X = x_k|Y = y) = \frac{f_{Y|X}(y|x_k)\mathbb{P}(X = x_k)}{\sum_j f_{Y|X}(y|x_j)\mathbb{P}(X = x_j)}, \qquad \forall\, y \in \mathbb{R}, \text{ and } k = 1,2,\ldots$$

3.1.4 Gaussian Case

The special case when X and Y are jointly Gaussian is encountered often in the applications. The celebrated Kalman filter that we study in the next chapter being a prime example. Therefore, in this subsection we derive general formulae for this particular structure.

We first recall that for a Gaussian random variable $Z \in \mathbb{R}^m$ the probability density function is given by,

$$f_Z(z) = (2\pi|det(\Sigma_Z))^{-m/2} \exp(-\frac{1}{2}(z-\mu_z)^\top \Sigma_Z^{-1}(z-\mu_Z)), \qquad z \in \mathbb{R}^m,$$

where $\mu_Z \in \mathbb{R}^m$ is the mean of Z and Σ_Z is the $(m \times m)$ symmetric, positive covariance matrix of Z.

To motivate the general theory, we start with a in a one-dimensional example.

■ **Example 3.6** Suppose X,Y are jointly Gaussian with means $\mu_X = 1, \mu_Y = 2$, $\sigma_X = 1$, $\sigma_Y = 2$ and correlation of $1/2$. Hence, the covariance matrix is given by,

$$\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}, \qquad \Rightarrow \qquad \Sigma^{-1} = \frac{1}{3}\begin{bmatrix} 4 & -1 \\ -1 & 1 \end{bmatrix},$$

and

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-1)^2}, \quad f_Y(y) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{1}{8}(y-2)^2}, \quad f_{X,Y}(x,y) = \frac{1}{2\pi\sqrt{3}} e^{-Q(x,y)},$$

where

$$Q(x,y) = \frac{1}{2}\Sigma^{-1}\begin{bmatrix} x-1 \\ y-2 \end{bmatrix} \cdot \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} = \frac{1}{6}\left[4(x-1)^2 - 2(x-1)(y-2) + (y-2)^2\right].$$

A tedious but direct calculation yields,

$$f_{X|Y}(x|2) = \frac{f_{X,Y}(x,2)}{f_Y(y)} = \frac{2}{\sqrt{6\pi}} e^{-Q(x,2)} = \frac{2}{\sqrt{6\pi}} e^{-\frac{2}{3}(x-1)^2}.$$

Hence, X given the event $\{Y = 2\}$ is also Gaussian with mean 1 and variance $3/4$. ■

Exercise 3.7 In the above example, compute $f_{X|Y}(x|y)$ for a general y. ■

In fact, the above example generalizes and the conditional density of jointly Gaussian random variable is again Gaussian. The formula that we state below without a proof is proved by a direct calculation using the continuous Bayes' formula (3.3) and the Gaussian densities. This formula is central to Kalman filter. Please recall to the notations provided at the beginning of this Chapter.

Theorem 3.1.1 Suppose that $X \in \mathbb{R}^k, Y \in \mathbb{R}^d$ are jointly Gaussian random variables. Then, the random variable X conditioned on Y, denoted by $X|Y$, is also Gaussian with

$$\mathbb{E}[X|Y] = \mathbb{E}[X] + \mathbb{C}(X,Y)\mathbb{C}(Y)^{-1}\,(Y - \mathbb{E}[Y]),$$

or equivalently,

$$\mu_{X|Y=y} = \mathbb{E}[X|Y=y] = \mu_X + \mathbb{C}(X,Y)\mathbb{C}(Y)^{-1}\,(y-\mu_Y),$$

and the $k \times k$ conditional covariance matrix is given by,

$$\mathbb{C}(X|Y) = \mathbb{C}(X) - \mathbb{C}(X,Y)\mathbb{C}(Y)^{-1}\mathbb{C}(X,Y)^{\top}.$$

Then, the condition probability density is given by,

$$f_{X|Y}(x|y) = (2\pi|\det(\mathbb{C}(X|Y)|)^{-k/2}\,\exp(-\frac{1}{2}(x-\mu_{X|Y=y})^{\top}\mathbb{C}(X|Y)^{-1}(x-\mu_{X|Y=y})), \qquad x \in \mathbb{R}^k.$$

Proof. Without loss of generality we may assume that $\mathbb{E}[X] = \mathbb{E}[Y] = 0$. Set

$$Z := X - CY, \qquad \text{where} \qquad C = \mathbb{C}(X,Y)\,\mathbb{C}(Y)^{-1}.$$

It is clear that $\mathbb{E}[Z] = 0$. For $i = 1,\dots,k$ and $j = 1,\dots,d$, as $(CY)_j = \sum_{n=1}^{d} C_{j,n}Y_n$,

$$\mathbb{E}[Z_iY_j] = \mathbb{E}[X_iY_j] - \sum_{n=1}^{d} C_{j,n}\mathbb{E}[Y_jY_n] = \mathbb{C}(X,Y)_{i,j} - \sum_{n=1}^{d} C_{i,n}\mathbb{C}(Y)_{n,j}.$$

In matrix notation, $\mathbb{C}(Z,Y) = \mathbb{C}(X,Y) - C\mathbb{C}(Y) = \mathbb{C}(X,Y) - \mathbb{C}(X,Y)\,\mathbb{C}(Y)^{-1}\mathbb{C}(Y) = 0$. As Z and Y are Gaussian, this means that Z and Y are independent. Since $\mathbb{E}[Z] = 0$,

$$\mathbb{E}[X\,|\,Y] = \mathbb{E}[Z\,|\,Y] + CY = \mathbb{E}[Z] + \mathbb{C}(X,Y)\,\mathbb{C}(Y)^{-1}\,Y = \mathbb{C}(X,Y)\,\mathbb{C}(Y)^{-1}\,Y = CY.$$

Hence, $X - \mathbb{E}[X\,|\,Y] = Z$. Since $\mathbb{E}[Z] = 0$ and Z is independent of Y, for $i,n = 1,\dots,k$,

$$\begin{aligned}
(\mathbb{C}(X|Y))_{i,n} &= \mathbb{E}[(X - E(X|Y))_i\,(X - E(X|Y))_n\,|\,Y] \\
&= E[Z_iZ_n] = \mathbb{E}[Z_i\,(X-CY)_n] = \mathbb{E}[Z_iX_n] - \mathbb{E}[(CY)_n]\,\mathbb{E}[Z_i] \\
&= \mathbb{E}[(X-CY)_iX_n] = \mathbb{E}[X_iX_n] - \mathbb{E}[(CY)_iX_n] \\
&= \mathbb{C}(X)_{i,n} - \left(C\,\mathbb{C}(X,Y)^T\right)_{i,n}.
\end{aligned}$$

This implies that $\mathbb{C}(X|Y) = \mathbb{C}(X) - \mathbb{C}(X,Y)\,\mathbb{C}(Y)^{-1}\,\mathbb{C}(X,Y)^T$.

■

Exercise 3.8 Consider the one-dimensional case $X,Y \in \mathbb{R}$ with correlation $\rho_{X,Y} \in [-1,1]$ and variances $\mathrm{var}(X) = \sigma_X^2$, $\mathrm{var}(Y) = \sigma_Y^2$. Then, $\mathbb{C}(X,Y) = \rho_{X,Y}\sigma_X\sigma_Y$. Derive the conditional density from (3.3) and verify that it agrees with the above formula. In particular, show that

$$\mathbb{E}[X|Y] = \mathbb{E}[X] + \frac{\rho_{X,Y}\sigma_X}{\sigma_Y}(Y - \mathbb{E}[Y]), \quad \text{and} \quad \mathrm{var}(X|Y) = \sigma_X^2(1-\rho^2).$$

■

Solution. We write ρ for $\rho_{X,Y}$. We first express the hypothesis on X and Y in terms of densities :

$$f_X(x) = \frac{1}{\sigma_X\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2\sigma_X^2}(x-\mu_X)^2}, \quad f_Y(y) = \frac{1}{\sigma_Y\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2\sigma_Y^2}(y-\mu_Y)^2}, \quad \text{and} \quad \Sigma = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix},$$

where $\Sigma = \mathbb{C}((X,Y),(X,Y))$ is the full covariance matrix. Then, $\det(\Sigma) = (1-\rho^2)\sigma_X^2\sigma_Y^2$ and then the inverse of Σ:

$$\Sigma^{-1} = \frac{1}{\det(\Sigma)}\begin{pmatrix} \sigma_Y^2 & -\rho\sigma_X\sigma_Y \\ -\rho\sigma_X\sigma_Y & \sigma_X^2 \end{pmatrix} = \frac{1}{(1-\rho^2)\sigma_X^2\sigma_Y^2}\begin{pmatrix} \sigma_Y^2 & -\rho\sigma_X\sigma_Y \\ -\rho\sigma_X\sigma_Y & \sigma_X^2 \end{pmatrix}.$$

We can then compute the joint density of the pair (X,Y), defined by:

$$f_{X,Y}(x,y) = \frac{1}{2\pi\sqrt{\det(\Sigma)}} e^{-Q(x,y)}, \quad \text{where } Q(x,y) = \frac{1}{2}\begin{pmatrix} x-\mu_X & y-\mu_Y \end{pmatrix} \Sigma^{-1} \begin{pmatrix} x-\mu_X \\ y-\mu_Y \end{pmatrix}.$$

By standard computations, we obtain:

$$Q(x,y) = \frac{1}{2}\frac{1}{1-\rho^2}\left(\left(\frac{x-\mu_X}{\sigma_X}\right)^2 + \left(\frac{y-\mu_Y}{\sigma_Y}\right)^2 - 2\rho\frac{x-\mu_X}{\sigma_X}\frac{y-\mu_Y}{\sigma_Y} \right).$$

To simplify the computations below, denote

$$\tilde{x} := \frac{x-\mu_X}{\sigma_X}, \text{ and } \tilde{y} := \frac{y-\mu_Y}{\sigma_Y}.$$

We can then use (3.3) to compute the conditional density:

$$\begin{aligned} f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} &= \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} e^{-Q(x,y)} \left(\frac{1}{\sigma_Y\sqrt{2\pi}} e^{-\frac{1}{2\sigma_Y^2}(y-\mu_Y)^2} \right)^{-1} \\ &= \frac{\sqrt{2\pi}}{2\pi\sigma_X\sqrt{1-\rho^2}} \exp\left(-Q(x,y) + \frac{1}{2}\tilde{y} \right) \\ &= \frac{1}{\sigma_X\sqrt{1-\rho^2}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{1}{1-\rho^2}\left(\tilde{x}-\rho\tilde{y}\right)^2 \right). \end{aligned}$$

by noticing that

$$-Q(x,y) + \frac{1}{2}\tilde{y} = -\frac{1}{2}\frac{1}{1-\rho^2}\left(\tilde{x}^2 + \rho^2\tilde{y}^2 - 2\rho\tilde{x}\tilde{y}\right) = -\frac{1}{2}\frac{1}{1-\rho^2}\left(\tilde{x}-\rho\tilde{y}\right)^2.$$

Remark that we have:

$$\tilde{x} - \rho\tilde{y} = \frac{x-\mu_X}{\sigma_X} - \rho\frac{y-\mu_Y}{\sigma_Y} = \frac{\sigma_Y x - \sigma_Y\mu_X - \rho\sigma_X y + \rho\sigma_X\mu_Y}{\sigma_X\sigma_Y} = \frac{x-\mu_{X|Y=y}}{\sigma_X}$$

with

$$\mu_{X|Y=y} := \mu_X + \frac{\rho\sigma_X}{\sigma_Y}\left(y-\mu_Y\right).$$

Furthermore, by denoting:

$$\sigma^2_{X|Y=y} := (1-\rho^2)\sigma_X^2$$

we finally obtain

$$f_{X|Y}(x|y) = \frac{1}{\sigma_{X|Y=y}\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2_{X|Y=y}}\left(x-\mu_{X|Y=y}\right)^2 \right).$$

We recognize the density of a Gaussian random variable with mean $\mu_{X|Y=y}$ and variance $\sigma^2_{X|Y=y}$. To conclude, the random variable $X|Y$ is also Gaussian, with mean $\mu_{X|Y=y}$ and variance $\sigma^2_{X|Y=y}$ defined above.

■

3.2 Estimation

Estimation is one of the most important topic in statistics. Here we only cover a rather limited part of this vast subject. The set-up that we consider has two random variables X, Y and we observe Y. Based on this observation, we want to estimate either X or more generally a function h of X. We consider two approaches, one is the Bayesian estimators which is an application of the Bayes' formula. The second is the Maximum Likelihood Estimator (MLE). Of course, there are many other approaches but in this course Bayesian approach will be main tool and we discuss MLE to provide a comparison to another method which is widely used.

3.2.1 Bayesian Estimators

We consider the case in which X, Y are continuous random variables with a probability density. We observe Y and with knowledge, we want to estimate $h(X)$ for a given h.

Definition 3.2.1 Given a *loss* function ℓ, the *Bayesian estimate* is given as

$$\widehat{h(X)} := \operatorname{argmin}_a \mathbb{E}\left[\ell(a, h(X)) \,|\, Y\right].$$

Theorem 3.2.1 With quadratic loss function $\ell(\theta, h(X)) = |\theta - h(X)|^2$, the Bayesian estimator is given as the condition expectation:

$$\boxed{\widehat{h(X)} = \mathbb{E}[h(X)|Y = y].}$$

In particular, for continuous random variables the Bayes' formula implies that

$$\widehat{h(X)} = \int h(x)\, f_{X|Y}(x|y)\, \mathrm{d}x = \int h(x)\, \frac{f_{X,Y}(x,y)}{f_Y(y)}\, \mathrm{d}x = \int h(x)\, \frac{f_{Y|X}(y|x)\, f_X(x)}{f_Y(y)}\, \mathrm{d}x.$$

A similar formulae also holds when X or Y are discrete as well. The above result provides a way to construct a relevant estimator $\widehat{h}(X)$ for $h(X)$ given Y. This is known as the minimum mean square error (MMSE) estimator. The use of the Bayes formula highlight that, in order to construct this estimator, we need, in addition to the density f_Y for Y, either:

(i) the joint density $f_{X,Y}$ of the pair (X, Y);
(ii) or the conditional density $f_{Y|X}$ of $Y|X$ and the density f_X for X.

Proof. Set $H := \mathbb{E}[h(X) \,|\, Y]$. Then, for any a

$$\begin{aligned}\mathbb{E}\left[\ell(a, h(X)) \,|\, Y\right] &= \mathbb{E}\left[\,|(a - H) + (H - h(X))|^2 \,|\, Y\right] \\ &= \mathbb{E}\left[\,(a-H)^2 \,|\, Y\right] + \mathbb{E}\left[\,(H - h(X))^2 \,|\, Y\right] + 2\mathbb{E}\left[\,(a-H)\,(H - h(X)) \,|\, Y\right].\end{aligned}$$

Since $(a - H) = (a - \mathbb{E}[h(X) \,|\, Y])$ is a function of Y, its conditional expectation with respect to Y is equal to itself. Hence, we have the following identities:

$$E\left[\,(a-H)^2 \,|\, Y\right] = (a-H)^2, \qquad \text{and} \qquad \mathbb{E}[H - h(X) \,|\, Y] = H - \mathbb{E}[h(X) \,|\, Y] = 0.$$

Hence,

$$\mathbb{E}\left[\,(a-H)\,(H - h(X)) \,|\, Y\right] = (a-H)\,\mathbb{E}[H - h(X) \,|\, Y] = 0.$$

Using these, we conclude that

$$\mathbb{E}\left[\ell(a, h(X)) \,|\, Y\right] = (a-H)^2 + \mathbb{E}\left[\,(H - h(X))^2 \,|\, Y\right]$$

Since the second term is independent of a,

$$\widehat{h(X)} := \operatorname{argmin}_a \mathbb{E}[\,\ell(a,h(X))\,|\,Y\,] = \operatorname{argmin}_a (a-H)^2 = H = \mathbb{E}[h(X)\,|\,Y\,].$$

■

■ **Example 3.7** Consider the Example 3.6: X,Y are jointly Gaussian with means $\mu_X = 1, \mu_Y = 2$, $\sigma_X = 1$, $\sigma_Y = 2$ and correlation of $\rho = 1/2$. Then, in view of Exercise 3.8, the conditional random variable $X|Y$ is Gaussian with,

$$\mathbb{E}[X|Y=y] = \mathbb{E}[X] + \frac{\rho\sigma_X}{\sigma_Y}\,(y-\mathbb{E}[Y]) = 1 + \frac{1}{4}(y-2) = \frac{(6-y)}{4}.$$

and

$$\mathbb{C}(X|Y) = \sigma_X^2(1-\rho^2) = \frac{3}{4}.$$

Then, for any y, the Bayesian estimate for $h(X)$ given $\{Y=y\}$ is

$$\widehat{h(X)} = \mathbb{E}[h(X)|Y=y] = \int_{-\infty}^{\infty} \frac{2}{\sqrt{6\pi}}\, h(x)\, e^{-\frac{(4x+y-6)^2}{24}}\, \mathrm{d}x.$$

With $y=2$, we have

$$\widehat{X} = \mathbb{E}[X|Y=2] = 1, \qquad \widehat{X^2} = \mathbb{C}(X|Y=2) + (\mathbb{E}[X|Y=2])^2 = \frac{7}{4}.$$

■

3.2.2 Maximum Likelihood Estimator

We compare this popular estimator with the Bayesian one. In this approach, one postulates a parametric family of distributions. Then, given an observation, we first find the parameter that maximizes a *likelihood function* and use the distribution corresponding to this parameter to compute the estimates. In all our examples, the likelihood function is the conditional density. We start with an example.

■ **Example 3.8** We revisit the coin-flipping Examples 3.4 and 3.5. Here Y is the number heads in 100 flips and the head probability is not known. In the Bayesian approach, we need a prior for this probability. In the MLE, we treat it as a parameter p. Then, the probability distribution of Y is

$$\mathbb{P}(Y=k;p) = \binom{100}{k}\, p^k\, (1-p)^{100-k}, \qquad k = 0,1,\ldots,100.$$

Suppose we observe that $Y=52$ and try to estimate p. A convenient likelihood function in this case is

$$L(p) := \ln(\mathbb{P}(Y=k;p)) = 52\ln(p) + 48\ln(1-p) + \text{constant}.$$

We maximize $\ell(p)$ to obtain an estimate p^* of the head probability:

$$0 = L'(p^*) = \frac{52}{p^*} - \frac{48}{1-p^*}, \quad \Rightarrow \quad 52(1-p^*) - 48p^* = 0, \quad \Rightarrow \quad p^* = \frac{52}{100}.$$

Compare this to Example 3.5. In that example, the mean of the posterior distribution is $53/102$ which is not exactly equal to the MLE estimate of $52/100$, but it is very close.

In the Example 3.4, the posterior distribution $\widehat{p}_k = \mathbb{P}(P = \frac{k}{10} \,|\, Y = 52)$ for $k = 0, 1, \ldots 10$. Then, the Bayesian estimate $\hat{P}_b$ of p would be

$$\hat{P}_b = \sum_{k=1}^{10} \frac{k}{10} \widehat{p}_k.$$

In most examples, this estimate differs from the natural MLE estimate of $52/100$. This is caused by the influence of the prior.

■

■ **Example 3.9** Suppose that $\mathbf{Y} := (Y_1, Y_2, \ldots, Y_n)$ are i.i.d. random variables that are uniformly distributed on $[0, \theta]$ where $\theta > 0$ is an unknown parameter. That is

$$f_{Y_i}(x; \theta) = \frac{1}{\theta}, \quad \text{if } x \in [0, \theta] \quad \text{and is equal to zero otherwise.}$$

The joint density is then given by,

$$f_{\mathbf{Y}}(y_1, \ldots, y_n\,; \theta) = \prod_{i=1}^{n} f_{Y_i}(y; \theta) = \begin{cases} \frac{1}{\theta^n}, & \text{if} \quad y_i \in [0, \theta,] \ \ \forall i = 1, \ldots, n, \\ 0, & \text{else.} \end{cases}$$

For a given n-tuble $x = (y_1, \ldots, y_n)$ with $y_i > 0$, the map $\theta \mapsto f_{\mathbf{Y}}(y; \theta)$ is maximized at

$$\theta^* = \max\{y_1, \ldots, y_n\}.$$

Hence, $\widehat{\theta}_{MLE} = \max\{Y_1, \ldots, Y_n\}$ is the ML estimator of θ. We continue by calculating the distribution of the random variable θ^* in terms of the unknown parameter θ. Indeed, for any $x \in [0, \theta]$,

$$F_{\theta*;\theta}(x) = \mathbb{P}(\max\{Y_1, \ldots, Y_n\} \leqslant x) = \prod_{i=1}^{n} \mathbb{P}(Y_i \leqslant x) = \left(\frac{x}{\theta}\right)^n.$$

Then, the probability density of θ^* is

$$f_{\theta*}(x; \theta) = \frac{\mathrm{d}}{\mathrm{d}x} F_{\theta*;\theta}(x) = n x^{n-1} \theta^{-n}, \qquad y \in [0, \theta].$$

This implies that

$$\mathbb{E}_\theta[\theta^*] = \int_0^\theta x \, f_{\theta*}(x; \theta) \, \mathrm{d}x = \frac{n}{n+1} \, \theta.$$

Hence, θ^* is a *biased* (because $\mathbb{E}[\theta^*] \neq \theta$) but a *consistent* (because $\lim_{n\to\infty} \theta^* = \theta$) estimate of θ. Also, we directly calculate that (details left as an exercise)

$$var(\theta^*) = \frac{n}{(n+1)^2 \, (n+2)} \, \theta^2 \ \to \ 0 \quad \text{as } n \text{ tends to infinity.}$$

Hence, the estimate θ^* converges to the true value θ with probability one as n approaches to infinity.

■

■ **Example 3.10** We now consider the previous example and obtain a Bayesian estimate of θ. To simplify we take $n = 1$ and use $Y = Y_1$.

Let $f_\theta(\theta)$ be the prior density of θ. Note that, this function is not needed in the MLE approach. We know that for any $0 < y < \theta$,

$$f_{Y|\theta}(y|\theta) = \frac{1}{\theta}, \qquad f_{Y,\theta}(y, \theta) = f_{Y|\theta}(y|\theta) \, f_\theta(\theta) = \frac{f_\theta(\theta)}{\theta}, \quad \text{for } 0 < y < \theta.$$

Moreover,

$$f_Y(y) = \int f_{Y,\theta}(y,\theta)\,\mathrm{d}\theta = \int_y^\infty \frac{f_\theta(\theta)}{\theta}\,\mathrm{d}\theta.$$

We can also compute the conditional density of θ given Y. For $y \in [0,\theta]$,

$$f_{\theta|Y}(\theta|y) = \frac{f_{Y,\theta}(y,\theta)}{f_Y(y)} = \frac{f_{Y|\theta}(y|\theta)\,f_\theta(\theta)}{f_Y(y)} = \frac{f_\theta(\theta)}{\theta\, f_Y(y)},$$

and it is zero if $y \notin [0,\theta]$.

As an example suppose that f_θ is exponential with parameter one:

$$f_\theta(\theta) = e^{-\theta}, \quad \theta \geqslant 0, \qquad f_Y(y) = \int_y^\infty \frac{1}{\theta\, e^\theta}\,\mathrm{d}\theta.$$

Then,

$$f_{\theta|Y}(\theta|y) = \frac{1}{f_Y(y)\,\theta\, e^\theta}, \quad \theta > y.$$

Therefore, the Bayesian estimate of θ given $\{Y = y\}$ is

$$\widehat{\theta}_b = \int \theta\, f_{\theta|Y}(\theta|y)\,\mathrm{d}\theta = \frac{\int_y^\infty e^{-\theta}\,\mathrm{d}\theta}{f_Y(y)} = \frac{\int_y^\infty e^{-\theta}\,\mathrm{d}\theta}{\int_y^\infty \frac{1}{\theta} e^{-\theta}\,\mathrm{d}\theta} > y = \widehat{\theta}_{MLE}.$$

■

Exercise 3.9 Consider the case $n > 1$ and we observe $\mathbf{Y} := (Y_1,\ldots,Y_n) = (y_1,\ldots,y_n) =: y$. As above assume that θ is exponential with parameter one. Compute the Bayesian estimate of θ given that $\mathbf{Y} = y$. ■

3.3 Exercises

Problem 3.1 Consider a medical test which that gives three possible results: 'p=positive', 'n=negative', and 'u=uncertain'. A randomly chosen person is tested. Let

A = the person is infected,
B_p = the test is positive
B_n = the test is negative
B_u = the test is uncertain.

We know that $\mathbb{P}(A) = 0.05$ and

$$\mathbb{P}(B_p\,|A) = 0.7,\ \ \mathbb{P}(B_n\,|A) = 0.1,\ \ \mathbb{P}(B_p\,|A^c) = 0.1,\ \ \mathbb{P}(B_n\,|A^c) = 0.6,$$

a. Compute $\mathbb{P}(B_u\,|A)$, and $\mathbb{P}(B_u\,|A^c)$.
b. Compute $\mathbb{P}(A\,|B_p)$.
c. Compute $\mathbb{P}(A\,|B_u)$.
d. Compute $\mathbb{P}(A\,|B_n)$.

Problem 3.2 In a game, we are told that only one of the three people Halil (first person), Mete (second), or Soner (third) has a reward and it will be given to us if we guess it correctly. All of them know who has the reward. When asked *"who has the reward?"*, they reply truthfully, if they do not have it. But when they have it, they randomly answer one of the other two. They chose the one with lower ranking with probability p and the other with probability $(1-p)$.

A_h = Halil has it,

A_m = Mete has it,
A_s = Soner has it.

Our prior is that all are equally likely, i.e., have probability $1/3$. We ask Mete and the answer is Soner. Let B be this event and let $p = 0.25$.

a. Compute $\mathbb{P}(B\,|\,A_h)$, $\mathbb{P}(B\,|\,A_m)$, and $\mathbb{P}(B\,|\,A_s)$.
b. Compute $\mathbb{P}(B)$.
c. Use Bayes' formula to compute all three posterior probabilities $\mathbb{P}(A_h)$, $\mathbb{P}(A_m)$, and $\mathbb{P}(A_s)$.

Problem 3.3 Consider a medical test which that gives two possible results: 'p=positive', and 'n=negative'. There are *three stages of this infection*: 's=illness (sick)', 'c=in incubation', and 'h=not infected (healthy)'. A randomly chosen person is tested. Let

A_s = the person is ill,
A_c = the infection is in incubation in this person,
A_h = the person is not infected,

B_p = the test is positive,
B_n = the test is negative.

We know that

$$\mathbb{P}(A_s) = 0.05, \qquad \mathbb{P}(A_c) = 0.05,$$

and

$$\mathbb{P}(B_p\,|\,A_s) = 0.96, \;\; \mathbb{P}(B_p\,|\,A_c) = 0.64, \;\; \mathbb{P}(B_p\,|\,A_h) = 0.1.$$

a. Compute $\mathbb{P}(A_h)$, $\mathbb{P}(B_p)$, and $\mathbb{P}(B_n)$.
b. Compute $\mathbb{P}(A_s\,|\,B_p)$ and $\mathbb{P}(A_c\,|\,B_n)$ (notice that the indices are different).

Problem 3.4 In the above problem, we change the formula for the answer to the question *"who has it?"*. In all below cases, compute the posterior probabilities.

a. Suppose that the answer is one of the other two with equal probability $p = 0.5$ no matter who actually has it.
b. The answer is always with probability $p = 0.25$ the smaller ranked person other than the one asked, and with probability $(1-p)$ the higher ranked person other than the one asked, regardless of who has the reward.
c. If the person has it, the answer is chosen equally likely from all three names including his. When he does not have it, he answers truthfully.

Problem 3.5 In Example 3.5, suppose that the prior distribution of P is $f_P(p) = 2\,p$ for $p \in (0,1)$.

a. Compute the posterior distribution of P.
b. Compute the Bayesian estimate for P for this prior.

Hint: The Beta distribution with parameters $a, b > 0$, i.e., the p.d.f. is given by,

$$f_P(p) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)}\; p^{a-1}\,(1-p)^{b-1}, \qquad p \in (0,1),$$

where Γ is the *gamma function*:

$$\Gamma(a) := \int_0^\infty u^{a-1}\,e^{-u}\,du.$$

Moreover, its *mean is equal to* $a/(a+b)$.

Problem 3.6 Suppose that $Y_1, \ldots, Y_n \in \{-1, +1\}$ are i.i.d. from the Bernoulli distribution with parameter $P \in (0,1)$, i.e., $\mathbb{P}(Y_1 = 1\,|\,P = p) = p$. The parameter P is not known and hence, is a

random variable. Suppose that the prior distribution of P is the beta distribution with parameters $a, b > 0$, Show that the posterior distribution of P again beta with parameters,

$$a + \sum_i x_i, \qquad b + n - \sum_i x_i.$$

Problem 3.7 Suppose the proportion θ of detective items in a large manufactured lot is unknown. When eight items were selected at random from the lot, and it is found that three are defective. In each of the following cases, compute the posterior distribution and the Bayesian estimate of θ (with quadratic loss).

a. Suppose that and the prior distribution of θ is uniform on the unit interval $[0,1]$.

b. Suppose that and the prior distribution of θ is

$$f_\theta(\theta) = 2(1-\theta), \qquad \theta \in [0,1].$$

c. What is the maximum likelihood estimator?

Problem 3.8 Let θ be the proportion of registered voters in a large city who are in favor of a certain proposition. Suppose that the value of θ is not known and has to be estimated. In a random sample of $1,00$ registered voters from the city, it is found that 710 are in favor of the proposition. In each of the following cases, compute the posterior distribution and the Bayesian estimate of θ.

a. Suppose that the prior is $f_\theta(\theta) = 2\,\theta$ for $\theta \in [0,1]$.

b. Suppose that the prior is $f_\theta(\theta) = 4\,\theta^3$ for $\theta \in [0,1]$.

c. What is the maximum likelihood estimator?

Problem 3.9 Let X be the *proportion* of registered voters in a large city who are in favor of a certain proposition. Suppose that the value of X is not known and has to be estimated. A random sample of 500 registered voters from the city were chosen. Let Y be the number of them who are in favor of the proposition. If we were to know the exact value $X = x$, then Y would be Binomial with success probability x. It is observed that $Y = 198$. Suppose that our prior of X is $f_X(x) = 5x^4$ for $x \in [0,1]$

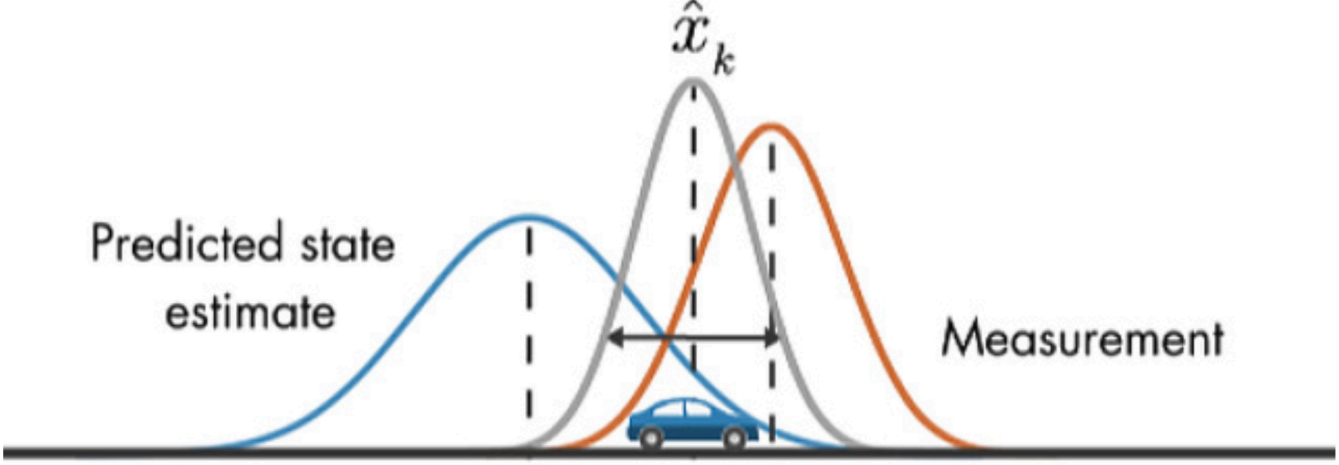
a. Compute the posterior distribution and the Bayesian estimate of X.

b. Compute $\mathbb{E}[X^2 \,|\, Y = 198]$.

Problem 3.10 Suppose that $Y_1, Y_2, \ldots, Y_n$ form a random sample from a distribution for which the probability density function is

$$f_{Y|\theta}(y\,|\,\theta) = \theta\, y^{\theta-1}, \qquad y \in (0,1).$$

Compute the maximum likelihood estimator of θ.



4. Kalman Filter

We first develop the Kalman filter and then use it to solve the Linear Gaussian Control (LGR) which is widely used in engineering applications. In particular, any auto-pilot contains a version of LGR.

Notation: We sometimes use the notation $Z \sim \mathcal{N}(\mu, Q)$ for a Gaussian random variable $Z \in \mathbb{R}^m$ with mean $\mu_Z \in \mathbb{R}^m$ and the $m \times m$ covariance matrix Q_Z. We recall that the probability density of X is given by,

$$f_Z(z) = (2\pi \det(Q_Z))^{-m/2} \exp(-\frac{1}{2}(z-\mu_z)^\top Q_Z^{-1}(z-\mu_Z)), \qquad z \in \mathbb{R}^m.$$

When X and Y are two Gaussian random variables then the conditional distribution of X given Y is also Gaussian, and its mean and covariance is derived in Theorem 3.1.1. Additionally, Exercise 3.8 studies the special case when X, Y are real-valued with correlation $\rho_{X,Y} \in [-1,1]$. Then, we have $\mathbb{C}(X,Y) = \mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] = \rho_{X,Y}\sigma_X\sigma_Y$, $\text{var}(X|Y) = \sigma_X^2(1-\rho_{X,Y}^2)$, and

$$\mathbb{E}[X|Y] - \mathbb{E}[X] = \frac{\mathbb{C}(X,Y)}{\sigma_Y^2}(Y-\mathbb{E}[Y]) = \frac{\rho_{X,Y}\sigma_X}{\sigma_Y}(Y-\mathbb{E}[Y]),$$

where $\sigma_X^2 = \text{var}(X)$, $\sigma_Y^2 = \text{var}(Y)$.

4.1 An Example: Simplified Position Estimator

We would like to estimate the position of a vehicle trying follow a given path. To simplify we assume that it moves on a straight line and has unit mass. Its movement is controlled by applying force to it. A target trajectory $p_d : \mathbb{R}_+ \mapsto \mathbb{R}$ is given and force is applied to the particle so as to keep its trajectory equal to the target. However, it is also subject to random force in addition to the computed one. We first obtain the *dynamics without noise*. Let $p_a(t)$ be the actual position of the particle. Without noise $p_a(t) = p_d(t)$. Then, the force is given by,

$$\text{Force} \;=\; \text{mass} \times \text{acceleration} = p_a'' p(t) \;=\; p_d''(t).$$

Therefore, *with noise*, the actual position satisfies $p_a''(t) \;=\; p_d''(t) + \text{noise}$.

Set the continuous-time state vector be $x(t) = [\,p_a(t) - p_d(t)\,,\, p_a'(t) - p_d'(t)\,)\,]^\top$. Then,

$$x'(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{noise}.$$

Fix a small time step h and set $x_k := x(ht)$ and ξ_{k+1} be the unknown noise that influenced the particle during time interval $[kh,(k+1)h]$. Then, the following difference equation is a good approximation and we use it as our state equation:

$$x_{k+1} = A x_k + \omega_{k+1}, \qquad \text{where} \qquad A = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \ \omega_{k+1} := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \xi_{k+1}.$$

The position also observed with noise v_k. So the observations are given by,

$$z_k = H x_k + v_k, \qquad \text{where} \qquad H = [1 \ \ 0].$$

In the above calculations, we fixed the force to be equal to $p_d''(t)$, as this is the optimal force in the deterministic problem. But with random perturbations we have apply additional force to keep the trajectory closer to the given one based on the noisy observations. This problem is the subject of Section 4.3 below. Here we first solve the problem of estimation and this solution is central to the solution given in Section 4.3.

4.2 Linear Gaussian (Kalman) Filter

We now describe the general problem. Suppose that $x_0, x_1, x_2, \ldots$ is a $\mathbb{R}^d$-valued stochastic process solving the random difference equation,

$$x_{k+1} = A_k x_k + \omega_{k+1}, \qquad k = 0,1,\ldots \tag{4.1}$$

where A_k is a known $d \times d$ matrix (not necessarily symmetric) and the noise process $\omega_k \in \mathbb{R}^d$ is an independent sequence of Gaussian random variables with zero mean and $\mathbb{C}(\omega_k) = Q_k$. The initial condition x_0 is *assumed* to be Gaussian with

$$\hat{x}_0 := \mathbb{E}[x_0], \qquad \text{and} \qquad P_0 := \mathbb{C}(x_0) = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^\top], \tag{4.2}$$

Moreover, we assume that the noise process is independent of x_0. We *do not observe* the process x directly but rather noisy observations $z_k \in \mathbb{R}^\ell$ are available:

$$z_k = H_k x_k + v_k, \qquad k = 1,2,\ldots \tag{4.3}$$

where $\ell \times d$ matrix H_k is known and the observation noise $v_k \in \mathbb{R}^\ell$ is another i.i.d. Gaussian sequence that is also independent of x_0, the noise ω_k and has mean zero with $\mathbb{C}(v_k) = R_k$.

In this framework, the goal is to find a relevant estimation for the state x_k at time step k, knowing the sequence of observations $z_1, \ldots, z_k$. In other words, the goal is to find a tractable formula for

$$\hat{x}_k := \mathbb{E}[x_k \,|\, z_1, \ldots, z_k], \qquad k = 1,2,..$$

In the following sections, we obtain a recursive formula starting from initial conditions $\widehat{x}_0$ and $P_0 = \mathbb{C}(x_0)$ which are assumed to be known. Towards this goal, for $k = 1, 2, \ldots$ we set

$$\begin{aligned}
&\widehat{x}_{k+1|k} := \mathbb{E}[x_{k+1}|z_1, \ldots, z_k], && \text{(pre-observation estimate)}\\
&\iota_k := z_k - H_k \widehat{x}_{k|k-1}, && \text{(innovation process)}\\
&P_k := \mathbb{C}(x_k - \widehat{x}_k) = \mathbb{E}[(x_k - \widehat{x}_k)(x_k - \widehat{x}_k)^\top], && \text{(error covariance)}\\
&\overline{P}_k := \mathbb{C}(x_k - \widehat{x}_{k|k-1}) = \mathbb{E}[(x_k - \widehat{x}_{k|k-1})(x_k - \widehat{x}_{k|k-1})^\top], && \text{(pre-observation error covariance.)}
\end{aligned}$$

By convention

$$\widehat{x}_{1|0} := \mathbb{E}[x_1] = A_0 \widehat{x}_0.$$

In words, $\widehat{x}_k$ is the estimate of x_k given all the observations $z_1, \ldots, z_k$. The process $\widehat{x}_{k|k-1}$ is the pre-observation estimate of x_k and ι_k called the *innovation process* (or *measurement residual*), is related to the information gained by the z_k compared to the pre-observation estimate $\widehat{x}_{k|k-1}$. It plays a fundamental role in computing the estimation of the state process x. By definition, *i.e.* $\iota_k := z_k - H_k \widehat{x}_{k|k-1}$, it can be computed after making the observation z_k, using the pre-observation estimate of x_k.

We first observe that each z_k is a linear function of $x_0, \omega_1, \ldots, \omega_k$ and ν_k, which are all independent Gaussian random variables. Therefore, it is a Gaussian process, and it is independent of all future noise processes ω_j, ν_j for $j > k$. In particular, we have:

$$\mathbb{E}[\omega_{k+1}|z_1, \ldots, z_k] = \mathbb{E}[\omega_{k+1}] = 0.$$

Therefore,

$$\widehat{x}_{k+1|k} = \mathbb{E}[x_{k+1}|z_1, \ldots, z_k] = \mathbb{E}[A_k x_k + \omega_{k+1}|z_1, \ldots, z_k] = A_k \mathbb{E}[x_k|z_1, \ldots, z_k].$$

Hence,

$$\boxed{\widehat{x}_{k+1|k} = A_k \widehat{x}_k, \qquad k = 0, 1, 2, \ldots.} \tag{4.4}$$

4.2.1 One-dimensional setting

Although all interesting applications are in multi-dimensions, to gain insight we start by solving the one-dimensional case. To motivate the problem, we start with an example.

■ **Example 4.1** Argor-Heraus company located in Mendrisio, Switzerland is a leading producer of gold bars. They are pressed in several standard weights. However, these are nominal or defined weights used for standardization and the actual weight varies as manufacturing processes result in small variations. In fact, it is neither possible nor often cost-effective to produce exact weights as per the product specifications. Additionally, gold bars are not 100% pure. Argor-Heraus reaches a value of 99.99% but not one. The actual fine weight is calculated by multiplying its gross weight by the reported purity. That means the true gross weight of a bar is random and is above the specified weight to make the actual fine weight close to the specification. One standard bar is 400 troy ounces which is equal to 12.3 kilos or roughly 27.4 pounds and was worth around $750,000 at the time o of the first version of these notes in 2021.

As an *academic exercise*, we would like to set the problem of estimating the weight as a Kalman filter. The production process ensures that the weight distribution is Gaussian with mean 400.004

troy pounds and standard deviation 0.004 troy pounds which is 0.001%. Suppose that one bar is randomly chosen and its weight is measured several times. The measurement has a measurement noise of 0.01 troy pounds which is about 0.0025%. We would like to set the problem of estimating the weight as a Kalman filter.

Since the true weight of the gold bar is not changing in time, we have $x_k = x_0$ for every $k = 0, 1, \ldots$. Therefore, $d = 1$, $A_k = 1$ and $\omega_k = 0$. Moreover, $\widehat{x}_0 = 400.004$ and $P_0 = 1.6 \times 10^{-5}$. The observations are true weight x_0 plus noise. This means H_k is equal to one. We take the covariance R_k of the measurement noise to be ν_k is 10^{-4}.

Summarizing, $x_0 \sim \mathcal{N}(400.004,\ 1.6 \times 10^{-5})$ and for $k = 1, 2 \ldots$

$$x_k = x_{k-1}, \quad z_k = x_k + \nu_k, \quad \nu_k \sim \mathcal{N}(0,\ 1.6 \times 10^{-3}).$$

In particular, $A_k = H_k = 1$, $\widehat{x}_0 = 400.004$, $P_0 = 1.6 \times 10^{-5}$, $Q_k = 0$, $R_k = 10^{-4}$.

Clearly, after n measurements, we could use the standard estimate of the simple average;

$$\overline{x}_n := \frac{1}{n} \sum_{k=1}^{n} z_k. \tag{4.5}$$

This is an *unbiased* estimate of the true weight with a standard deviation of $1/100\sqrt{n}$. The Kalman filter on the other hand uses the Gaussian prior to estimate the weight. See Example 4.2. ∎

In the one-dimensional setting $d = \ell = 1$, and we set $A_k = a$, $H_k = h$ and $Q_k = q$, $R_k = r$ with scalars $q, r > 0$. Hence,

$$x_{k+1} = a x_k + \omega_{k+1}, \qquad z_k = x_k + \nu_k, \qquad \mathbb{E}[\omega_k] = \mathbb{E}[\nu_k] = 0, \qquad \mathbb{E}[\omega_k^2] = q,\ \ \mathbb{E}[\nu_k^2] = r,$$

where $\{\omega_k\}$ and $\{\nu_k\}$ independent sequences. As in the general model, we denote by $\widehat{x}_k$ the estimate at step k, *i.e.*

$$\widehat{x}_k := \mathbb{E}[x_k | z_1, \ldots, z_k], \qquad k = 1, 2, \ldots,$$

and $\widehat{x}_{k|k-1}$ is the pre-observation estimate of x_k knowing $z_1, \ldots, z_{k-1}$. In dimension one, the covariances are scalars and we thus use the following notations for the covariance error and the pre-observation covariance error:

$$p_k = \mathbb{E}[(x_k - \widehat{x}_k)^2], \qquad \overline{p}_k = \mathbb{E}[(x_k - \widehat{x}_{k|k-1})^2], \qquad k = 1, 2, \ldots$$

We assume that $\widehat{x}_0$ and $p_0 = \mathrm{var}(x_0)$ are known. The goal is to determine a recursive formula for the estimate $\widehat{x}_k$ at each step k. To give an intuition, we develop the case $k = 1$ and then $k = 2$, before stating the general result in Theorem 4.2.1.

Recall first that the only observable variable at time k is the process $z_1, z_2, \ldots, z_k$. Nevertheless, by computing $\widehat{x}_{k+1|k}$ the pre-observation estimate of x_k knowing $z_1, \ldots, z_k$, we can also compute the innovation process, defined by:

$$\iota_{k+1} := z_{k+1} - h\widehat{x}_{k+1|k}.$$

Remark that for all k,

$$\begin{aligned}\mathbb{E}\big[\iota_{k+1}\big] &= \mathbb{E}\big[z_{k+1} - h\widehat{x}_{k+1|k}\big] = \mathbb{E}\big[hx_{k+1} + \nu_{k+1} - ah\widehat{x}_k\big] = \mathbb{E}\big[h(ax_k + \omega_{k+1}) + \nu_{k+1} - ah\widehat{x}_k\big] \\ &= ah\mathbb{E}\big[(x_k - \widehat{x}_k)\big] + h\mathbb{E}\big[\omega_{k+1}\big] + \mathbb{E}\big[\nu_{k+1}\big] = 0.\end{aligned}$$

Case $k = 1$.
Recall that we want to estimate x_1 using the observation z_1. More precisely, at step $k = 1$, our estimate is defined by $\widehat{x}_1 := \mathbb{E}[x_1|z_1]$. Remark first that the pair (x_1, z_1) is jointly Gaussian. As we have seen in the previous chapter (Theorem 3.1.1 and Exercise 3.8), since the pair (x_1, z_1) is jointly Gaussian, the conditional random variable $x_1|z_1$ is also Gaussian, with mean:

$$\mathbb{E}[x_1|z_1] = \mathbb{E}[x_1] + K_1(z_1 - \mathbb{E}[z_1]), \text{ where } K_1 := \frac{\mathbb{C}(x_1, z_1)}{\text{var}(z_1)}.$$

We first compute the following expectations:

$$\mathbb{E}[x_1] = \mathbb{E}[ax_0 + \omega_1] = a\widehat{x}_0 = \widehat{x}_{1|0}, \qquad \text{and} \qquad \mathbb{E}[z_1] = \mathbb{E}[hx_1 + \nu_1] = ah\widehat{x}_0 = h\widehat{x}_{1|0}.$$

We can also remark that:

$$z_1 - \mathbb{E}[z_1] = z_1 - h\widehat{x}_{1|0} = \imath_1.$$

We then need to compute the following variance and covariation:

$$\begin{aligned}\text{var}(z_1) &= \mathbb{E}\big[\big(z_1 - \mathbb{E}[z_1]\big)^2\big] = \mathbb{E}\big[\imath_1^2\big],\\ \mathbb{C}(x_1, z_1) &= \mathbb{E}\big[\big(x_1 - \mathbb{E}[x_1]\big)\big(z_1 - \mathbb{E}[z_1]\big)\big] = \mathbb{E}\big[\big(x_1 - \widehat{x}_{1|0}\big)\imath_1\big].\end{aligned}$$

Using the following formula for the innovation process:

$$\imath_1 := z_1 - h\widehat{x}_{1|0} = h(x_1 - \widehat{x}_{1|0}) + \nu_1,$$

and that $x_1 - \widehat{x}_{1|0}$ and ν_1 are independent, we obtain:

$$\begin{aligned}\text{var}(z_1) &= \mathbb{E}\big[\big(h(x_1 - \widehat{x}_{1|0}) + \nu_1\big)^2\big] = h^2\mathbb{E}\big[(x_1 - \widehat{x}_{1|0})^2\big] + \mathbb{E}\big[\nu_1^2\big] = h^2\overline{p}_1 + r,\\ \mathbb{C}(x_1, z_1) &= \mathbb{E}\big[(x_1 - \widehat{x}_{1|0})(h(x_1 - \widehat{x}_{1|0}) + \nu_1)\big] = h\mathbb{E}\big[(x_1 - \widehat{x}_{1|0})^2\big] + \mathbb{E}\big[(x_1 - \widehat{x}_{1|0})\nu_1\big] = h\overline{p}_1.\end{aligned}$$

To conclude, we finally obtain:

$$\widehat{x}_1 := \mathbb{E}[x_1|z_1] = a\widehat{x}_0 + K_1\imath_1, \text{ where } K_1 := \frac{h\overline{p}_1}{h^2\overline{p}_1 + r}. \tag{4.6}$$

Therefore, the estimation process is the following: at step $k = 0$, we know $\widehat{x}_0 := \mathbb{E}[x_0]$ and we can thus compute the pre-observation estimate $\widehat{x}_{1|0} = a\widehat{x}_0$. Then, by observing z_1, we can compute the innovation process $\imath_1 = z_1 - h\widehat{x}_{1|0}$. Since we know r, the estimation problem is solved once the pre-observation covariance error $\overline{p}_1$ is computed. We leave this computation to later and proceed to step $k = 2$

Case $k = 2$.
We first compute the innovation process, using the previous estimation (4.6) for $\widehat{x}_1$:

$$\begin{aligned}\imath_2 &:= z_2 - h\widehat{x}_{2|1} = z_2 - ah\widehat{x}_1 = z_2 - ah\big(a\widehat{x}_0 + K_1\imath_1\big) = z_2 - a^2h\widehat{x}_0 - ahK_1\imath_1\\ &= z_2 - a^2h\widehat{x}_0 - ahK_1\big(z_1 - ah\widehat{x}_0\big) = z_2 - ahK_1z_1 + a^2h(hK_1 - 1)\widehat{x}_0.\end{aligned}$$

Hence,

$$\begin{pmatrix}\imath_1\\ \imath_2\end{pmatrix} = \begin{pmatrix}1 & 0\\ -ahK_1 & 1\end{pmatrix}\begin{pmatrix}z_1\\ z_2\end{pmatrix} + \begin{pmatrix}-ah\\ a^2h(hK_1 - 1)\end{pmatrix}\widehat{x}_0.$$

In particular, the map (z_1, z_2) to (ι_1, ι_2) is invertible. Therefore, conditioning on (z_1, z_2) is the same as conditioning on (ι_1, ι_2). Note that this is also true in the general case. Also, $\widehat{x}_{2|1} = \mathbb{E}[x_2|z_1] = \mathbb{E}[x_2|\iota_1]$. We thus obtain:

$$\begin{aligned}\widehat{x}_2 &:= \mathbb{E}[x_2|z_1, z_2] = \mathbb{E}[x_2|\iota_1, \iota_2] = \mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_1, \iota_2] + \mathbb{E}[\widehat{x}_{2|1}|\iota_1, \iota_2] \\ &= \mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_1, \iota_2] + \widehat{x}_{2|1}.\end{aligned} \tag{4.7}$$

Moreover,

$$\begin{aligned}\mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_1] &= \mathbb{E}[x_2 - a\widehat{x}_1|z_1] = \mathbb{E}[x_2 - a\mathbb{E}[x_1|z_1]|z_1] = \mathbb{E}[ax_1 + \omega_2|z_1] - a\mathbb{E}[x_1|z_1] \\ &= \mathbb{E}[\omega_2|z_1] = \mathbb{E}[\omega_2] = 0.\end{aligned}$$

Since the pair $(x_2 - \widehat{x}_{2|1}, \iota_1)$ is jointly Gaussian with zero mean, the previous result implies that they are independent of each other. Hence,

$$\mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_1, \iota_2] = \mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_2].$$

On the other hand, the pair $(x_2 - a\widehat{x}_1, \iota_2)$ is jointly Gaussian, implying that the conditional random variable $x_2 - a\widehat{x}_1|\iota_2$ is also Gaussian with mean:

$$\mathbb{E}[x_2 - \widehat{x}_{2|1}|\iota_2] = \mathbb{E}[x_2 - \widehat{x}_{2|1}] + K_2\big(\iota_2 - \mathbb{E}[\iota_2]\big), \text{ where } K_2 := \frac{\mathbb{C}(x_2 - \widehat{x}_{2|1}, \iota_2)}{\mathrm{var}(\iota_2)}.$$

We first have $\mathbb{E}[x_2 - \widehat{x}_{2|1}] = 0$ and $\mathbb{E}[\iota_2] = 0$. As before, using the equation

$$\iota_2 = z_2 - h\widehat{x}_{2|1} = hx_2 + \nu_2 - h\widehat{x}_{2|1}$$

we can then compute:

$$\begin{aligned}\mathbb{E}[(x_2 - \widehat{x}_{2|1})\iota_2] &= \mathbb{E}[(x_2 - \widehat{x}_{2|1})(hx_2 + \nu_2 - h\widehat{x}_{2|1})] = h\mathbb{E}\big[(x_2 - \widehat{x}_{2|1})^2\big] + \mathbb{E}[(x_2 - \widehat{x}_{2|1})\nu_2] \\ &= h\overline{p}_2 + \mathbb{E}[x_2 - \widehat{x}_{2|1}]\mathbb{E}[\nu_2] = h\overline{p}_2, \\ \mathbb{E}\big[\iota_2^2\big] &= \mathbb{E}\big[\big(hx_2 + \nu_2 - h\widehat{x}_{2|1}\big)^2\big] = h^2\mathbb{E}\big[\big(x_2 - \widehat{x}_{2|1}\big)^2\big] + \mathbb{E}\big[\nu_2^2\big] \\ &= h^2\overline{p}_2 + r.\end{aligned}$$

Using (4.7), we finally conclude:

$$\widehat{x}_2 = a\widehat{x}_1 + \mathbb{E}[x_2 - a\widehat{x}_1|\iota_2] = a\widehat{x}_1 + K_2\iota_2, \text{ where } K_2 = \frac{h\overline{p}_2}{h^2\overline{p}_2 + r}.$$

This is exactly the same formula as the one derived for $k = 1$. The general formula is given by the following theorem.

Theorem 4.2.1 For each $k = 1, 2, \ldots$, $\widehat{x}_{k+1|k} = a\widehat{x}_k$ and

$$\widehat{x}_k = a\widehat{x}_{k-1} + K_k\iota_k, \quad \text{where} \quad K_k = \frac{h\overline{p}_k}{h^2\overline{p}_k + r}.$$

Proof. The proof of the previous result can be done by mathematical induction. We have already proven that the result holds for $k = 1$ (even for $k = 2$). We now assume that it holds for some $k \in \{1, 2, \ldots\}$, and prove that it is still true for $k + 1$.

We will prove in the next subsection that $(x_{k+1} - \widehat{x}_{k+1|k})$ is actually independent of $\iota_1, \ldots, \iota_k$ and that conditioning on $z_1, \ldots, z_k$ is the same as conditioning on $\iota_1, \ldots, \iota_k$ (this has already been proved above for $k = 1, 2$). Using this, we have:

$$\begin{aligned}\widehat{x}_{k+1} &= \mathbb{E}[x_{k+1}|z_1, \ldots, z_{k+1}] = \widehat{x}_{k+1|k} + \mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}|z_1, \ldots, z_{k+1}] \\ &= \widehat{x}_{k+1|k} + \mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}|\iota_1, \ldots, \iota_{k+1}] = \widehat{x}_{k+1|k} + \mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}|\iota_{k+1}].\end{aligned}$$

Then, we know that the pair $(x_{k+1} - \widehat{x}_{k+1|k}, \iota_{k+1})$ is jointly Gaussian, which implies that the conditional random variable $x_{k+1} - \widehat{x}_{k+1|k}|\iota_{k+1}$ is jointly Gaussian, with mean:

$$\mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}|\iota_{k+1}] = \mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}] + K_{k+1}\big(\iota_{k+1} - \mathbb{E}[\iota_{k+1}]\big),$$

where

$$K_{k+1} = \frac{\mathbb{E}[(x_{k+1} - \widehat{x}_{k+1|k})\,\iota_{k+1}]}{\mathbb{E}[\iota_{k+1}^2]}.$$

As before, we first compute the expectations:

$$\mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}] = 0, \text{ and } \mathbb{E}[\iota_{k+1}] = 0.$$

Then, by definition of the innovation process, we have $\iota_{k+1} = z_{k+1} - h\widehat{x}_{k+1|k} = h(x_{k+1} - \widehat{x}_{k+1|k}) + \nu_{k+1}$. Therefore,

$$\begin{aligned}\mathbb{E}[(x_{k+1} - \widehat{x}_{k+1|k})\iota_{k+1}] &= h\mathbb{E}[(x_{k+1} - \widehat{x}_{k+1|k})^2] = h\overline{p}_{k+1} \\ \text{and } \mathbb{E}[\iota_{k+1}^2] &= h^2\mathbb{E}[(x_{k+1} - \widehat{x}_{k+1|k})^2] + \mathbb{E}[\nu_{k+1}^2] = h^2\overline{p}_{k+1} + r\end{aligned}$$

We conclude that

$$\begin{aligned}\widehat{x}_{k+1} &= \widehat{x}_{k+1|k} + \mathbb{E}[x_{k+1} - \widehat{x}_{k+1|k}|\iota_{k+1}] \\ &= a\widehat{x}_k + K_{k+1}\iota_{k+1},\end{aligned}$$

for the appropriate K_{k+1}, which implies that the claimed formula also holds for $k+1$. We conclude by mathematical induction that it holds for all $k \geqslant 1$. ■

To complete the calculation of $\widehat{x}_k$ we need to derive a recursive formula for $\overline{p}_k$.

Computing the variances p_k, $\overline{p}_k$ and then the gains K_k.
We first note that for any $k = 0, 1, \ldots$, $(x_{k+1} - \widehat{x}_{k+1|k})$ and $(x_k - \widehat{x}_k)$ have mean zero. Also, by definition, $x_{k+1} - \widehat{x}_{k+1|k} = a(x_k - \widehat{x}_k) + \omega_{k+1}$, and $(x_k - \widehat{x}_k)$ is independent of ω_{k+1}. Hence,

$$\overline{p}_{k+1} = \mathbb{E}[(x_{k+1} - \widehat{x}_{k+1|k})^2] = a^2\mathbb{E}[(x_k - \widehat{x}_k)^2] + \mathbb{E}[\omega_{k+1}^2] = a^2 p_k + q, \qquad k = 0, 1, \ldots.$$

We continue by computing p_k. We start with

$$\begin{aligned}x_k - \widehat{x}_k &= x_k - (\widehat{x}_{k|k-1} + K_k\iota_k) \;= (x_k - \widehat{x}_{k|k-1}) - K_k(z_k - h\widehat{x}_{k|k-1}) \\ &= (x_k - \widehat{x}_{k|k-1}) - K_k(hx_k - h\widehat{x}_{k|k-1}) - K_k\nu_k = (1 - hK_k)(x_k - \widehat{x}_{k|k-1}) - K_k\nu_k.\end{aligned}$$

As $(x_k - \widehat{x}_{k|k-1})$ and v_k are independent, we have

$$\begin{aligned}
p_k &:= \mathbb{E}[(x_k - \widehat{x}_k)^2] = \mathbb{E}\Big[\big((1-hK_k)(x_k - \widehat{x}_{k|k-1}) - K_k v_k\big)^2\Big] \\
&= (1-hK_k)^2 \mathbb{E}\Big[\big(x_k - \widehat{x}_{k|k-1}\big)^2\Big] + K_k^2 \mathbb{E}\big[v_k^2\big] = (1-hK_k)^2 \overline{p}_k + K_k^2 r \\
&= \left(1 - h\frac{h\overline{p}_k}{h^2\overline{p}_k + r}\right)^2 \overline{p}_k + \left(\frac{h\overline{p}_k}{h^2\overline{p}_k + r}\right)^2 r \\
&= \frac{r^2}{(h^2\overline{p}_k + r)^2}\overline{p}_k + \frac{rh^2\overline{p}_k^2}{(h^2\overline{p}_k + r)^2} = \frac{\overline{p}_k r^2 + rh^2\overline{p}_k^2}{(h^2\overline{p}_k + r)^2} \\
&= \frac{r\overline{p}_k}{h^2\overline{p}_k + r}.
\end{aligned}$$

Summary of the results.
For $k = 1, 2, \ldots$, the system dynamics is given by $x_k = ax_{k-1} + \omega_k$, with x_0 Gaussian with mean $\widehat{x}_0$ and variance p_0. The Gaussian random variable ω_k are i.i.d. with mean zero and variance q.

We observe a noisy version of the state of the system, *i.e.* $z_k = hx_k + v_k$, with i.i.d. Gaussian random variable v_k, with mean zero and variance r.

Starting with given values $\widehat{x}_0$ and p_0, at each step k with $k \geqslant 1$, using $\widehat{x}_{k-1}$, p_{k-1} computed at step $k-1$ and the new observation z_k, we recursively compute:

$$\begin{aligned}
\widehat{x}_{k|k-1} &= a\widehat{x}_{k-1}, \\
\iota_k &= z_k - h\widehat{x}_{k|k-1}, \\
\overline{p}_k &= \operatorname{var}(x_k - \widehat{x}_{k|k-1}) = a^2 p_{k-1} + q, \\
p_k &= \operatorname{var}(x_k - \widehat{x}_k) = \frac{r\overline{p}_k}{h^2\overline{p}_k + r} \\
\widehat{x}_k &= \widehat{x}_{k|k-1} + K_k \iota_k \quad \text{where} \quad K_k = \frac{h\overline{p}_k}{h^2\overline{p}_k + r}.
\end{aligned}$$

The variances $p_k, \overline{p}_k$ can be calculated independent of the observations directly using the known system dynamics. The gains factor can also be computed as it is a simple functions the covariances and it is related to the strength of the observation noise variance r and the variance $\overline{p}_k$ of the pre-observation error. So starting with p_0, the variances p_k and $\overline{p}_k$, as well as the resulting gains K_k, can be calculated prior to the observations. The pre-observation estimate $\widehat{x}_{k|k-1}$ is then computed recursively using the system dynamics and the prior estimate $\widehat{x}_{k-1}$. After the observation z_k is made, the estimate $\widehat{x}_k$ is updated by using the innovation process which "records" the additional information obtained by the new observation and the gains factor.

■ **Example 4.2** In the Example 4.1, the stochastic process x we would like to estimate is the weight of the considered gold bar. Since the true weight is not changing over time, we first have $x_k = x_0$ for every $k = 0, 1, \ldots$. In other words, using the notations introduced before, we have $d = 1$, $a = h = 1$ and $\omega_k = 0$. Moreover, we know that the production process ensures that the weight distribution is Gaussian with mean 400.004 and standard deviation 0.004. Therefore, $\widehat{x}_0 = 400.004$ and

$$p_0 := \mathbb{C}(x_0) = \operatorname{var}(x_0) = 0.004^2 = 1.6 \times 10^{-5}.$$

The observation process z corresponds to the different measurements, *i.e.* the true weight x_0 plus some noise. In other words, the matrix h is equal to one, and we take the covariance r of the measurement noises v to be equal to 10^{-4}.

Summary of the problem's formulation.
The initial condition is $x_0 \sim \mathcal{N}(400.004, 1.6 \times 10^{-5})$ and for $k = 1, 2 \ldots$, the state dynamics and observed process are respectively given by:

$$x_k = x_{k-1}, \quad z_k = x_k + v_k, \quad v_k \sim \mathcal{N}(0, 10^{-4}).$$

In particular, using the previous notations, $a = h = 1$, $\widehat{x}_0 = 400.004$, $p_0 = 1.6 \times 10^{-5}$, $q = 0$, and $r = 10^{-4}$.

Clearly, after n measurements, we could use the standard estimate of the simple average, *i.e.*

$$\overline{x}_n := \frac{1}{n} \sum_{k=1}^{n} z_k.$$

This is an *unbiased* estimate of the true weight.

On the other hand, the Kalman filter uses the Gaussian prior to estimate the weight. Using the previous results, we can first compute the variances:

$$p_{k-1} = \mathrm{var}(x_{k-1} - \widehat{x}_{k-1}) = \frac{r\overline{p}_{k-1}}{h^2\overline{p}_{k-1} + r} = \frac{r\overline{p}_{k-1}}{\overline{p}_{k-1} + r}$$
$$\text{and } \overline{p}_k = \mathrm{var}(x_k - \widehat{x}_{k|k-1}) = a^2 p_{k-1} + q = p_{k-1}.$$

Therefore, the variances p_k solve the following recursion starting with $p_0 = 1.6 \times 10^{-5}$:

$$p_k = \frac{r\overline{p}_k}{\overline{p}_k + r} = \frac{rp_{k-1}}{p_{k-1} + r}, \qquad k = 1, 2, \ldots$$

Let $\alpha = p_0/r = 0.16$. Then we have

$$p_0 = r\alpha, \qquad p_1 = \frac{rp_0}{p_0 + r} = \frac{r\alpha}{1 + \alpha}, \qquad p_2 = \frac{rp_1}{p_1 + r} = \frac{r^2\alpha}{r\alpha + r(1 + \alpha)} = \frac{r\alpha}{1 + 2\alpha}, \ \ldots$$

Actually, by mathematical induction, one can prove the general formula:

$$p_k = \frac{r\alpha}{1 + k\alpha}, \qquad k = 1, 2, \ldots$$

Once we have computed the sequence of p_k, we can use the fact that $\overline{p}_k = p_{k-1}$ to compute the sequence K_k:

$$K_k = \frac{h\overline{p}_k}{h^2\overline{p}_k + r} = \frac{p_{k-1}}{p_{k-1} + r} = \frac{\alpha}{1 + k\alpha}.$$

We can finally compute the estimation, using the equalities $\widehat{x}_{k|k-1} = \widehat{x}_{k-1}$ and $\iota_k = z_k - \widehat{x}_{k|k-1}$:

$$\widehat{x}_k = \widehat{x}_{k|k-1} + K_k \iota_k = \widehat{x}_{k-1} + \frac{\alpha}{1 + k\alpha}(z_k - \widehat{x}_{k-1}).$$

The solution is

$$\widehat{x}_0 = 400.004,$$
$$\widehat{x}_1 = 400.004 + \frac{0.16}{1.16}(z_1 - 400.004),$$
$$\widehat{x}_2 = 400.004 + \frac{0.16}{1.32}(z_2 - 400.004) + \frac{0.16}{1.32} \times \frac{0.16}{1.16}(z_1 - 400.004), \ \ldots$$

One can easily compute this sequence numerically. The associated Python file can be found in Canvas. Below is the output of the python code:

```
Number of observations is 7

Observations are
[400.004 400.004 400.004 400.004 400.004 400.004 400.004]

Kalman estimate is
400.003987

100 times the Kalman standard deviation is
0.274721128

Number of observations is 1000000

Observations are
[400.004 400.004 400.004 ... 400.004 400.004 400.004]

Kalman estimate is
400.004

100 times the Kalman standard deviation is
0.000999997
```

■

4.2.2 Innovation Process

In this subsection, we extend the properties of the the innovation process to the general case. Let the dynamics be as in (4.1), (4.2), (4.3). Recall that the *innovation process* is given by,

$$\iota_k := z_k - H_k \hat{x}_{k|k-1}, \qquad \forall\, k = 1,2,\dots$$

By the definition of z_k, $\iota_k := H_k \left[x_k - \hat{x}_{k|k-1}\right] + v_k$. Moreover, $x_k - \hat{x}_{k|k-1}$ is a linear function of x_0, $\omega_1,\dots,\omega_k$ and $v_0, v-1,\dots,v_{k-1}$. Hence, $x_k - \hat{x}_{k|k-1}$ is Gaussian with mean zero and is independent of v_k. Hence,

$$\mathbb{E}[\iota_k] = 0, \qquad \mathbb{C}(\iota_k) = \mathbb{E}[\iota_k \iota_k^\top] = H_k \overline{P}_k H_k^\top + R_k, \qquad \forall\, k = 1,2,\dots \tag{4.8}$$

Following result collects other important properties of the innovation process.

Theorem 4.2.2 — Innovation process. The innovation process has the following properties:

1. $\{\iota_k\}_{k=1,2,\dots}$ are independent of each other and have mean zero.
2. For each $k \geqslant 1$, $(x_{k+1} - \hat{x}_{k+1|k})$ is independent of all $\iota_1,\dots,\iota_k$.
3. $\iota_1,\dots,\iota_k$ is an affine function of $z_1,\dots,z_k$. Hence, it is Gaussian and conditioning on $z_1,\dots,z_k$ and $\iota_1,\dots,\iota_k$ are same.

Proof. To prove the first point, let $k \geqslant 1$. By the definition of ι_{k+1},

$$\begin{aligned}
\mathbb{E}[\iota_{k+1}|z_1,\dots,z_k] &= \mathbb{E}[z_{k+1}|z_1,\dots,z_k] - H_{k+1}\,\mathbb{E}[\hat{x}_{k+1|k}|z_1,\dots,z_k] \\
&= \mathbb{E}[H_{k+1}x_{k+1} + v_{k+1}|z_1,\dots,z_k] - H_{k+1}\hat{x}_{k+1|k}, && \text{(definition of } \hat{x}_{k+1|k}) \\
&= H_{k+1}\,\mathbb{E}[x_{k+1}|z_1,\dots,z_k] - H_{k+1}\hat{x}_{k+1|k}, && (v_{k+1} \text{ is ind. of } z_0,\dots,z_k) \\
&= 0, && \text{(definition of } \hat{x}_{k+1|k}).
\end{aligned}$$

Therefore, ι_{k+1} and $\{z_1,\dots,z_k\}$ are uncorrelated. As they are also jointly Gaussian, this implies that they are independent. Since by their definitions $\{\iota_1,\dots,\iota_k\}$ are functions of $z_1,\dots,z_k$, the next innovation ι_{k+1} is also independent of $\{\iota_1,\dots,\iota_k\}$. The fact that the expectation is zero is proved above in (4.8).

To prove the second property, first observe that

$$\begin{aligned}\mathbb{E}[x_{k+1}-\widehat{x}_{k+1|k}|\iota_1,\dots,\iota_k] &= \mathbb{E}[x_{k+1}-\widehat{x}_{k+1|k}|z_1,\dots,z_k]\\ &= \mathbb{E}\left(x_{k+1}-\mathbb{E}[\widehat{x}_{k+1|k}|z_1,\dots,z_k]|z_1,\dots,z_k\right)=0.\end{aligned}$$

Since all are Gaussian, this implies that $(x_{k+1}-\widehat{x}_{k+1|k})$ is independent of $\iota_1,\dots,\iota_k$.

The last statement follows from a direct calculation as demonstrated in the previous subsection for the one-dimensional example and we skip it. ■

4.2.3 General case

We only state the results. The system dynamics is (4.1), the initial data x_0 is assumed to satisfy (4.2) and the observations are as in (4.3). Recalling them briefly

$$\begin{aligned}&x_{k+1} = A_k x_k + \omega_k,\\ &z_k = H_k x_k + \nu_k\\ &\widehat{x}_k := \mathbb{E}[x_k\,|\,z_1,\dots,z_k\,],\\ &\widehat{x}_{k+1|k} := \mathbb{E}[x_{k+1}\,|\,z_1,\dots,z_k\,] = A_k\widehat{x}_k,\\ &\iota_{k+1} := z_{k+1}-H_{k+1}\widehat{x}_{k+1|k} = z_{k+1}-H_{k+1}A_k\widehat{x}_k,\\ &P_k := \mathbb{C}(x_k-\widehat{x}_k) = \mathbb{E}[(x_k-\widehat{x}_k)(x_k-\widehat{x}_k)^\top],\\ &\overline{P}_k := \mathbb{C}(x_k-\widehat{x}_{k|k-1}) = \mathbb{E}[(x_k-\widehat{x}_{k|k-1})(x_k-\widehat{x}_{k|k-1})^\top].\end{aligned}$$

We follow the same computations given in the one-dimensional case. The result is the following set of computational formulae.. We do not provide the derivation in these notes.

Theorem 4.2.3 Starting from P_0, for $k=1,2,\dots$, $\overline{P}_k$ and P_k are computed recursively by,

$$\overline{P}_k = A_{k-1}P_{k-1}A_{k-1}^\top + Q_k,\quad K_k = \overline{P}_k H_k^\top\left(H_k\overline{P}_k H_k^\top + R_k\right)^{-1},\quad P_k = (I-K_kH_k)\overline{P}_k,$$

Given a sequence of observations, starting with $\widehat{x}_0$, and we recursively compute,

$$\widehat{x}_{k|k-1} = A_{k-1}\widehat{x}_{k-1},\quad \iota_k = z_k - H_k\widehat{x}_{k|k-1},\quad \widehat{x}_k = \widehat{x}_{k|k-1}+K_k\iota_k.$$

We rewrite the equation for the state estimate as follows

$$\widehat{x}_{k+1} = A_k\widehat{x}_k + \widehat{\omega}_{k+1},\quad \text{where}\quad \widehat{\omega}_{k+1} = K_{k+1}\iota_{k+1}.$$

Moreover, in view of the covariance formula (4.8), we have

$$\mathbb{C}(\widehat{\omega}_k) = K_k\,\mathbb{C}(\iota_k)\,K_k^\top = K_k\left(H_k\overline{P}_kH_k^\top+R_k\right)K_k^\top,\qquad \forall\, k=1,2,\dots \tag{4.9}$$

Exercise 4.1 — Kalman Riccati Equations. Show that $\overline{P}_k$ solves,

$$\overline{P}_{k+1} = Q_{k+1} + A_k \overline{P}_k A_k^\top - A_k \overline{P}_k H_k^\top \left(H_k \overline{P}_k H_k^\top + R_k \right)^{-1} H_k \overline{P}_k A_k^\top .$$

Note that the above equation is exactly the Riccati equation (2.7) studied in Chapter 2, with $\rho = 1$, and (B, M, N) replaced with (H_k, Q_{k+1}, R_k). ∎

R In the next chapter, we will use the above Kalman filter for a controlled process. As in Section 2.4, assume that the dynamics is given as

$$x_{k+1} = A_k\, x_k + B_k\, u_k + \omega_{k+1}, \qquad k = 0, 1, \ldots$$

However, as opposed to Section 2.4, we now assume that the control u_k *uses only the information given by the observations* and not do not have access to the state process x_k. Then, the presence of this additional "blue" term does not change the structure of the filter. Indeed, the equations for $\widehat{x}_k$, $\imath_k$, $K_k, P_k, \overline{P}_k$ are exactly as before. For future reference we note that

$$\widehat{x}_k = \widehat{x}_{k|k-1} + K_k\, \imath_k, \quad \text{and} \quad \widehat{x}_{k+1|k} = A_k \widehat{x}_k + B_k\, u_k, \qquad k = 1, 2, \ldots \tag{4.10}$$

Suppose that there no observations and we estimate x_k using only our knowledge of the system dynamics. In this case, $H_k = 0$ for each k, yielding $K_k = 0$ and $\overline{P}_k = P_k$ (which makes sense as there is no difference between pre and after observation estimate as there are no observations). The covariance matrix solves

$$P_{k+1} = Q_{k+1} + A_k P_k A_k^\top .$$

Unless the dynamical system (4.1) is stable the covariances grow exponentially over time.
In the other extreme of full observation, we have $H_k = I$ and $R_k = 0$. Then, $K_k = I$ and $P_k = 0$.

■ **Example 4.3** We now consider the example studied in Section 4.1. In that case, $d = 2$, $\ell = 1$ and

$$A = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \quad H = [1\ \ 0], \quad \omega_{k+1} := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \xi_{k+1}, \quad \nu_k \in \mathbb{R}.$$

Hence, $Q = \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$, $R = [r]$. We solve this system numerically for the values $h = 0.01$, $r = 0.1$, $q = 10h$, and

$$P_0 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The resulting covariance matrices are given below. the full, python code is given at the end of the chapter.

```
This is the error covariance

[[0.01319277 0.0931704 ]
 [0.0931704  1.41598243]]

This is the pre-error covariance

[[0.01519777 0.10733022]
 [0.10733022 1.51598243]]

This is the gains matrix

[[0.13192765]
 [0.931704  ]]
```

We also simulate the system and the estimate it up to $k = 1{,}000$. The result for the position coordinate is given in the below figures. The first one is the actual position and its estimate. The second figure show the error between the actual position and its estimate.

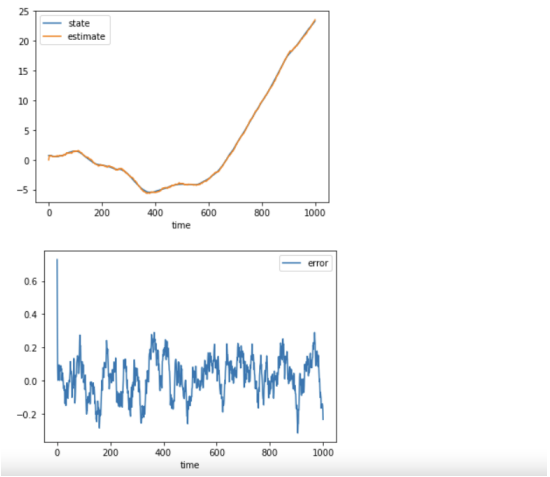


Figure 4.1: Position

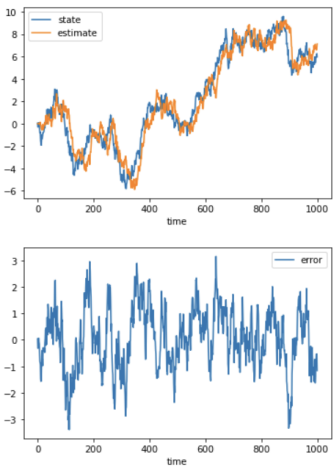


Figure 4.2: Velocity

■

4.2.4 Long time behavior

Suppose that all coefficients are time independent, i.e., $A = A_k$, $H = H_k$, $Q = Q_k$, $R = R_k$. Further assume that

$$P_* := \lim_{k\to\infty} P_k,$$

exists. We do not discuss the condition under which this happens. Under these assumptions, we have

$$\begin{aligned}
\overline{P}_* &:= \lim_{k\to\infty} \overline{P}_k = A P_* A^\top + Q,\\
K_* &:= \lim_{k\to\infty} K_k = \overline{P}_* H^\top (H \overline{P}_* H^\top + R)^{-1},\\
P_* &= (I - K_* H)\overline{P}_*.
\end{aligned}$$

Combining all we obtain then following equation for P_*,

$$P_* = Q + A P_* A^\top - \overline{P}_* H^\top (H \overline{P}_* H^\top + R)^{-1} (A P_* A^\top + Q).$$

Additionally, $\overline{P}_*$ satisfies,

$$\overline{P}_* = Q + A \overline{P}_* A^\top - A \overline{P}_* H^\top (H \overline{P}_* H^\top + R)^{-1} H \overline{P}_* A^\top.$$

Suppose that we initially we start with $P_0 = P_*$, P_1 is given by,

$$P_1 = Q + A P_* A^\top - P_* H^\top (H P_* H^\top + R)^{-1} (A P_* A^\top + Q).$$

By the equation above we conclude that $P_1 = P_*$ and therefore $P_k = P_*$ for every k. Similarly, we conclude that $\overline{P}_k = \overline{P}_*$ and $K_k = K_*$ for all k. Moreover,

$$\hat{x}_{k+1} = A \hat{x}_k + K_* \imath_k.$$

We know that the innovation process has mean zero, independent of each other and

$$\mathbb{C}(\iota_k) = H\overline{P}_k H^\top + R = H\overline{P}_* H^\top + R.$$

Hence, the innovation process is also identical making it an i.i.d. process.

We summarize all of the above into the following result.

Theorem 4.2.4 The matrices $(P_*, \overline{P}_*, K_*) := \lim_{k\to\infty}(P_k, \overline{P}_k, K_k)$ solve

$$\begin{aligned}
\overline{P}_* &= Q + A\overline{P}_* A^\top - A\overline{P}_* H^\top (H\overline{P}_* H^\top + R)^{-1} H\overline{P}_* A^\top \\
K_* &= \overline{P}_* H^\top (H\overline{P}_* H^\top + R)^{-1} \\
P_* &= (I_d - K_* H)\overline{P}_*.
\end{aligned}$$

Moreover, if $P_0 = P_*$, then the Kalman covariances $(P_k, \overline{P}_k, K_k)$ are all equal to $(P_*, \overline{P}_*, K_*)$. Additionally, the innovation process is i.i.d. with mean zero and

$$\mathbb{C}(\iota_k) = H\overline{P}_* H^\top + R.$$

4.3 Linear (Quadratic) Gaussian Regulator

We continue by solving the widely applicable extension of the linear quadratic regulator LQR.

4.3.1 General framework

The dynamics of this problem is as in Section 2.4 (see (2.10)). However, as opposed to the standard LQR, the decision maker do not have access the state but has observations related to the state as in the previous section, see (4.3). Thus, the actions/control are required to be functions of only the observations up to the current time. Therefore, the optimisation problem is to minimise the following cost with finite horizon:

$$J_n(x, \mathbf{u}) := \sum_{k=0}^{n-1} \mathbb{E}\left[x_k^\top M x_k + u_k^\top N u_k\right] + \mathbb{E}[x_n^\top \widehat{M} x_n], \tag{4.11}$$

or infinite horizon with discount $\rho < 1$:

$$J_\infty(x, \mathbf{u}) := \sum_{k=0}^{\infty} \rho^k \mathbb{E}\left[x_k^\top M x_k + u_k^\top N u_k\right] \tag{4.12}$$

while the system dynamics are given by:

$$x_0 = x, \quad \text{and} \quad x_{k+1} = A x_k + B u_k + \omega_{k+1}, \qquad k = 0, 1, \ldots$$

The main difference with the framework studied in Section 2.4 is that here, the control u_k is a function of $z_1, \ldots, z_k$, where

$$z_k = H x_k + \nu_k, \qquad k = 0, 1, \ldots,$$

and the random sequences ω_k and ν_k are as in the previous section, *i.e.* i.i.d Gaussians with mean and $\mathbb{C}(\omega_k) = Q_k$, $\mathbb{C}(\nu_k) = R_k$.

4.3.2 Solution via Riccati Equations : Finite Horizon

Using the Kalman equation (4.10), we conclude that

$$\widehat{x}_{k+1} = A\widehat{x}_k + Bu_k + \widehat{\omega}_{k+1}, \qquad \text{where} \qquad \widehat{\omega}_{k+1} = K_{k+1}\iota_{k+1}, \qquad k = 0,1,\ldots$$

We have shown in Theorem 4.2.2 that the sequence $\{\iota_k\}_{k=1,2,\ldots}$ and hence, $\{\widehat{\omega}_k\}_{k=1,2,\ldots}$ is a sequence of independent random variables. We recall the equation (4.9):

$$\widehat{Q}_k := \mathbb{C}(\widehat{\omega}_k) = K_k\,\mathbb{C}(\iota_k)\,K_k^\top = K_k\left[H_k\overline{P}_k H_k^\top + R_k\right]K_k^\top.$$

Moreover, as $\mathbb{E}[(x_k - \widehat{x}_k)] = 0$,

$$\begin{aligned}
\mathbb{E}[x_k^\top M x_k] &= \mathbb{E}[((x_k-\widehat{x}_k)+\widehat{x}_k)^\top M((x_k-\widehat{x}_k)+\widehat{x}_k)] \\
&= \mathbb{E}[(x_k-\widehat{x}_k)^\top M(x_k-\widehat{x}_k)] + \mathbb{E}[\widehat{x}_k^\top M\widehat{x}_k] + 2\,\mathbb{E}[(x_k-\widehat{x}_k)^\top M\widehat{x}_k] \\
&= \text{trace}(M\,\mathbb{C}(x_k-\widehat{x}_k)) + \mathbb{E}[\widehat{x}_k^\top M\widehat{x}_k] \\
&= \text{trace}(MP_k) + \mathbb{E}[\widehat{x}_k^\top M\widehat{x}_k].
\end{aligned}$$

Therefore, $J_n(x,\mathbf{u}) = \widehat{J}_n(\widehat{x}_0,\mathbf{u}) + \tilde{J}_n(x)$, where

$$\widehat{J}_n(\widehat{x}_0,\mathbf{u}) := \sum_{k=0}^{n-1}\mathbb{E}\left[\widehat{x}_k^\top M\widehat{x}_k + u_k^\top N u_k\right] + \mathbb{E}[\widehat{x}_n^\top M\widehat{x}_n],$$

$$\tilde{J}_n(x) := \sum_{k=0}^{n-1} trace(MP_k) + trace(MP_n).$$

Since $\tilde{J}_n(x)$ is independent of control, the original problem (4.12) is equivalent to minimizing $\widehat{J}_n(\widehat{x}_0,\mathbf{u})$, when the state $\widehat{x}_k$ solves $\widehat{x}_{k+1} = A\widehat{x}_k + Bu_k + \widehat{\omega}_{k+1}$ with a given $\widehat{x}_0$ and a mean-zero Gaussian noise process $\widehat{\omega}_k$ whose covariance matrix $\widehat{Q}_k$ is given by (4.9). This is exactly a linear quadratic regulator as formulated in Chapter 2.4, but with a new state variable $\widehat{x}$.

By Theorem 2.4.2, the optimal linear feedback control is given by $u_k^\star = -F_{(n-k-1)}\widehat{x}_k$, where F_k is as in Chapter 2.2.6. Note that the new state variable $\widehat{x}$ needs to be computed, using the Kalman filter equations. We summarize below these equations for an easy reference.

Problem. The linear quadratic Gaussian (LQG)) (stochastic) control problem is to minimize

$$J_n(x,\mathbf{u}) := \sum_{k=0}^{n-1}\mathbb{E}\left[x_k^\top M x_k + u_k^\top N u_k\right] + \mathbb{E}[x_n^\top \widehat{M} x_n],$$

subject to dynamics $x_0 = x$, $x_{k+1} = Ax_k + Bu_k + \omega_{k+1}$, for $k = 0,1,\ldots$, and where the control u_k is a function of the observations $z_1,\ldots,z_k$ (with $z_k = Hx_k + \nu_k$), for $k = 0,1,\ldots$.

Equivalent Problem. Consider the standard (fully observed) linear quadratic optimal control problem to minimize

$$\widehat{J}_n(\widehat{x}_0,\mathbf{u}) := \sum_{k=0}^{n-1}\mathbb{E}\left[\widehat{x}_k^\top M\widehat{x}_k + u_k^\top N u_k\right] + \mathbb{E}[\widehat{x}_n^\top \widehat{M}\widehat{x}_n],$$

subject to dynamics $\widehat{x}_0 = x$, $\widehat{x}_{k+1} = A\widehat{x}_k + Bu_k + \widehat{\omega}_{k+1}$, for $k = 0,1,\ldots$, and where the control u_k is a function of $\widehat{x}_1,\ldots,\widehat{x}_k$, for $k = 0,1,\ldots$. The minimizer of this problem is the minimizer of the linear quadratic Gaussian control problem given above.

We leverage the above equivalence to construct the solutions as follows.

Solution of the LQGR problem. By Theorem 2.2.6, the *optimal linear feedback control* is given by $u_k^\star = -F_{(n-k-1)}\,\widehat{x}_k$, where

$$F_k = (N + B^\top V_k B)^{-1} B^\top V_k A$$
$$V_{k+1} = M + A^\top V_k A - A^\top V_k B \left(N + B^\top V_k B\right)^{-1} B^\top V_k A, \qquad k = 0, 1, \ldots, n-1.$$

The new state variable $\widehat{x}$ is as in Theorem 4.2.3: starting with a given $\widehat{x}_0$ and P_0, its dynamic is given for all $k = 1, 2 \ldots$ by

$$\widehat{x}_k = A_{k-1}\,\widehat{x}_{k-1} + B\,u_k + K_k\,[z_k - A_{k-1}\,\widehat{x}_{k-1}],$$

with

$$\overline{P}_k = \widehat{Q}_k + A_{k-1}\,P_{k-1}\,A_{k-1}^\top, \quad K_k = \overline{P}_k\,H_k^\top \left(H_k\,\overline{P}_k\,H_k^\top + R_k\right)^{-1}, \quad P_k = (I - K_k\,H_k)\,\overline{P}_k$$

where $\widehat{Q}_k$ is as in (4.9):

$$\widehat{Q}_k := \mathbb{C}(\widehat{\omega}_k) = K_k\,\mathbb{C}(\iota_k)\,K_k^\top = K_k \left[H_k\,\overline{P}_k\,H_k^\top + R_k\right] K_k^\top.$$

Recall that the above equations can be expressed as one Riccati equation:

$$\overline{P}_{k+1} = \widehat{Q}_{k+1} + A_k\,\overline{P}_k\,A_k^\top - A_k\,\overline{P}_k\,H_k^\top \left(H_k\,\overline{P}_k\,H_k^\top\right)^{-1} H_k\,\overline{P}_k\,A_k^\top.$$

R The above solution is obtained by replacing the state process x_k by its estimate $\widehat{x}_k$ and then conTrolling the resulting standard *fully-observed* problem; it is fully-observed as we know that the new state $\widehat{x}_k$. This is known as the separation principle, and it *rarely holds* outside the linear, quadratic, Gaussian structure.

4.3.3 Solution via Riccati Equations : Infinite Horizon

Infinite Horizon Problem. The infinite horizon linear quadratic Gaussian (LQG)) (stochastic) control problem is to minimize

$$J_\infty(x, \mathbf{u}) := \sum_{k=0}^{\infty} \mathbb{E}\,\rho^k \left[x_k^\top M x_k + u_k^\top N u_k\right],$$

subject to dynamics $x_0 = x$, $x_{k+1} = Ax_k + Bu_k + \omega_{k+1}$, for $k = 0, 1, \ldots$, and where the control u_k is a function of the observations $z_1, \ldots, z_k$ (with $z_k = H x_k + \nu_k$), for $k = 0, 1, \ldots$.

As in the finite horizon case, we obtain an equivalent problem by the separation principle replacing the state x_k by its estimate $\widehat{x}_k$ and the equations for the *"new state"* process $\widehat{x}_k$:

$$\widehat{x}_{k+1} = A\widehat{x}_k + Bu_k + \widehat{\omega}_{k+1},$$

and the noise process $\widehat{\omega}_k$ is independent of each other, but may have time-dependent covariances.

To obtain a time-homogenous problem, we assume that all given matrices A, B, H, Q, R, M, N are all independent of time and we consider the infinite horizon problem. Still, for a general initial covariance, the state dynamics for $\widehat{x}_k$, in fact the covariance $\widehat{Q}_k$ given in (4.9) of the noise $\widehat{\omega}_k$, is time-dependent. To alleviate this difficulty, we assume that P_0 is equal to its long time limit $P_\star$ constructed in subsection 4.2.4. Then, for all k we have

$$(P_k, \overline{P}_k, K_k) = (P_\star, \overline{P}_\star, K_\star), \qquad k = 1, 2, \dots$$

Moreover, the innovation process $\imath_k$ as well as the noise process $\widehat{\omega}_k$ are both i.i.d., and with the covariance matrix $\widehat{Q}_* := \mathbb{C}(\widehat{\omega}_k) = K_* \left[H \overline{P}_* H^\top + R \right] K_*^\top$.

This equivalent problem can be solved as a regular infinite horizon LQR with state $\widehat{x}_k$. Here are the outcome of the numerical experiments for the example studied in Section 4.1.

R *Case* $\rho = 1$. In many engineering applications, there is no reason to discount. However, since the covariances of the noise process $\widehat{\omega}_k$ do not vanish, without a discount the cost functionals would be infinite. But if the Ricatti equations are solvable, then one can use the feedback control corresponding to $\rho = 1$. In fact, this control is optimal for appropriately renormalized control problem. In many of our numerical experiments, we use this observation and do not use discounting.

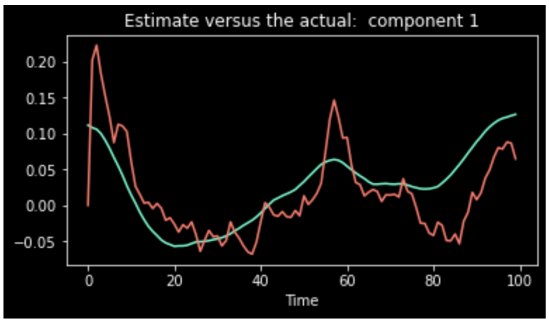


Figure 4.3: Position

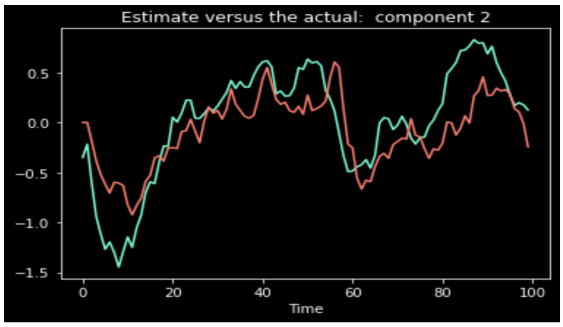


Figure 4.4: Velocity

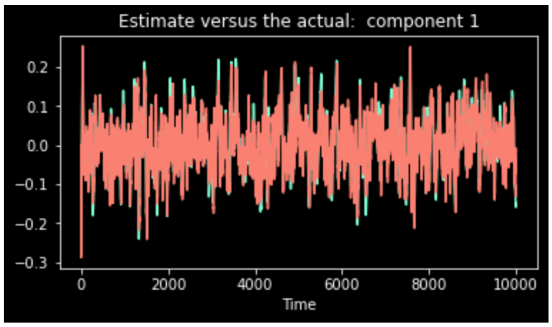


Figure 4.5: Position

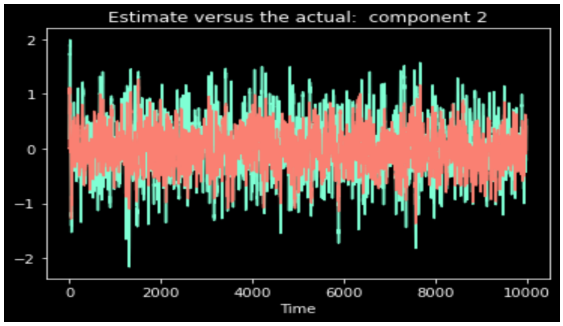
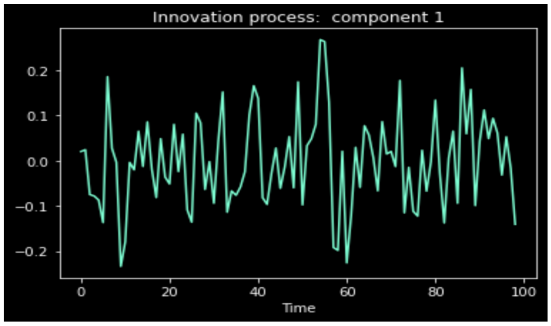


Figure 4.6: Velocity



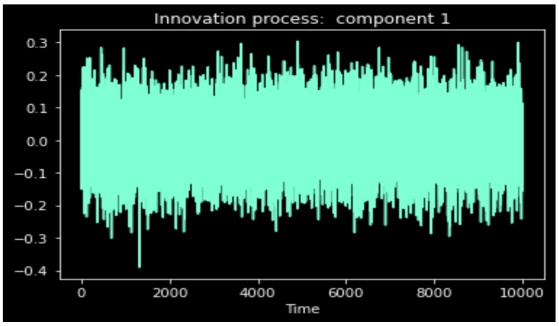


Figure 4.7: Innovation process

4.4 Exercises

Problem 4.1 Consider a one dimensional Kalman filter problem with $a = 1$, $h = 3/2$ $p_0 = p = r = 1$ and $\widehat{x}_0 = 1$.

a. Compute $\bar{p}_k$ and p_k for each $k = 1, 2$.

b. Suppose that all limits $\bar{p}^* := \lim_{k\to\infty} \bar{p}_k$, $K^* := \lim_{k\to\infty} K_k$ and $p^* := \lim_{k\to\infty} p_k$ exists. Compute $\bar{p}^*$, K^*, p^*.

c. Suppose that p_0 is equal to p^* computed in part a. Show that p_k, $\bar{p}_k$, and K_k are all independent of k and are equal to p^*, $\bar{p}^*$, K^*, respectively.

d. With p_0 equal to p^*, show that the innovation process is i.i.d.

Problem 4.2 Consider the one-dimensional system

$$x_{k+1} = \sqrt{3}x_k + \omega_{k+1}, \qquad k = 0, 1, \ldots,$$

where $x_0 \sim \mathcal{N}(1, (1/3))$, and $\omega_{k+1} \sim \mathcal{N}(0, 1)$. Observations are

$$z_k = x_k + v_k, \qquad v_k \sim \mathcal{N}(0, 2/5).$$

As in the lectures, set $\widehat{x}_0 = \mathbb{E}[x_0]$, $\widehat{x}_{1|0} = \mathbb{E}[x_1]$ and for $k = 1, 2, \ldots$

$$\widehat{x}_k = \mathbb{E}[x_k \,|\, z_1, \ldots, z_k], \qquad \widehat{x}_{k+1|k} = \mathbb{E}[x_{k+1} \,|\, z_1, \ldots, z_k],$$

$$P_k = E[(x_k - \widehat{x}_k)^2 \,|\, z_1, \ldots, z_k], \qquad \bar{P}_{k+1} = E[(x_k - \widehat{x}_{k+1|k})^2 \,|\, z_1, \ldots, z_k].$$

a. Compute P_k and $\overline{P}_k$ for all $k = 1, 2, \ldots$.

b. Compute $\hat{x}_{1|0}, \hat{x}_1, \hat{x}_{2|1}, \hat{x}_2$ as functions of z_1 and z_2.

c. Let ι_k be the innovation process. Compute $\mathbb{E}[\,\iota_k^2\,]$ for each $k = 1, 2, \ldots$

d. Compute $\mathbb{E}[x_k^2]$ for each $k = 1, 2, \ldots$

Problem 4.3 Consider the one-dimensional system:

$$x_{k+1} = 5x_k + \omega_{k+1}, \qquad k = 0, 1, \ldots,$$

where $x_0 \sim \mathcal{N}(0.5, 2)$, and $\omega_{k+1} \sim \mathcal{N}(0, 3)$. Observations are

$$z_k = x_k + v_k, \qquad v_k \sim \mathcal{N}(0, 4).$$

As in the lectures, set $\widehat{x}_0 = \mathbb{E}[x_0]$, $\widehat{x}_{1|0} = \mathbb{E}[x_1]$ and for $k = 1, 2, \ldots$

$$\widehat{x}_k = \mathbb{E}[x_k \,|\, z_1, \ldots, z_k], \qquad \widehat{x}_{k+1|k} = \mathbb{E}[x_{k+1} \,|\, z_1, \ldots, z_k],$$

$$P_k = E[(x_k - \widehat{x}_k)^2 \,|\, z_1, \ldots, z_k], \qquad \bar{P}_{k+1} = E[(x_k - \widehat{x}_{k+1|k})^2 \,|\, z_1, \ldots, z_k].$$

a. Compute P_0, P_1, P_2, K_1, K_2 and $\overline{P}_1, \overline{P}_2$.

b. Compute $\hat{x}_{1|0}, \hat{x}_1, \hat{x}_{2|1}, \hat{x}_2, \hat{x}_{3|2}$ as functions of z_1 and z_2.

c. Let ι_k be the innovation process. Compute $\mathbb{E}[\,\iota_k^2\,]$ for $k = 1, 2, 3$.

d. Compute $\mathbb{E}[x_k^2]$ for each $k = 1, 2, \ldots$

Problem 4.4 Consider the one-dimensional system:

$$x_{k+1} = \frac{1}{\sqrt{2}} x_k + u_k + \omega_{k+1}, \qquad k = 0, 1, \ldots,$$

where $x_0 \sim \mathcal{N}\left(3, \frac{2\sqrt{2}}{1+\sqrt{2}}\right)$, and $\omega_{k+1} \sim \mathcal{N}(0, \sqrt{2})$. Observations are

$$z_k = x_k + \nu_k, \qquad \nu_k \sim \mathcal{N}(0, 2\sqrt{2}).$$

The control u_k is a function the observations $z_1, \ldots, z_k$. But since the state-estimate $\widehat{x}_k$ is a function of $z_1, \ldots, z_k$, the control can also use $\widehat{x}_k$.

a. Compute P_k, K_k and $\overline{P}_k$ for all $k = 1, 2, \ldots$.

b. Compute $\widehat{x}_{1|0}, \widehat{x}_1, \widehat{x}_{2|1}, \widehat{x}_2, \widehat{x}_{3|2}$ as functions of z_1, u_1 and z_2, u_2.

c. Suppose $u_k \equiv 0$ for all k. Compute $\mathbb{E}[x_k^2]$ for $k = 1, 2, \ldots$

d. Suppose $u_k = (1/\sqrt{2})\widehat{x}_k$ for k. Compute $\mathbb{E}[x_k^2]$ for $k = 1, 2, \ldots$

e. Suppose

$$u_k = \frac{1}{\sqrt{2}} \widehat{x}_k, \qquad k = 0, 1, 2, \ldots$$

Compute $\mathbb{E}[x_k^2]$ for $k = 1, 2, \ldots$

Problem 4.5 Suppose that $d = 2$, $\ell = 1$, $\hat{x} = 0 \in \mathbb{R}^2$, $A_k = I$ is the 2×2 identity matrix, I_2, $H_k = [1, 1]$ and $P_0 = Q_k = I_2$, and $R_k = 1$. Hence, $x_0 \sim \mathcal{N}(0, I_2)$,

$$x_{k+1} = x_k + w_k, \quad z_k = [1, 1]x_k + \nu_k, \quad \omega_k \sim \mathcal{N}(0, I_2), \quad \nu_k \sim \mathcal{N}(0, 1).$$

a. Compute $\overline{P}_1, K_1, P_1, \bar{P}_2, K_2$.

b. Compute $\hat{x}_{1|0}, \hat{x}_1, \hat{x}_{2|1}, \hat{x}_2$ as functions of z_1 and z_2.

c. Let $x_k = (v_k, s_k)$. Argue that for each k, $\widehat{v}_k = \widehat{s}_k$ and

$$P_k = \begin{bmatrix} b_k & c_k \\ c_k & b_k \end{bmatrix}, \quad \bar{P}_k = \begin{bmatrix} \bar{b}_k & \bar{c}_k \\ \bar{c}_k & \bar{b}_k \end{bmatrix}, \quad K_k = \begin{bmatrix} m_k \\ m_k \end{bmatrix},$$

for some b_k, c_k, $\bar{b}_k$, $\bar{c}_k$, and m_k. Is this consistent with part a?

d. Set $y_k := v_k + s_k$. Formulate a Kalman filter using only y_k as the state.

e. Let $p_k = \mathbb{E}[(y_k - \widehat{y}_k)^2]$. Show that $p_k = 2[b_k + c_k]$.

Problem 4.6 Consider the two-dimensional system:

$$x_{k+1} = A x_k + \omega_{k+1}, \qquad \text{where} \quad A = \begin{bmatrix} 1 & -1 \\ 0 & 0.8 \end{bmatrix}, \quad \omega_{k+1} := \begin{bmatrix} 1 \\ 1 \end{bmatrix} \xi_{k+1}.$$

where $x_0 \sim \mathcal{N}(\widehat{x}_0, P_0)$, and $\xi_{k+1} \sim \mathcal{N}(0, 2)$.

Observations with Gaussian noise $\nu_k \sim \mathcal{N}(0, 3)$ are:

$$z_k = H x_k + \nu_k, \qquad \text{where} \qquad H = [1 \;\; 1].$$

We are given that

$$\widehat{x}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad P_0 = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}.$$

a. What are $Q = \mathbb{C}(\omega_k)$ and $R_k = \mathbb{C}(\nu_k)$? Write a python code that calculates P_k, K_k and $\overline{P}_k$ the first 1,000-the of them. Print the 1,000th triple.

b. Numerically, compute the limiting values of the covariances:

$$P_* := \lim_{k\to\infty} P_k, \qquad \overline{P}_* := \lim_{k\to\infty} \overline{P}_k.$$

c. Simulate the first $1,000$ x_k.
d. Using the above x values, simulate the first $1,000$ z_k.
e. Using the above x, z value, compute the first $1,000$ $\widehat{x}_k$. Plot the map $k \mapsto \widehat{x}_k$ for $k = 1, \ldots, 1000$.

Problem 4.7 Consider the above system with control, i.e,

$$x_{k+1} = A\,x_k + B\,u_k + \omega_{k+1}$$

where all parameters are as before and $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Suppose that

$$u_k = [-0.9 \;\; 0.9]\,\widehat{x}_k.$$

Note that the covariances are as in the third problem! So you do not need to compute them again!
a. Simulate the first $1,000$ x_k.
b. Using the above x values, simulate the first $1,000$ z_k.
c. Using the above x, z value, compute the first $1,000$ $\widehat{x}_k$. Plot the map $k \mapsto \widehat{x}_k$ for $k = 1, \ldots, 1000$.

LGR-Simulate and Plot

October 19, 2021

0.1 LGR WITH STATIONARY COEFFICIENTS

0.1.1 Covariance Matrix Calculations

[73]:
```
import numpy as np
from scipy import linalg as la
import matplotlib.pyplot as plt; plt.style.use('dark_background')
np.set_printoptions(suppress=True)

def kalman_var(a,h,p0,q,r,nfinal):
    # Computes the Kalman variances
    # Outputs are the sequences barP_k, P_k and K_k
    # We compute upto a given nfinal
    d, num_cols = a.shape
    Id=np.identity(d)
    p=[p0]
    barp=[0] # creates a dummny value for barP_0
    gains=[0] # creates a dummny value for K_0
    for n in range(1,nfinal+1):
        pbef=p[-1]
        barpn= a@pbef@a.T +q
        rr=((h@barpn)@(h.T))+r
        kn=(barpn@h.T)@la.inv(rr)
        pnext= (Id-(kn@h))@barpn
        p.append(pnext)
        barp.append(barpn)
        gains.append(kn)
    return p, barp, gains
```

0.1.2 Simulate with Control

[74]:
```
def kalman_simulate_control(a,h,b,f,p0,xhat0,q,r,nfinal):
    # simulates state and observations from Gaussian
    # output are the state and the observation
    # b is the B matrix
    #f is the matrix so that control = - f state
    dim,_ = a.shape
    ell,_ = h.shape
    pp, barp, k= kalman_var(a,h,p0,q,r,nfinal)
```

```
    xmu  =np.zeros(dim)
    omu  =np.zeros(ell)
    x0   =np.random.multivariate_normal(xmu,p0,1).T + xhat0
    x    =[x0]
    xhat =[xhat0]
    obs=[0]  # dummy observation
    for t in range (1,nfinal+1):
        omegat  = np.random.multivariate_normal(xmu,q,1).T
        control = -(b@f)@xhat[-1]
        xt      = (a@x[-1]) + omegat + control
        #print('x', xt)
        #print()
        nut     = np.random.multivariate_normal(omu,r,1).T
        zt      = h@xt + nut
        xhatpre = a@xhat[-1] + control
        iotat   = zt -(h@xhatpre)
        xhatt   = xhatpre + (k[t]@iotat)
        #print('xhat, k(t)',xhatt, k[t])
        #print()
        x.append(xt)
        obs.append(zt)
        xhat.append(xhatt)
    #print(x[-1],xhat[-1],obs[-1])
    return x, obs, xhat
```

0.1.3 Visualization with control

[75]:
```
def kalman_plot_control(a,h,b,f,p0,xhat0,q,r,nfinal):
    #ploting the component 'com'
    # the actual simulated state - x - is given
    # estimate is using the simualted observations - obs-
    plt.figure(figsize=(6,3))

    dim,_=a.shape
    ell,_=h.shape
    x, obs, xhat=kalman_simulate_control(a,h,b,f,p0,xhat0,q,r,nfinal)

    #printing the state and its estimate
    print("\nSTATE and ITS ESTIMATE\n")
    for com in range(dim):
        print("This is the %d-th component\n"%(com+1))
        xcom=[x[t][com,0] for t in range(nfinal)]
        xhatcom=[xhat[t][com,0] for t in range(nfinal)]

        plt.plot(np.arange(nfinal),xcom,color ="aquamarine")
        plt.plot(np.arange(nfinal),xhatcom,color ="salmon")
```

```
        plt.xlabel("Time"); plt.title("Estimate versus the actual:  component␣
↪%d"%(com+1))
        plt.show()

        #ploting the difference
        diff=[x[t][com,0]-xhat[t][com,0] for t in range(nfinal)]
        plt.plot(np.arange(nfinal),diff,color ="steelblue")
        plt.xlabel("Time"); plt.title("Difference between actual and estimate:␣
↪component %d "%(com+1))
        plt.show()

    #printing the innovation
    print("\nINNOVATION\n")
    for ind in range(ell):
        print("This is the %d-th component\n"%(ind+1))
        iota_ell=[obs[t][ind,0]-(a@xhat[t])[ind,0] for t in range(1,nfinal)]

        plt.plot(np.arange(nfinal-1),iota_ell,color ="aquamarine")
        plt.xlabel("Time"); plt.title("Innovation process:  component␣
↪%d"%(ind+1))
        plt.show()


    #Printing the final Covariance Matrix
    P,barP, K =kalman_var(A,H,P0,Q,R,nfinal)
    print("\nThe final Error Covariance is\n")
    print(P[-1])
    print("\nThe final Pre-observation Error Covariance is\n")
    print(barP[-1])
    print("\nThe final Gains Matrix is\n")
    print(K[-1])
    print()
```

0.1.4 Example: Simple Position Estimator

Input

[83]:
```
h=0.01 # time step
var1= h # strength of the dynamics noise
var2= h # strength of the observation noise
A=np.array([[1,h], [0,1]])
H = np.array([[1.,0.]])
Q=np.array([[0.,0.], [0.,var1]])
R=np.array([[var2]])
p01=0.1  #uncertainty on initial position
p02=0.1  #uncertainty on initial velocity
P0=np.array([[p01,0.], [0.,p02]])
```

```
Xhat0=np.reshape(np.zeros(2),(2,1))

B=np.reshape(np.array([[0.],[1.]]),(2,1))
M = np.array([[1.,0.], [0.,0.]])
N = np.array([[1]])
```

Gains Matrix

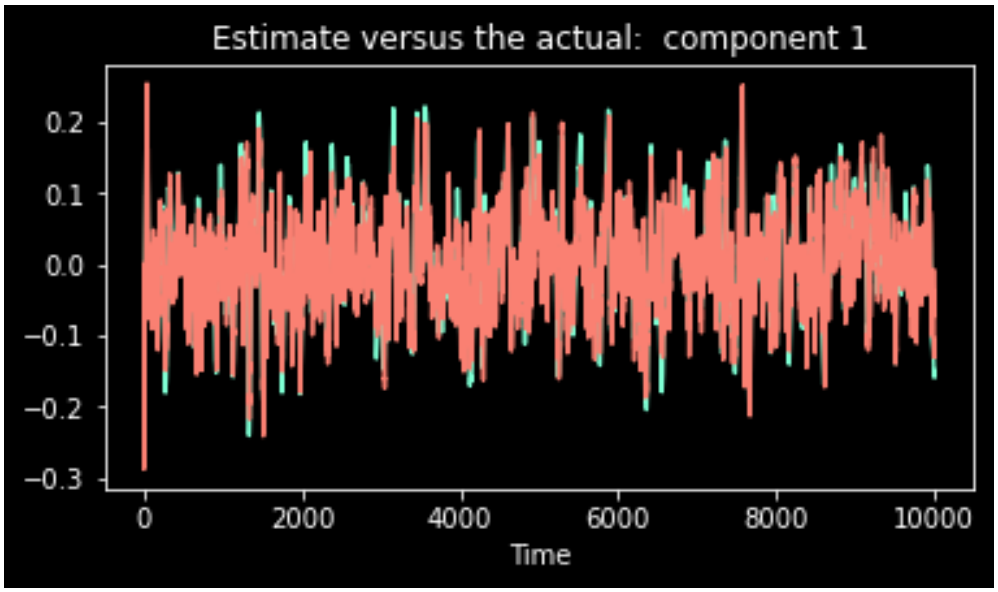
[84]:
```
def dlqr(a, b, m, n):
    # Get the feedback controls from"
    # linearized system at the current time step
    # for a discrete time system Ax+Bu
    # find the infinite horizon optimal feedback controller
    # to steer the system to the origin
    # with u = -F*x

    v = la.solve_discrete_are(a, b, m, n)
    f = la.inv((b.T @ v) @ b + n) @ (b.T @( v @ a))
    return f , v

F,_ = dlqr(A,B,M,N)
print("Gains matrix is given by F=", F)
eif,_ = la.eig(A-B@F)
print("Its eigenvalues are",eif)
```

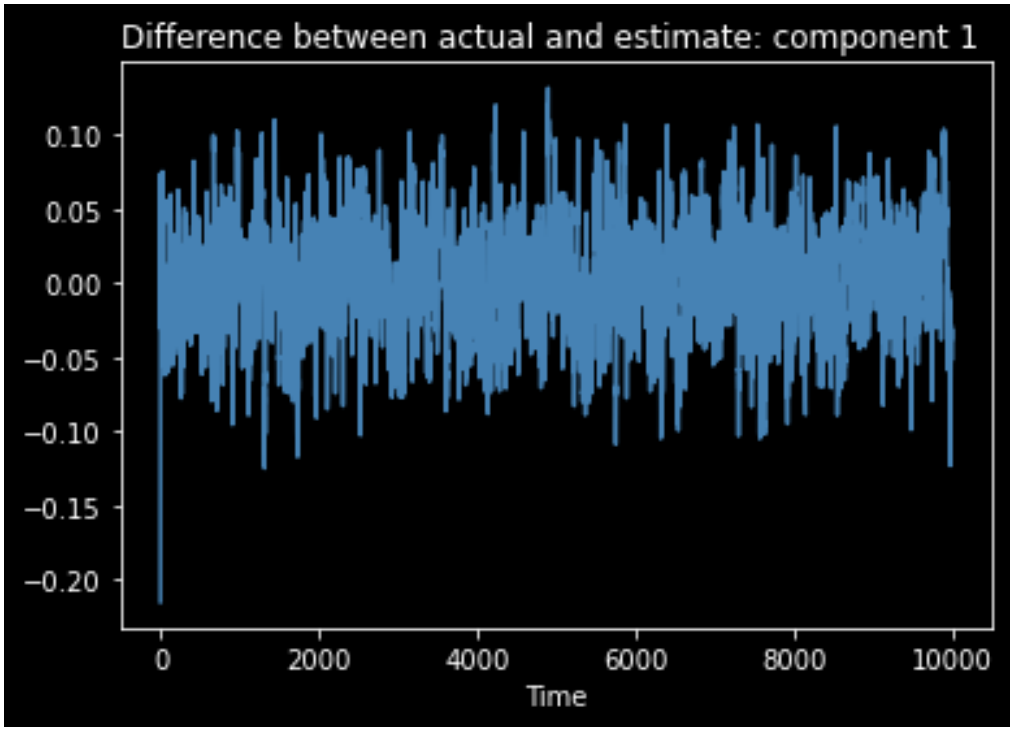
```
Gains matrix is given by F= [[0.931704   0.14124469]]
Its eigenvalues are [0.92937765+0.06579912j 0.92937765-0.06579912j]
```

Results

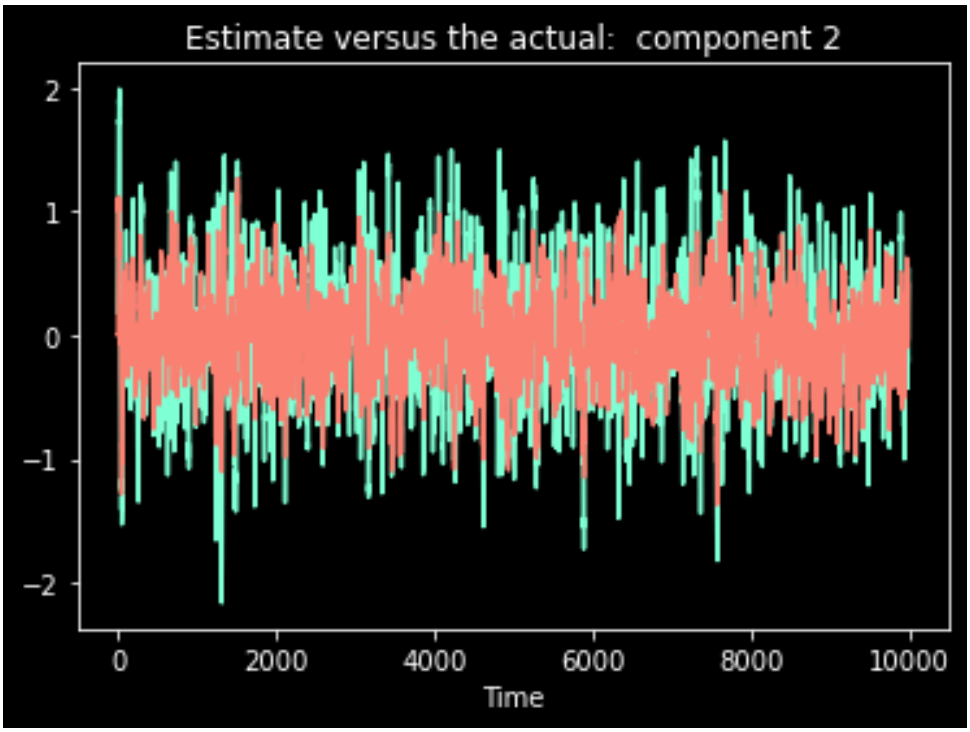
[85]:
```
maturity=10**4

kalman_plot_control(A,H,B,F,P0,Xhat0,Q,R, maturity)
```

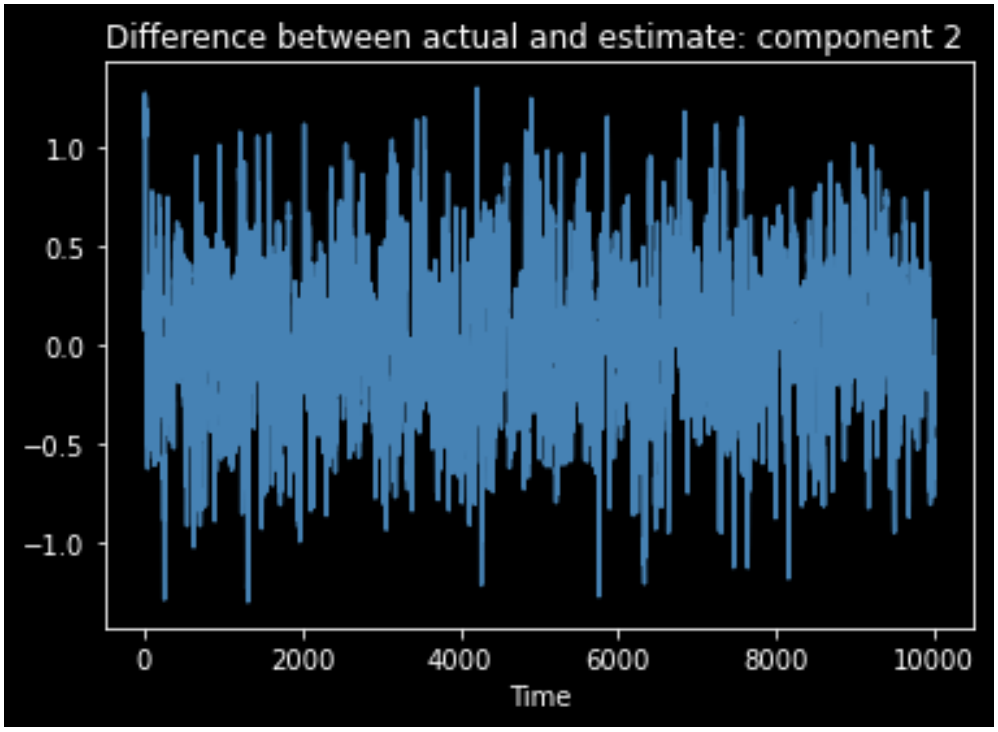
```
STATE and ITS ESTIMATE

This is the 1-th component
```





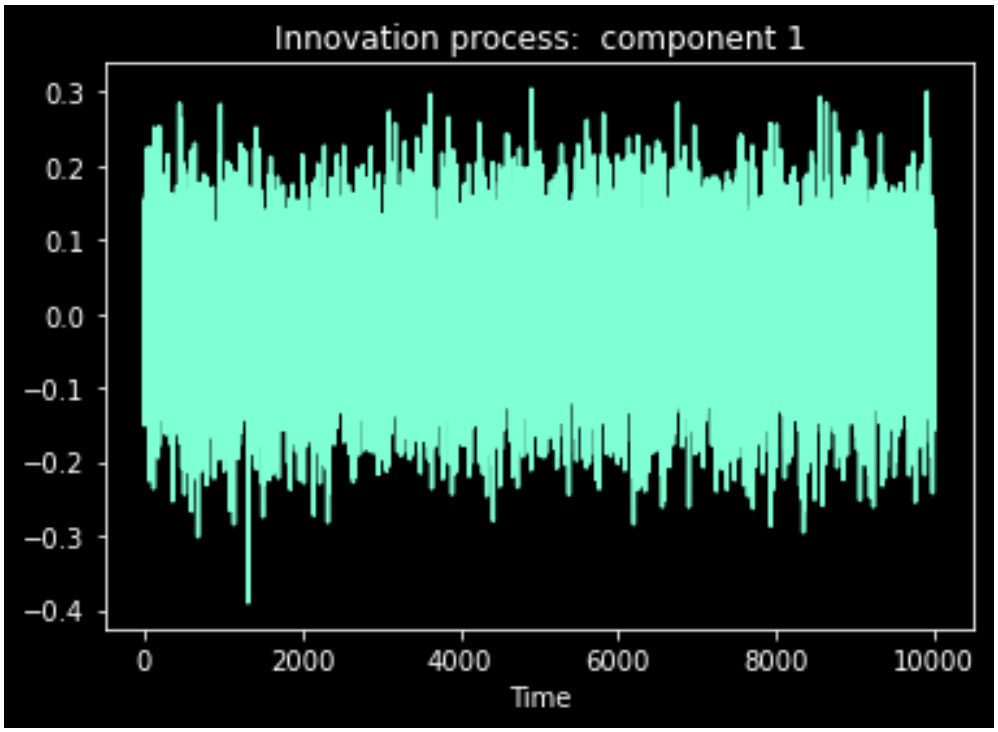

```
This is the 2-th component
```


Estimate versus the actual: component 2
2
1
0
−1
−2
0
2000
4000
6000
8000
10000
Time


Difference between actual and estimate: component 2
1.0
0.5
0.0
−0.5
−1.0
0
2000
4000
6000
8000
10000
Time

INNOVATION

This is the 1-th component



The final Error Covariance is

```
[[0.00131928 0.00931704]
 [0.00931704 0.14159824]]
```

The final Pre-observation Error Covariance is

```
[[0.00151978 0.01073302]
 [0.01073302 0.15159824]]
```

The final Gains Matrix is

```
[[0.13192765]
 [0.931704  ]]
```

5. Markov Decision Processes - MDP

In this chapter, we introduce and study the classical concept of Markov Decision processes. For now, this is similar to an optimal control problem without learning, but an extension with learning will be considered in the next (final) chapter.

5.1 Formulation

We consider a *controlled Markov chain* taking values in a space $\mathscr{X}$. The dynamics of the Markov Chain can be influenced by some actions chosen in a set $\mathscr{A}$. To simplify, we consider here a finite state space $\mathscr{X} := \{x^{(1)}, \dots, x^{(M)}\}$, $M \in \mathbb{N}^\star$, as well as a finite action space $\mathscr{A} := \{a^{(1)}, \dots, a^{(K)}\}$, $K \in \mathbb{N}^\star$. At a state $x \in \mathscr{X}$ if an action $a \in \mathscr{A}$ is chosen, then the state moves to a new state $y \in \mathscr{X}$ with probability $p(x,y,a)$ and a reward (or cost) $r(x,y,a)$ is obtained.

We begin with an illustrative example.

5.1.1 A first illustrative example

■ **Example 5.1 — Recycling Robot.** This very stylized example is taken from the classical textbook of Sutton and Barto: *Reinforcement Learning: an Introduction*. The description of the problem was previously introduced in the Introduction, but we recall it here for convenience.

The main goal is to optimize the journey of a mobile robot. This robot runs on a rechargeable battery and can actively collect cans in an office on his onboard bin, wait for people to bring cans to it, or go to the charging station. However, the robot only has a rough estimate of the remaining battery, and if it runs out of battery, it has to be manually placed (by a non-robot person) at the charging station. To simplify, we will assume that the battery has two states, high and low. In each of the state, the robot should decide its optimal action: collect, wait, or go to the charging station. The chosen action will have an impact on the expected number of cans collected, but also on the charging status of the battery. Here, we will assume that collecting decreases the battery with a given probability, going to the charging station increases it to high level, and waiting has no impact on the battery.

The goal here is to maximize the number of cans collected (a random reward) while avoiding being stuck without power outside the charging station, which would incur a significant penalty. ■

As mentioned above, in this simplified model, we summarize the state of the robot by its battery level, which can be either high 'h' or low 'ℓ'. In each of the two states, the robot can choose among three possible actions, that are search 's', wait 'w', or go to charging station 'g'. Using the previous notations, we thus have:

$$\mathscr{X} := \{\ell, h\} \quad \text{and} \quad \mathscr{A} := \{s, w, g\}.$$

To properly define the *controlled Markov chain* associated to this problem, we need to introduce the *transition probabilities* from one state to another, depending on the chosen action. As mentioned in the example's description, if the robot is in the search mode, the level of the battery will decrease with a given probability. More precisely, we assume that if the robot is in the search mode and has high battery, then the level of the battery will remain high with probability p_h, or decrease to the low level with probability $1 - p_h$. On the other hand, if the battery is already low when choosing the search mode, it will stay low with probability p_ℓ, or the robot will run out of battery with probability $1 - p_\ell$. In the latter case, the robot needs to be rescued, which incurs a penalty of $-r_p < 0$, and automatically placed on charged mode. We further assume that in the wait mode, the battery level does not change, and in the charging mode, the battery returns to the high level with probability 1.

The reward is equal to the expected number of collected cans if the robot does not run out of battery, and equal to a penalty term if the robot runs out of battery and has to be recovered. During the search mode, we define by r_s the expected number of collected cans if it does not run out of battery, while in the wait mode, it is equal to r_w. When charging, the reward is equal to 0, and as mentioned above, if the robot runs out of battery when searching, the reward is negative (penalty), equal to $-r_p < 0$.

The natural goal is to maximize the expectation of the sum of the (discounted) rewards. As in the Linear-Quadratic problem, we may consider either a finite or infinite time horizon. The discounted infinite horizon corresponds to the case in which the robot has no final time and keeps repeating the same task, while a finite horizon problem could be used to model a single day. We will focus here on the infinite time horizon problem. We remark that now, we assume that the rewards in different modes are known, and there is no need for learning. An extension which allows for learning will be discussed in the next chapter.

In order to formalize the problem as a Controlled Markov Chain, we first denote respectively by x_t and a_t the state of the battery and the action chosen at time $t \geqslant 0$. For all t, we should have $x_t \in \mathscr{X} := \{\ell, h\}$ and $a_t \in \mathscr{A} := \{s, w, g\}$. For any $x \in \mathscr{X}$, $y \in \mathscr{X}$ and $a \in \mathscr{A}$, we then define the transition probabilities and the rewards as follows:

$$\begin{aligned} p(x,y,a) &:= \mathbb{P}(x_{t+1} = y \mid x_t = x, a_t = a) \text{ and} \\ r(x,y,a) &:= \text{reward when } x_t = x, a_t = a, x_{t+1} = y. \end{aligned}$$

Note that in this example, the transition probabilities and the reward do not depend on time. As already mentioned, the goal is to maximize the sum of the expected reward with infinite horizon and given discount factor $\rho < 1$, *i.e.*

$$J := \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^k L(x_t, a_t)\right], \text{ where } L(x,a) := \mathbb{E}\big[r(x_t, y_{t+1}, a_t) \mid x_t = x, a_t = a\big],\ x \in \mathscr{X}, a \in \mathscr{A}.$$

Figure 5.1 below summarizes all the above information about transition probabilities and rewards, while Figure 5.2 gives the value of the function $L(x,a)$ for all $x \in \mathscr{X}, a \in \mathscr{A}$.

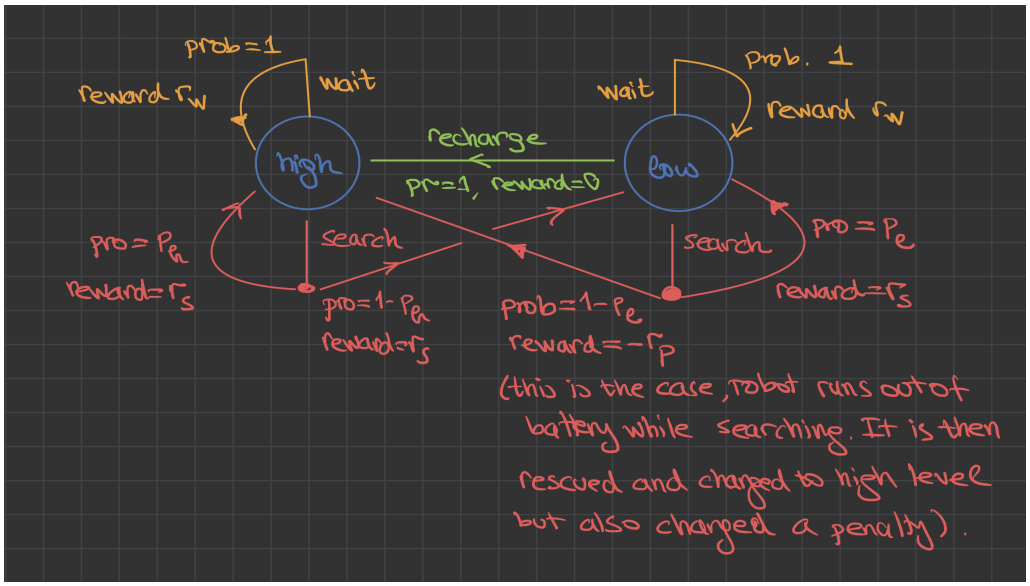
State at t $x_t = x$	Action chosen $a_t = a$	State at $t+1$ $x_{t+1} = y$	Transition probability $p(x,y,a)$	Associated reward $r(x,y,a)$
h	s	h	p_h	r_s
h	s	ℓ	$1-p_h$	r_s
h	w	h	1	r_w
h	g	h	1	0
ℓ	s	ℓ	p_ℓ	r_s
ℓ	s	h	$1-p_\ell$	$-r_p$
ℓ	w	ℓ	1	r_w
ℓ	g	h	1	0

Figure 5.1: Transition probabilities and rewards

Note that when $x_t = \ell, a_t = s$, the robot may run out of battery. In that case, it is taken to a charging station and charged up to full battery, and the next state will be h with cost $-c_p$. The expected cost for each state $L(x,a)$, action pair (x,a) is given in the table below.

State x	h	h	h	ℓ	ℓ	ℓ
Action a	s	w	g	s	w	g
Function $L(x,a)$	r_s	r_w	0	$p_\ell r_s - (1-p_\ell) r_p$	r_w	0

Figure 5.2: Expected reward



5.1.2 Mathematical formulation

We now consider a *controlled Markov chain* taking values in a finite state space $\mathscr{X} := \{x^{(1)}, \ldots, x^{(M)}\}$, and with a finite action space $\mathscr{A} := \{a^{(1)}, \ldots, a^{(K)}\}$. We denote by $x_t \in \mathscr{X}$ the state of the system and by $a_t \in \mathscr{A}$ the action chosen at time t. The dynamics of the Markov process is given as follows: at any time t, if the state is $x_t = x \in \mathscr{X}$ and an action $a_t = a \in \mathscr{A}$ is chosen, then with some probability defined below, the state moves to a new state $x_{t+1} = y \in \mathscr{X}$ and a reward $r(t,x,y,a)$ is received. The transition probability may depend on time, on the current state, on the chosen action, and on the new state:

$$p(t,x,y,a) := \mathbb{P}(x_{t+1} = y \mid x_t = x, a_t = a), \qquad (t,x,y,a) \in \mathbb{N} \times \mathscr{X} \times \mathscr{X} \times \mathscr{A}.$$

In what follows, we denote by $\mathbf{x} := (x_0, x_1, \ldots)$ the controlled Markov chain with action sequence $\mathbf{a} := (a_0, a_1, \ldots)$. The controlled Markov process $\mathbf{x}$ depends on the chosen action sequence $\mathbf{a}$, but for simplicity we suppress this dependency in the notation.

The set of admissible set of controls needs to be specified to complete the definition of the control problem. We consider the following different classes of controls.

Definition 5.1.1 — Adapted actions. We call an action sequence $\mathbf{a} := (a_0, a_1, \ldots)$ *adapted* if for each t, the action $a_t \in \mathscr{A}$ is a function of time t and the state values $(x_0, \ldots, x_t)$ observed up to time t. We let $\mathscr{A}_d$ be the set of all adapted action sequences $\mathbf{a} = (a_0, a_1, \ldots)$.

A special subclass of adapted actions are feedback actions.

Definition 5.1.2 — Feedback actions. An adapted action sequence $\mathbf{a}$ is a *feedback action* if there exists a function $A : \mathbb{N} \times \mathscr{X} \mapsto \mathscr{A}$ such that for all $t \in \mathbb{N}$, $a_t = A(t, x_t)$. In other words, the action $a_t \in \mathscr{A}$ chosen at time t is only a function of the current time t and the current state x_t. With a slight abuse of notation, the action sequence $(A(0, x_0), A(1, x_1), \ldots)$ is denoted by A.

Infinite time horizon.

Let $x_0 = x \in \mathscr{X}$ be the starting point of the Markov Chain at time $t = 0$. *The discounted infinite horizon* cost/reward function associated to an adapted action sequence $\mathbf{a} \in \mathscr{A}_d$ for a given discount factor $\rho \in (0, 1)$ is defined as follows:

$$J_\infty(x, \mathbf{a}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t L(x_t, a_t)\right],$$

where $L : \mathscr{X} \times \mathscr{A} \longrightarrow \mathbb{R}$ is given by,

$$L(x, a) := \mathbb{E}\big[r(x_t, x_{t+1}, a_t) \mid x_t = x, a_t = a\big].$$

For a feedback function $A : \mathscr{X} \mapsto \mathscr{A}$, we have,

$$J_\infty(x, A) := \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t L\big(x_t, A(x_t)\big) \middle| x_0 = x\right].$$

If the function r represents a cost, then the Markov Decision Process problem is defined as the following minimisation problem:

$$\text{minimise} \quad J_\infty(x, \mathbf{a}), \quad \text{over all} \quad \mathbf{a} \in \mathscr{A}_d,$$

and the value function of this problem is simply defined as the minimum possible value:

$$v_\infty(x) := \inf_{\mathbf{a} \in \mathscr{A}_d} J_\infty(x, \mathbf{a}), \qquad x \in \mathscr{X}. \tag{5.1}$$

On the contrary, if the function r represents a reward, then the problem of interest will be to maximize the expected reward, and thus consider the following value function:

$$v_\infty(x) := \sup_{\mathbf{a} \in \mathscr{A}_d} J_\infty(x, \mathbf{a}), \qquad x \in \mathscr{X}.$$

R For infinite time horizon problems, it is a desirable to have a "time-homogenous" problem. More precisely, in the infinite time horizon problem, the cost/reward as well as the transition probabilities generally do not depend on time.

Finite time horizon.
Similarly, we may consider *the finite horizon problem*, for a specific horizon time T. In this finite horizon case, the expected cost associated to an adapted action sequence $\mathbf{a} \in \mathscr{A}_d$, when starting at time t with state $x_t = x \in \mathscr{X}$ is defined by:

$$J_T(t,x,\mathbf{a}) := \mathbb{E}\left[\sum_{k=t}^{T-1} L(k,x_k,a_k) + \widehat{L}(x_T)\middle| x_t = x\right],$$

where, in this case, the cost function $L : \{t,t+1,\dots,T-1\} \times \mathscr{X} \times \mathscr{A} \longrightarrow \mathbb{R}$ can depend on time. Similarly, in this finite horizon case, the transition probabilities can depend on time as well. The value function of the corresponding minimization problem is also defined as the minimum possible value when starting at time t with state $x_t = x \in \mathscr{X}$:

$$v_T(t,x) := \inf_{\mathbf{a}\in\mathscr{A}_d} J_T(t,x,\mathbf{a}). \tag{5.2}$$

Note that when starting at time t, only the actions $a_t,\dots,a_{T-1}$ are relevant for this problem, and $v_T(T,x) = \widehat{L}(x)$, where $\widehat{L} : \mathscr{X} \longrightarrow \mathbb{R}$ is a terminal cost function.

Moreover, for any function $g : \mathscr{X} \mapsto \mathbb{R}$ and the state-action pair $(x,a) \in \mathscr{X} \times \mathscr{A}$,

$$\mathbb{E}[g(x_{t+1}) \mid x_t = x, a_t = a] = \sum_{y\in\mathscr{X}} g(y)p(t,x,y,a).$$

More generally, for any $t \in \mathbb{N}$ and $(x,a) \in \mathscr{X} \times \mathscr{A}$, we have:

$$L(t,x,a) := \mathbb{E}\big[r(t,x_t,x_{t+1},a_t) \mid x_t = x, a_t = a\big] = \sum_{y\in\mathscr{X}} r(t,x,y,a)p(t,x,y,a).$$

5.2 Dynamic programming: examples

As we have already seen in the framework of Linear-Quadratic optimization problems in Chapter 2, the *dynamic programming principle* is a powerful tool for problems with some 'time-like' variable. We give in this section several deterministic and stochastic examples to provide more insight on this principle. Associated python codes to solve some of the following examples will be provided in Canvas.

5.2.1 Knapsack problem

The Knapsack problem is a deterministic control problem. Suppose that a non-empty set A of items is given. Each item $a \in A$ is characterised by a value $v_a > 0$ and a weight $w_a > 0$, and we thus write $a = (v_a, w_a)$. Given a weight limit of $W \geqslant 0$, our goal is to choose a subset so as to maximize the total value while remaining within the weight limit.

Mathematically, for any subset $B \subset A$, the value and the weight of the set B are respectively defined by:

$$v(B) := \sum_{b\in B} v_b, \quad \text{and} \quad w(B) := \sum_{b\in B} w_b.$$

Then, the problem is to maximize $v(B)$ over all subsets $B \subset A$ satisfying the weight limit $w(B) \leqslant W$. Let $V(A,W)$ be the maximum possible value, *i.e.*

$$V(A,W) := \max_{B\subset A:\, w(B)\leqslant W} v(B).$$

If there is no subset B of A with $w(B) \leqslant W$, then we set $V(A,W) = 0$.

The basic approach to this problem would be to consider each subset and compute the associated values if they satisfy the weight limit. However, this method of exhaustion has high complexity as there are 2^n many subsets, where n is the number of elements in A. Thus, the number of computations increases exponentially with n. The dynamic programming principle on the other hand, allows us to reduce the complexity of the problem to a quadratic function of n.

To formulate the dynamic programming approach, we first "order" the elements of A:

$$A = \{a_1 = (v_1, w_1), \dots, a_n = (v_n, w_n)\},$$

where v_i represents the value of the element a_i and w_i is its weight. For any $k = 1, 2, \dots, n$, and any weight limit $0 \leqslant w \leqslant W$, we define $A_k := \{a_1, \dots, a_k\}$ and

$$V_k(w) = V(A_k, w).$$

In other words, $V_k(w)$ is the value function for a weight limit $0 \leqslant w \leqslant W$ when we restrict the set A to its k-first elements $A_k := \{a_1, \dots, a_k\}$. We naturally initialize $V_0(w) \equiv 0$, for any w. The dynamic programming principle is given by the following theorem.

Theorem 5.2.1 For any $k = 1, \dots, n$ and any $w \in [0, W]$, we have

$$V_k(w) = \begin{cases} V_{k-1}(w), & \text{if } w_k > w, \\ \max\{V_{k-1}(w), v_k + V_{k-1}(w - w_k)\}, & \text{if } w_k \leqslant w. \end{cases}$$

Proof.
Step 1. First, if $w_k > w$, then a_k cannot be part of any subset satisfying the weight limit. Hence, the subsets of A_k satisfying the weight limit are exactly the subsets of A_{k-1} satisfying this weight limit. Therefore, $V_k(w) = V_{k-1}(w)$.

Step 2. Now, suppose that $w_k \leqslant w$. We first claim that

$$V_k(w) \geqslant \max\{V_{k-1}(w), v_k + V_{k-1}(w - w_k)\}. \tag{5.3}$$

Indeed, $A_{k-1} \subset A_k$, we clearly have:

$$V_k(w) := \max_{B \subset A_k:\, w(B) \leqslant w} v(B) \geqslant \max_{B \subset A_{k-1}:\, w(B) \leqslant w} v(B) =: V_{k-1}(w).$$

It remains to prove that $V_k(w) \geqslant v_k + V_{k-1}(w - w_k))$. Let $B_{k-1}^{\tilde{\omega}} \subset A_{k-1}$ be an optimizer with weight limit $\tilde{\omega} := w - w_k \geqslant 0$. Then, $w(B_{k-1}^{\tilde{\omega}}) \leqslant w - w_k$ and

$$V_{k-1}(w - w_k) := \max_{B \subset A_{k-1}:\, w(B) \leqslant w - w_k} v(B) = v(B_{k-1}^{\tilde{\omega}}). \tag{5.4}$$

Set

$$C_k := \{a_k\} \cup B_{k-1}^{\tilde{\omega}}.$$

Since $\{a_k\} \in A_k$ and $B_{k-1}^{\tilde{\omega}} \subset A_{k-1}$, we have $C_k \subset A_k$ and $\{a_k\} \cap B_{k-1}^{\tilde{\omega}} = \varnothing$. These and (5.4) imply that

$$w(C_k) = \sum_{a \in C_k} w_a = w_k + \sum_{a \in B_{k-1}^{\tilde{\omega}}} w_a = w_k + w(B_{k-1}^{\tilde{\omega}}) \leqslant w_k + \tilde{\omega} = w,$$

$$\text{and } v(C_k) = \sum_{a \in C_k} v_a = v_k + \sum_{a \in B_{k-1}^{\tilde{\omega}}} v_a = v_k + v(B_{k-1}^{\tilde{\omega}}) = v_k + V_{k-1}(w - w_k).$$

Recall that, by definition, $V_k(w) = \max v(B)$ for $B \subset A_k$ and $w(B) \leqslant w$. We have also shown that the weight of C_k is smaller than the weight limit w, $C_k \subset A_k$ and $w(C_k) \leqslant w$. Hence,

$$v(C_k) \leqslant \max_{B \subset A_k:\, w(B) \leqslant w} v(B) = V_k(w).$$

Since the value of C_k satisfies $v(C_k) = v_k + V_{k-1}(w - w_k)$, we conclude that

$$v_k + V_{k-1}(w - w_k) \leqslant V_k(w), \tag{5.5}$$

and therefore inequality (5.3) holds.

Step 3. It remains to prove that (5.3) is in fact an equality. Similarly as before, consider $B_k^w \subset A_k$, an optimizer with weight limit $w \geqslant 0$. Then, $w(B_k^w) \leqslant w$ and

$$V_k(w) := \max_{B \subset A_k:\, w(B) \leqslant w} v(B) = v(B_k^w).$$

We analyze two cases separetely: $a_k \in B_k^w$ and $a_k \notin B_k^w$.
Case (i) If $a_k \notin B_k^w$, then $B_k^w \subset A_{k-1}$, and therefore,

$$V_k(w) = V_{k-1}(w), \qquad \text{if} \qquad a_k \notin B_k^w.$$

Case (ii) If $a_k \in B_k^w$ (and $w_k \leqslant w$), we set $C_{k-1} := B_k^w \backslash \{a_k\}$. Clearly, $C_{k-1} \subset A_{k-1}$ and

$$w(C_{k-1}) = w(B_k^w) - w_k \leqslant w - w_k.$$

As B_k^w is optimal, we have

$$V_{k-1}(w - w_k) := \max_{B \subset A_{k-1}:\, w(B) \leqslant w - w_k} v(B) \geqslant v(C_{k-1}) = v(B_k^w) - v_k = V_k(w) - v_k.$$

Reorganizing the previous equation, we finally obtain:

$$V_k(w) \leqslant v_k + V_{k-1}(w - w_k), \qquad \text{if} \qquad a_k \in B_k^w.$$

Combining the two cases, we conclude that either $V_k(w) \leqslant V_{k-1}(w)$ or $V_k(w) \leqslant v_k + V_{k-1}(w - w_k)$. Hence,

$$V_k(w) \leqslant \max\{V_{k-1}(w), v_k + V_{k-1}(w - w_k))\}.$$

This together with (5.3) implies that (5.3) holds as an equality. ■

5.2.2 Pots of gold

The "Pots of gold" example is a deterministic game between two players. We are given an *ordered* set of numbers representing number of gold coins in that bag:

$$A = \{x_1, x_2, \ldots, x_n\},\ n \in \mathbb{N}^\star.$$

There are two players, P_1 and P_2, who takes turn to pick one of the numbers at the edge, until all numbers are picked. More precisely, at the first round, P_1 can choose either x_1 or x_n. If P_1 chooses x_1, then P_2 can choose either x_2 or x_n. Conversely, if P_1 chooses x_n, then P_2 can choose either x_1 or x_{n-1}. Then, once P_2 has played, P_1 can play, etc. The goal is to maximize the sum of the collected numbers. This is a so-called *constant-sum game* (similar to *zero-sum game*), as the sum of the numbers collected by either player is always constant, equal to the sum of the original set, *i.e.* $\sum_{i=1}^n x_i$.

Exercise 5.1 Suppose $A = \{1, 12, 1, 2\}$. Describe all the possible outcomes and determine the optimal strategy for each player. ■

To describe the dynamic programming approach, we assume that both players make optimal decisions, and we introduce the following notations:

- k is the number of bags already collected, $k = 1, 2, \ldots, n$;
- i is the number of left bags collected, $i = 1, 2, \ldots, k$;
- $v_k(i)$ is the maximum coins that can be collected starting from state (k, i) by the player who has the turn to pick;
- $w_k(i)$ is the maximum coins that can be collected starting from state (k, i) by the player who will pick in the next round, if there is one.

Remark that when k is equal to the total number of bags n, then there is no remaining bag and both $v_n(i) = w_n(i) = 0$ for every i.

At state (k, i) for $k \leqslant n-1$, assume without loss of generality that P_1 is playing. P_1 has two choices: the left-most bag or the right-most bag (if $k = n-1$ both are equal and in fact there is no choice). Since i many left-bags have already been picked among k picked bags, the left-most bag is x_{i+1} and the right-most bag is x_{n-k+i}. In the next round, P_1 will have to wait and hence, the maximum possible coins available to P_1 will be $w_{k+1}(i+1)$ if P_1 has chosen the left bag, or $w_{k+1}(i)$ if P_1 has chosen the right bag). Hence, these choices would respectively yield to the following rewards:

$$x_{i+1} + w_{k+1}(i+1), \text{ (if left chosen, } i.e.\ i \to i+1) \text{ or } x_{n-k+i} + w_{k+1}(i) \text{ (if right chosen, } i.e.\ i \to i).$$

Clearly, the optimal choice is the one which gives the largest reward, and therefore:

$$v_k(i) = \max\{x_{i+1} + w_{k+1}(i+1), x_{n-k+i} + w_{k+1}(i)\}, \qquad k = 0, 1, \ldots, n-1,\ i = 0, 1, \ldots, k. \tag{5.6}$$

Note that the case $k = n$ is already solved and is not included in the above recursion.

To complete the description, we have to provide a recursive formula for w as well. With this in mind, let $j^\star_{k,i} = 1$ if the left-most bag is optimal to choose in the above step and $j^\star_{k,i} = 0$ otherwise. As detailed above, we have:

$$v_k(i) = \begin{cases} x_{i+1} + w_{k+1}(i+1), & \text{if } j^\star_{k,i} = 1, \\ x_{n-k+i} + w_{k+1}(i), & \text{if } j^\star_{k,i} = 0. \end{cases}$$

Moreover, the player who will pick in the next round will be in a position with $(k+1)$ bags picked and, among them, $(i + j^\star_{k,i})$ left-most ones. Hence,

$$w_k(i) = v_{k+1}(i + j^\star_{k,i}), \qquad k = 0, 1, \ldots, n-1,\ \ i = 0, 1, \ldots, k.$$

5.2.3 Loot sharing

The "Loot sharing" example is a deterministic game between n players. We consider here the Loot sharing example with five players, ranked with strict order as follows: $p_5 > p_4 > \ldots > p_1$. They want to split 100 non-divisible units amongst them. This is done by voting on a proposal by the top ranked player (here, p_5). If there is a tie, or it is voted down, the proposer is removed from the list (and gets 0) and the remaining highest ranked player makes the next proposal. This is voted on and so on until a split is agreed by the remaining players. We make the assumption that each player is rational (as usual), and that players vote down the proposal if their optimal future share is greater or equal than the offer made in this round. The main question is: what division should the first player propose?

A first approach. If the top player is too greedy and wants all the 100 units, then the others are not getting any and they will vote him down, so the top player will get 0. This is not optimal. With a similar reasoning, if the top player keeps 99 and gives one to one of them, then the other three would vote down. This is still not optimal. Now, suppose the top player keeps 98 and gives one to two other players. Would that be enough?

Solution. We generalize the problem to n players with $n \leqslant 5$, and solve this problem recursively, starting with $n = 1$, then $n = 2$, ..., up to $n = 5$.

1. ($n = 1$) It is clear that the only player takes it all.
2. ($n = 2$) We have a pair of players (p_1, p_2) with $p_2 > p_1$, so p_2 first makes a proposal. If p_2 wants to get any non-zero amount, then p_1 would vote against, leading to a tie. Therefore, the proposer p_2 will be removed with gain 0, and thus p_1 would get all the 100 units. In other words, no matter what the proposal is, p_1 would vote it down and win the whole lot. The resulting sharing outcome is $(100, 0)$.
3. ($n = 3$) We have a set of players (p_1, p_2, p_3) with $p_3 > p_2 > p_1$, so p_3 first makes a proposal. If the outcome of the vote is negative, in the sense that p_3 is removed, then, by the reasoning developed for $n = 2$, p_2 would get 0 in the next round. Therefore, p_2 would be happy to accept any non-zero offer (if he refuses the offer, p_3 is removed and thus p_2 gets 0): for p_2 to accept, p_1 can offer at least 1. Therefore, it is optimal for p_3 to offer one unit to p_2 and take the remaining 99 units. This will be voted up by p_3 and p_2, and will consequently be accepted. So the resulting optimal sharing outcome is $(0, 1, 99)$.
4. ($n = 4$) We have a set of players $p_4 > p_3 > p_2 > p_1$, so p_4 first makes a proposal. If the outcome in this round is negative, then in the next round p_3 will get 99, p_2 will get 1 and p_1 will get 0. The top player p_4 needs at least two other players to support the proposal. The cheapest way of achieving this is to give to two players one unit more than what they would receive in the next rounds and zero to the others. It is clear that the cheapest choice is to give 1 to p_1 (he would accept as otherwise she/he would get nothing in the next round) and 2 to p_2. Then p_4 can give 0 to p_3, who will not vote in favor of the proposal, but the proposal will nevertheless be accepted by a strict majority since it is voted by three players. So the optimal sharing is $(1, 2, 0, 97)$.
5. ($n = 4$) Finally, we can solve the problem when we have a set of 5 players $p_5 > p_4 > p_3 > p_2 > p_1$, and p_5 first makes a proposal. As in the above analysis for $n = 4$, the top player p_1 needs the votes of two other players. The cheapest way of achieving this is to give to two players one unit more than what they would receive in the next rounds and zero to the others. Note that if p_5 is removed, the optimal allocation is $(1, 2, 0, 97, 0)$. Therefore, p_5 should offer 2 to p_1, 1 to p_3, so that they vote in favor, and 0 to p_2 and p_4. So the optimal sharing is $(2, 0, 1, 0, 97)$.

Exercise 5.2 (i) Solve the original problem with six players.
(ii) Solve the problem with a general number of players.
(iii) Solve this problem when the tie-breaker goes to the proposer, *i.e.* when the number of votes in favour is equal (or larger) than the votes against it, the proposal is accepted.

5.2.4 When to stop the coin-flip?

"*When to stop the coin-flip?*" *is a stochastic control problem.* The game starts with zero score. A fair coin is flipped: if the outcome of the coin-flip is heads, the score increases by one, and decreases by one otherwise. The player can stop the game at any time and collects a pre-determined function φ of the score divided by the number of coin-flips. Also, the game stops at a pre-determined deterministic time T. The goal of the player is to maximize the expected value of the collected value.

To model this problem, let $\xi_n \in \{-1,1\}$ be the change in the score after the n-th flip. If the game is stopped right after the n-th flip, the reward is $\varphi(X_n)$ where φ is a kwnon function and

$$X_n = \frac{1}{n}\sum_{t=1}^{n} \xi_t.$$

The goal is to chose a *stopping time* $\tau \in \{1,\dots T\}$ so as to maximize $\mathbb{E}[\varphi(X_\tau)]$. Clearly, this choice of τ can only use the values of ξ's observed up to that time τ, or equivalently X_τ.

The solution of this problem is similar to the one developed in Subsection 5.2.2 for the *Pots of gold* example, as both of them rely on backward induction and decision is binary. We also use similar notations:

- t is the number of coin-flips (or time), *i.e.* $t = 1,2,\dots,T$;
- i is the number of heads up to time t, $i = 1,2,\dots,t$;
- $v_t(i)$ is the maximum expected gains when starting at time t with i heads.

Note that the score at time t with i heads (that means $(t-i)$ many tails), is $i-(t-i) = 2i-t$.

As the game stops at time T, we know that the value at time T with i heads is:

$$v_T(i) \;=\; \varphi((2i-T)/T), \qquad i = 0,1,\dots,T.$$

At any time $t < T$, with $i \leqslant t$ heads, the player has the choice to stop and thus collect $\varphi((2i-t)/t)$. Otherwise, the player can continue, by flipping the coin. Depending on the result of the next coin flip, the number of heads at time $t+1$ will either stay at i or increase to $i+1$, both with probability $1/2$. By definition, the optimal value at time $t+1$ with i heads is $v_{t+1}(i)$, and the optimal value at time $t+1$ with $i+1$ heads is $v_{t+1}(i+1)$. Therefore, the expected value if the player continues is $(v_{t+1}(i)+v_{t+1}(i+1))/2$. Then, the optimal choice is given by the action (continue or stop) yielding to the maximum expected value:

$$v_t(i) = \max\{\varphi((2i-t)/t)\,, \frac{1}{2}\big(v_{t+1}(i)+v_{t+1}(i+1)\big)\}, \quad t = 0,\dots,T-1,\ i = 0,\dots,t.$$

The optimal action is also determined by the maximizer of the above equation.

5.2.5 Random journey in a grid

"Random journey in a grid" is a stochastic control problem. Consider a $n \times m$ planar grid, with a terminal point, and some grid points can be blocked. At any non-terminal and non-blocked points there are 4 possible actions: north, east, south or west. An action is admissible if it keeps the journey in the grid. However, an action is not necessarily successful: it succeeds only with a given probability $p_s \in (0,1)$. The remaining probability $(1-p_s)$ is divided among the *orthogonal* directions that keep the journey in the grid, if there are any. If there is no *orthogonal* direction possible, then the action succeeds with probability 1. The random journey stops when it reaches the terminal point **F**. The goal is to attain the terminal point with the smallest cost possible.

■ **Example 5.2** Consider the example with $n = 5$, $m = 6$ presented in Table 5.1. This Table gives the cost at each point, blocked points, and the terminal point **F**.

In this example, the state space is the set of "accessible" points in the grid, namely

$$\mathscr{X} := \{(i,j),\ i \in \{1,\dots,5\},\ j \in \{1,\dots,6\}\}\backslash\{(2,3),(4,3)\},$$

and the set of actions is $\mathscr{A} := \{\text{north, south, west, east}\}$.

As an example consider the node $(1,1)$. There are two possible actions 'west' and 'south'. If 'west' is chosen, then the action succeeds with probability p_s and the journey moves 'west'. With

	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	1	×	1	1	10
3	1	1	1	1	**F**	1
4	1	1	×	20	1	1
5	1	1	1	1	1	1

Table 5.1: Costs

probability $(1-p_s)$ the action is not successful, and since the only orthogonal allowed direction is 'south', the journey moves south. At node $(3,3)$, only 'west' and 'east' are allowed a nd if any is chosen the journey moves in that direction with probability one, as the orthogonal directions are not allowed. At node $(4,5)$ all four directions are allowed. If 'west' is chosen, then the journey moves 'west' with probability p_s, 'south' with probability $(1-p_s)/2$, or 'north' with probability $(1-p_s)/2$. ∎

For each non-blocked point $x := (i,j)$, $i \in \{1,\dots,n\}$ and $j \in \{1,\dots,m\}$, and an action sequence $\mathbf{a} = (a_0, a_1, \dots)$, let τ be the random time at which the journey ends (the terminal point is reached). The cost of this action sequence is defined as follows:

$$J(x,\mathbf{a}) := \mathbb{E}\Big[\sum_{t=0}^{\tau-1} c_t\Big],$$

where c_t is the cost at time t and it is determined by the grid point where the journey lands at time t. We then define the value function as the minimum expected cost over all possible action sequence:

$$v(x) := \min_{\mathbf{a}} J(x,\mathbf{a}).$$

We set $v(\mathrm{F}) = 0$, the value function at the terminal point. One can then write an equation for the value function based on dynamic programming.

Exercise 5.3 — Additional DPP example: gold mine. Given a gold mine of $n \times m$ dimensions. Each field in this mine contains a positive integer which is the amount of gold. Initially the miner is at first column but can be at any row. He can move only (right, right up, or right down) that is from a given cell, the miner can move to a cell diagonally up towards the right, right, or diagonally down towards the right. Find out maximum amount of gold he can collect. ∎

5.3 Dynamic Programming

The *dynamic programming principle* (DPP) developed by Richard Bellman in the 50s is an effective solution technique for all optimal control problems, as demonstrated in the above examples. We have already used it in the linear quadratic regulator, in Chapter 2. When applicable, this method provides a substantial reduction in the computational complexity. For instance in the above Markov decision process with T many times steps and N many states, the 'brute-force' algorithms would search every possible outcome. As there are N^T many possible paths, the complexity of such algorithm is exponential in the number of time steps. On the contrary, the complexity of dynamic programming is quadratic in the number of time steps.

5.3.1 Infinite Horizon

We first consider a Markov decision problem with infinite time horizon, as formulated in Subsection 5.1.2, with a discount factor $\rho \in (0,1)$. We also assume that the Markov process is time-homogenous

(transition probabilities and costs/reward are independent of time). To simplify here, we always assume that $\rho < 1$. However, in some applications where there is a terminal state (see for example the random journey on a planar grid in Subsection 5.2.5), discount factors greater or equal to one can also be used.

Theorem 5.3.1 — Dynamic programming in infinite horizon. The value function v_∞ of the MDP defined by (5.1) is the unique solution of the following dynamic programming equation:

$$\begin{aligned} v_\infty(x) &= \min_{a\in\mathscr{A}} \big\{ L(x,a) + \rho \mathbb{E}[v_\infty(x_1) \mid x_0 = x] \big\} \\ &= \min_{a\in\mathscr{A}} \left\{ L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a) v_\infty(y) \right\}. \end{aligned} \tag{5.7}$$

Proof. Let $a_0 = a$ and $x_0 = x$. Recall that the action sequence is assumed to be adapted, so, in particular, the action $a_0 = a \in \mathscr{A}$ chosen at time 0 is a function of x_0. Therefore, since at time 0, we know the initial state x, the initial cost $L(x_0, a_0) = L(x,a)$ is deterministic. We can thus write:

$$\begin{aligned} J(x,\mathbf{a}) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \rho^k L(x_k,a_k) \middle| x_0 = x \right] = L(x,a) + \mathbb{E}\left[\sum_{k=1}^{\infty} \rho^k L(x_k,a_k) \middle| x_0 = x \right] \\ &= L(x,a) + \rho \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t L(x_{t+1},a_{t+1}) \middle| x_0 = x \right]. \end{aligned}$$

By denoting $\widehat{\mathbf{a}} = (a_1, a_2, \ldots)$, we have

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t L(x_{t+1},a_{t+1}) \middle| x_0 \right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t L(x_{t+1},a_{t+1}) \middle| x_0, x_1 \right] \middle| x_0 \right] = \mathbb{E}\big[J(x_1,\widehat{\mathbf{a}}) \big| x_0 \big],$$

which implies that

$$J(x,\mathbf{a}) = L(x,a) + \rho \mathbb{E}\big[J(x_1,\widehat{\mathbf{a}}) \big| x_0 \big].$$

Using this formula in the definition of the value function, we obtain the dynamic programming equation:

$$\begin{aligned} v(x) &= \inf_{\mathbf{a}\in\mathscr{A}_d} J(x,\mathbf{a}) = \inf_{a\in\mathscr{A}} \inf_{\widehat{\mathbf{a}}\in\mathscr{A}_d} \big\{ L(x,a) + \rho \mathbb{E}\big[J(x_1,\widehat{\mathbf{a}}) \big| x_0 = x \big] \big\} \\ &= \min_{a\in\mathscr{A}} \big\{ L(x,a) + \rho \inf_{\widehat{\mathbf{a}}\in\mathscr{A}} \mathbb{E}\big[J(x_1,\widehat{\mathbf{a}}) \big| x_0 = x \big] \big\} = \min_{a\in\mathscr{A}} \big\{ L(x,a) + \rho \mathbb{E}[v(x_1) | x_0 = x] \big\}. \end{aligned}$$

■

Theorem 5.3.2 — Optimal feedback controls. Any feedback control $\mathbf{a} = (a_0, a_1, \ldots) \in \mathscr{A}_d$ is optimal if and only if $a_t = a^\star(x_t)$ for all $t \geqslant 0$, for some function $a^\star : \mathscr{X} \mapsto \mathscr{A}$ satisfying:

$$a^\star(x) \in \arg\min_{a\in\mathscr{A}} \left\{ L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a) v(y) \right\}, \qquad \forall x \in \mathscr{X}.$$

Proof. Let $a^\star : \mathscr{X} \mapsto \mathscr{A}$ be a feedback control satisfying

$$a^\star(x) \in \arg\min_{a\in\mathscr{A}} \{ L(x,a) + \rho\, \mathbb{E}[v(x_1) | x_0 = x] \}, \qquad \forall x \in \mathscr{X}.$$

Set $a_k^\star := a^\star(x_k)$. Then, with $x_0 = x$, $a_0^\star = a^\star(x)$, and by dynamic programming equation,

$$v(x) = \min_{a\in\mathscr{A}} \{L(x,a) + \rho\,\mathbb{E}[v(x_1)|x_0 = x]\} = L(x,a_0^\star) + \rho\,\mathbb{E}[v(x_1)|x_0 = x].$$

Again by dynamic programming equation and the definition of $a^\star$,

$$\mathbb{E}[v(x_1)|x_0 = x] = \mathbb{E}\big[L(x_1,a_1^\star) + \rho\mathbb{E}[v(x_2)|x_1]|x_0 = x\big].$$

Hence,

$$v(x) = L(x,a_0^\star) + \mathbb{E}\left(\rho\,L(x_1,a_1^\star) + \rho^2\,\mathbb{E}[v(x_2)]\right).$$

Substitute this identity in the previous equation, to arrive at

$$\begin{aligned} v(x) &= L(x,a_0^\star) + \rho\,\mathbb{E}[v(x_1)|x_0 = x] \\ &= L(x,a_0^\star) + \rho\mathbb{E}\big[L(x_1,a_1^\star)|x_0 = x\big] + \rho^2\mathbb{E}\big[\mathbb{E}[v(x_2)|x_1]|x_0 = x\big] \\ &= L(x,a_0^\star) + \rho\mathbb{E}\big[L(x_1,a_1^\star)|x_0 = x\big] + \rho^2\mathbb{E}\big[v(x_2)|x_0 = x\big]. \end{aligned}$$

By iterating this procedure (mathematical induction), we obtain that for any $m \geqslant 1$,

$$v(x) = \mathbb{E}\left[\sum_{k=0}^{m-1} \rho^k L(x_k,a_k^\star)\middle| x_0 = x\right] + \rho^m\mathbb{E}[v(x_m)|x_0 = x].$$

Recall that $\rho \in (0,1)$, which implies that ρ^m goes to zero as m approaches to infinity. We take the limit as n tends to infinity, to conclude that

$$v(x) := \inf_{\mathbf{a}\in\mathscr{A}_d} J(x,\mathbf{a}) = \mathbb{E}\left[\sum_{k=0}^{m-1} \rho^k L(x_k,a_k^\star)\middle| x_0 = x\right].$$

Hence, adapted control sequence $a^\star \in \mathscr{A}_d$ attains the infimum in the definition of the value function, and is therefore optimal.

To prove the converse, suppose that feedback control $A : \mathscr{X} \mapsto \mathscr{A}$ is optimal, then,

$$v(x) = J(x,A) = L(x,A(x)) + \rho\,\mathbb{E}[J(x_1,A)] \geqslant L(x,A(x)) + \rho\,\mathbb{E}[v(x_1,A)].$$

By dynamic programming, we also have

$$v(x) = \inf_{a\in\mathscr{A}} \{L(x,a) + \rho\,\mathbb{E}[v(x_1)]\} \leqslant L(x,A(x)) + \mathbb{E}[v(x_1,A)].$$

Therefore, for any optimal feedback control,

$$L(x,A(x)) + \rho\,\mathbb{E}[v(x_1,A)] = \inf_{\mathbf{a}\in\mathscr{A}} \{L(x,a) + \rho\,\mathbb{E}[v(x_1)]\}.$$

The above argument also proves uniqueness of the solutions to the dynamic programming equation (5.7). Still we provide another independent proof of uniqueness.

Let u be another solution to (5.7). Then, for any $x \in \mathscr{X}$,

$$\begin{aligned} u(x) - v(x) &= \min_{a\in\mathscr{A}} \left\{ L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a)u(y) \right\} - \min_{a\in\mathscr{A}} \left\{ L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a)v(y) \right\} \\ &\leqslant \rho \max_{a\in\mathscr{A}} \left\{ \sum_{y\in\mathscr{X}} p(x,y,a)\big(u(y) - v(y)\big) \right\} \\ &\leqslant \rho \max_{a\in\mathscr{A}} \left\{ \sum_{y\in\mathscr{X}} p(x,y,a) \max_{y\in\mathscr{X}} \big(u(y) - v(y)\big) \right\} \\ &\leqslant \rho \max_{y\in\mathscr{X}} \big(u(y) - v(y)\big) \max_{a\in\mathscr{A}} \left\{ \sum_{y\in\mathscr{X}} p(x,y,a) \right\} = \rho \max_{y\in\mathscr{X}} \big(u(y) - v(y)\big). \end{aligned}$$

Set $m := \max_{y\in\mathscr{X}}[u(y)-v(y)]$. We now take maximum over x in the above inequality to conclude that, $m \leqslant \rho\, m$. Since $\rho < 1$, this implies that $m \leqslant 0$ or equivalently, $u \leqslant v$. Since the argument is symmetric for u and v, we conclude that $u = v$. ■

The following exercise proves a simple fact that is used in the proof of Theorem 5.3.2 and is also used in the subsequent subsection.

Exercise 5.4 Let $f, g : \mathscr{A} \mapsto \mathbb{R}$ be two given functions. Show that,

$$\min_{a\in\mathscr{A}}[f(a)-g(a)] \leqslant \min_{a\in\mathscr{A}} f(a) - \min_{a\in\mathscr{A}} g(a) \leqslant \max_{a\in\mathscr{A}}[f(a)-g(a)].$$

Also,

$$\min_{a\in\mathscr{A}}[f(a)-g(a)] \leqslant \max_{a\in\mathscr{A}} f(a) - \max_{a\in\mathscr{A}} g(a) \leqslant \max_{a\in\mathscr{A}}[f(a)-g(a)].$$

■

The dynamic programming equation (5.7) is a non-linear equation for the unknown v. Its unique solution is the value function v_∞. As in the LQR framework in Chapter 2, it can be effectively used to compute the value function and the optimal control. We continue by illustrating the dynamic programming approach in the Recycling Robot problem discussed in Example 5.1. In this example, the dynamic programming principle implies that

$$v(h) = \max\{I_w, I_s, I_g\} \text{ and } v(\ell) = \max\{K_g, K_w, K_s\},$$

where

$$\begin{aligned} &I_w := r_w + \rho v(h),\ I_s := r_s + \rho\big(p_h v(h) + (1-p_h)v(l)\big),\ I_g := \rho v(h) \\ &K_g := \rho v(h), \quad K_w := r_w + \rho v(\ell), \quad K_s := p_\ell\big(r_s + \rho v(\ell)\big) + (1-p_\ell)\big(-r_p + \rho\, v(h)\big). \end{aligned}$$

Thus, we have two coupled non-linear equations for the unknowns $v(h)$ and $v(\ell)$. We solve these equations for the $r_s = 2, r_w = 1, r_p = 5, p_h = p_\ell = 1/2, \rho = 1/2$. With these choices,

$$\begin{aligned} &I_w = \frac{1}{2}v(h), \qquad I_s = 2 + \frac{1}{4}\big(v(h)+v(\ell)\big), \\ &K_g = \frac{1}{2}v(h), \qquad K_w = 1 + \frac{1}{2}v(\ell), \qquad K_s = \frac{1}{2}\left(2 + \frac{1}{2}\big(v(h)+v(\ell)\big) - 5\right), \end{aligned}$$

and the equations are

$$\begin{aligned} v(h) &= \max\left\{1 + \frac{1}{2}v(h), 2 + \frac{1}{4}\big(v(h)+v(\ell)\big)\right\}, \\ v(\ell) &= \max\left\{\frac{1}{2}v(h), 1 + \frac{1}{2}v(\ell), \frac{1}{2}\left(2 + \frac{1}{2}\big(v(h)+v(\ell)\big) - 5\right)\right\}. \end{aligned}$$

The solution and the optimal actions are given by,

$$v(h) = \frac{10}{3}, \quad a^\star(h) = \text{search}, \qquad \text{and} \qquad v(\ell) = 2, \quad a^\star(\ell) = \text{wait}.$$

Exercise 5.5 Solve the above problem with parameters

$$r_s = 5, r_w = 1, r_p = 5, p_h = p_\ell = \frac{1}{2},\ \rho = \frac{1}{2}.$$

■

5.3.2 Finite Horizon

In this subsection, we consider the Markov decision problem with a finite time horizon $T > 0$. One can use the same techniques as before to deduce the dynamic programming equation in this framework. More precisely, for any initial data $x_t = x$ and control sequence $\mathbf{a} \in \mathscr{A}_d$, one can write

$$\begin{aligned} J(t,x,\mathbf{a}) &:= \mathbb{E}\Big[\sum_{k=t}^{T-1} L(k,x_k,a_k) + \widehat{L}(x_T)\,\Big|\, x_t = x\Big] \\ &= L(t,x,a_t) + \mathbb{E}\Big[\sum_{k=t+1}^{T-1} L(k,x_k,a_k) + \widehat{L}(x_T)\Big| x_t = x\Big] \\ &= L(t,x,a_t) + \mathbb{E}[J(t+1,x_{t+1},\widehat{\mathbf{a}})|x_t = x], \end{aligned}$$

where $\widehat{\mathbf{a}} = (a_1, a_2, \ldots)$ is as before. We can thus modify the proof for the infinite horizon case using the above identity to prove the dynamic programming in this finite horizon framework. The result is given by the following theorem.

Theorem 5.3.3 — Finite horizon DPP. The value function v_T defined by (5.2) of the MDP with finite horizon is the unique solution of the following dynamic programming equation:

$$\begin{aligned} v_T(t,x) &= \min_{a\in\mathscr{A}} \{L(t,x,a) + \mathbb{E}[v_T(t+1,x_{t+1}) \mid x_t = x]\} \\ &= \min_{a\in\mathscr{A}} \Big\{L(t,x,a) + \sum_{y\in\mathscr{X}} p(t,x,y,a) v_T(t+1,y)\Big\}. \end{aligned}$$

Moreover, the adapted sequence of control $\mathbf{a} := (a_0, a_1, \ldots, a_{T-1}) \in \mathscr{A}_d$ is optimal if and only if $a_t = a^\star(t,x_t)$ for all $t \in \{0,\ldots,T-1\}$, for some function $a^\star$ satisfying,

$$a^\star(t,x) \in \arg\min_{a\in\mathscr{A}} \Big\{L(t,x,a) + \sum_{y\in\mathscr{X}} p(t,x,y,a) v_T(t+1,y)\Big\}, \qquad \forall x \in \mathscr{X},\ t = 0,\ldots,T-1.$$

Note that the dynamic programming equation with finite time horizon T is a recursive equation with terminal condition $v(T,x) = \widehat{L}(x)$. In other words, this dynamic programming equation can be solved backwards in time starting from the final time.

5.4 Algorithms

We discuss two effective algorithms to solve the dynamic programming equations for the infinite horizon Markov decision processes with finite state and action spaces, and $\rho < 1$. In this case, the dynamic programming equation is given by (5.7).

We first define the *Bellman operator* $\mathscr{T}$, which takes real-valued functions on $\mathscr{X}$ to another real-valued functions on $\mathscr{X}$. More precisely, for any $u : \mathscr{X} \longrightarrow \mathbb{R}$, we let

$$\mathscr{T}(u)(x) := \min_{a\in\mathscr{A}} \Big\{L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a) u(y)\Big\}, \qquad x \in \mathscr{X}.$$

Then, the dynamic programming equation (5.7) can be rewritten as

$$v(x) = \mathscr{T}(v)(x) \qquad\qquad x \in \mathscr{X}.$$

Hence, the value function is the *unique fixed point of the nonlinear operator* $\mathscr{T}$.

We continue by studying two algorithms, *value iteration* and *policy iteration*, based on the dynamic programming principle. In addition to being effective computational tools for a given Markov decision process, these algorithms also provide the main motivation for several methods in temporal difference (TD) learning, in particular, $\mathscr{Q}$-learning.

5.4.1 Value Iteration

The procedure for the value iteration algorithm is the following:

(i) start with any non-negative function v_0;
(ii) for $n = 1, 2, \ldots$, recursively define $v_n := \mathscr{T}(v_{n-1})$;
(iii) the value function is given by $v(x) = \lim_{n\to\infty} v_n(x)$.

The facts that the above sequence converges and the limit is equal to the value function are not immediate, the proofs are given below.

We continue with a simple example and provide a proof convergence. The convergence result will then be extended to the general case in Theorem 5.4.1.

■ **Example 5.3 — Value iteration for the Recycling Robot .** In example 5.1, the state space $\mathscr{X}$ has two elements. Hence, functions on $\mathscr{X}$ can be identified by $v = (v(h), v(\ell)) \in \mathbb{R}^2$. Then, $\mathscr{T}(v)$ has also two components given by

$$\mathscr{T}_1(v) = \max\left\{1 + \frac{v(h)}{2}, 1 + \frac{(v(h) + v(\ell))}{4}\right\},$$
$$\mathscr{T}_2(v) = \max\left\{\frac{v(h)}{2}, 1 + \frac{v(\ell)}{2}, -\frac{3}{2} + \frac{(v(h) + v(\ell))}{4}\right\}.$$

Then, starting with any positive $v_0 \in \mathbb{R}^2_+$, the value iteration produces a sequence of pairs v_n:

$$v_{n+1} = (\mathscr{T}_1(v_n), \mathscr{T}_2(v_n)), \qquad n = 0, 1, \ldots.$$

To study the convergence, set

$$m_n := \max\{|w_n(h)|, |w_n(\ell)|\}, \qquad \text{where} \qquad w_n := v_n - .v_{n-1}.$$

By Exercise 5.4, for $n = 1, \ldots$, we have:

$$w_n(h)(h) \leqslant \max\left\{\frac{1}{2} w_n(h), \frac{1}{4}[w_n(h) + w_n(\ell)]\right\} \leqslant \frac{1}{2} m_n,$$
$$-w_{n-1}(h) \leqslant \max\left\{-\frac{1}{2} w_{n-1}(h), -\frac{1}{4}[w_n(h) + w_n(\ell)]\right\} \leqslant \frac{1}{2} m_n.$$

Similarly,

$$w_n(\ell) \leqslant \max\left\{\frac{1}{2}[w_n(h) - v_{n-1}(h)], \frac{1}{2} w_n(\ell), \frac{1}{4}[w_n(h) + w_n(\ell)]\right\} \leqslant \frac{1}{2} m_n.$$

The term $-w_{n-1}(\ell)$ can be bounded by $m_n/2$ as well. Hence, $m_{n+1} \leqslant m_n/2$. By iterating we conclude that $m_{n+1} \leqslant 2^{-n} m_1$. Therefore, the sequence v_n converges to a pair $v^\star$. By passing to the limit in the defining equation of the pair v_n, we conclude that $v^\star$ solves the dynamic programming equation $v^\star = \mathscr{T}(v^\star)$. As the unique solution of this equation is the value function, this implies that $v^\star = (v(h), v(\ell))$. ■

For any $v_0 : \mathscr{X} \mapsto \mathbb{R}_+$, let v_n be defined recursively as above. For $n = 1, 2, \ldots$, set

$$m_n := \max_{x\in\mathscr{X}} |v_n(x) - v_{n-1}(x)|.$$

The general result is the following.

Theorem 5.4.1 — Value Iteration. For any $n \geqslant 0$, $m_{n+1} \leqslant \rho^n m_1$. In particular, v_n converges to the value function v uniformly:

$$\max_{x \in \mathscr{X}} |v_n(x) - v_{n-1}(x)| \leqslant \frac{\rho^n}{1-\rho} m_1.$$

Proof. To prove the previous result, we use the definitions and Exercise 5.4 to obtain the following inequalities for every $x \in \mathscr{X}$ and $n \geqslant 0$:

$$\begin{aligned}
|v_{n+1}(x) - v_n(x)| &= |\mathscr{T}(v_n)(x) - \mathscr{T}(v_{n-1})(x)| \\
&= |\min_{a \in \mathscr{A}} \left\{ L(x,a) + \rho \sum_{y \in \mathscr{X}} p(x,y,a)\, v_n(y) \right\} \\
&\qquad\qquad - \min_{a \in \mathscr{A}} \left\{ L(x,a) + \rho \sum_{y \in \mathscr{X}} p(x,y,a)\, v_{n-1}(y) \right\}| \\
&\leqslant \rho \max_{a \in \mathscr{A}} \left\{ \sum_{y \in \mathscr{X}} p(x,y,a)\, |v_n(y) - v_{n-1}(y)| \right\} \\
&\leqslant \rho \max_{a \in \mathscr{A}} \left\{ \sum_{y \in \mathscr{X}} p(x,y,a)\, m_n \right\} \\
&= \rho\, m_n.
\end{aligned}$$

Hence,

$$m_{n+1}(x) = \max_{x \in \mathscr{X}} |v_{n+1}(x) - v_n(x)| \leqslant \rho m_n.$$

Since the previous inequality is true for all $x \in \mathscr{X}$, we deduce that $m_{n+1} \leqslant \rho\, m_n$. Therefore, $m_{n+1} \leqslant \rho^n m_1$ and m_n converges to zero implying that the sequence of functions v_n converge uniformly to a function $v^\star$. Moreover,

$$v^\star(x) = \lim_{n \to \infty} v_n(x) = \lim_{n \to \infty} \mathscr{T}(v_n)(x) = \mathscr{T}(v^\star)(x).$$

Therefore, $v^\star$ solves the dynamic programming equation. As the value function is the unique solution, $v^\star$ is equal to it.

Finally, for any $x \in \mathscr{X}$ and $n \geqslant 0$, we have

$$\begin{aligned}
|v(x) - v_n(x)| &= \lim_{m \to \infty} |v_m(x) - v_n(x)| = \lim_{m \to \infty} |\sum_{k=n}^{m-1} v_{k+1}(x) - v_k(x)| \\
&\leqslant \lim_{m \to \infty} \sum_{k=n}^{m-1} |v_{k+1}(x) - v_k(x)| \leqslant \lim_{m \to \infty} \sum_{k=n}^{m-1} \rho^{k-n} m_n \\
&= \sum_{k=n}^{\infty} \rho^{k-n} m_n = \frac{1}{1-\rho} m_n \leqslant \frac{\rho^n}{1-\rho} m_1.
\end{aligned}$$

■

The nonlinear operator $\mathscr{T}$ is a *contraction*, i.e., for any two functions $u, w : \mathscr{X} \mapsto \mathbb{R}$.

$$\max_{x \in \mathscr{X}} |\mathscr{T}(u)(x) - \mathscr{T}(w)(x)| \leqslant \rho \max_{x \in \mathscr{X}} |u(x) - w(x)|.$$

One can easily generalize the above proof to show that operators with this property always have a *unique fixed-point*. We continue by formalizing this property for future use. Let $\mathscr{B}$ be the set of of bounded function on $\mathscr{X}$ and let

$$\|u\| := \max_{x\in\mathscr{X}} |u(x)|, \qquad u \in \mathscr{B}.$$

Definition 5.4.1 A map $\mathscr{L} : \mathscr{B} \mapsto \mathscr{B}$ is called a *contraction* on $\mathscr{B}$ if there exist a constant $\lambda < 1$ such that

$$\|\mathscr{L}(u) - \mathscr{L}(w)\| \leqslant \lambda \|u - w\|, \qquad \forall\, u, w \in \mathscr{B}(\mathscr{X}).$$

Let $v_0 \in \mathscr{B}$ be arbitrary and recursively define $v_{n+1} = \mathscr{L}(v_n)$ for $n = 0, 1, \ldots$.

Theorem 5.4.2 Let $\mathscr{L}$ be a contraction on $\mathscr{B}$. Then, there exist a unique fixed-point $v \in \mathscr{B}$ of $\mathscr{L}$. Moreover, for any v_0, the sequence v_n defined above converges to v.

Proof. Let $v_0 \in \mathscr{B}$ be arbitrary and define v_n recursively by, $v_{n+1} = \mathscr{L}(v_n)$ for $n = 0, 1, \ldots$. Then,

$$\|v_{n+1} - v_n\| \leqslant \lambda \|v_n - v_{n-1}\| \leqslant \cdots \leqslant \lambda^n \|v_1 - v_0\|.$$

Hence, $\lim_{n\to\infty} \|v_{n+1} - v_n\| = 0$ and $v := \lim_{n\to\infty} v_n$ exists. Morevover,

$$v = \lim_{n\to\infty} v_{n+1} = \lim_{n\to\infty} \mathscr{L}(v_n) = \mathscr{L}(v).$$

Let $w \in \mathscr{B}$ be another fixed-point of $\mathscr{L}$. Then,

$$\|v - w\| = \|\mathscr{L}(v) - \mathscr{L}(w)\| \leqslant \lambda \|v - w\|.$$

Therefore, $\|v - w\| = 0$ and $v = w$. ■

5.4.2 Policy Iteration

For a feedback action $A : \mathscr{X} \mapsto \mathscr{A}$, let $v(x, A)$ be the cost of this policy starting from the state x. That is,

$$v(x, A) := \mathbb{E}\left[\sum_{k=0}^{\infty} \rho^k L(x_k, A(x_k)) \middle| x_0 = x \right],$$

where the state process $\{x_k\}_{k=0,1\ldots}$ has the transition distribution, $p(x, y, A(x))$:

$$\mathbb{P}(x_{t+1} = y \,|\, x_t = x) = p(x, y, A(x)), \qquad \forall\, x, y \in \mathscr{X},\ t = 0, 1, \ldots.$$

We also let

$$Q(x, a; A) := L(x, a) + \rho \sum_{y\in\mathscr{X}} p(x, y, a)\, v(y, A), \qquad x \in \mathscr{X},\ a \in \mathscr{A}. \tag{5.8}$$

Then,

$$\begin{aligned}
v(x, A) &= L(x, A(x)) + \mathbb{E}\left[\sum_{k=1}^{\infty} \rho^k L(x_k, A(x_k)) \;\middle|\; x_0 = x \right] \\
&= L(x, A(x)) + \rho\, \mathbb{E}\left[\mathbb{E}\left[\sum_{t=0}^{\infty} \rho^T L(x_{t+1}, A(x_{t+1})) \;\middle|\; x_1 \right] \middle|\; x_0 = x \right] \\
&= L(x, A(x)) + \rho\, \mathbb{E}\big[v(x_1, A) \big|\; x_0 = x \big] \\
&= L(x, A(x)) + \rho \sum_{y\in\mathscr{X}} p(x, y, A(x))\, v(y, A) \\
&= Q(x, A(x); A).
\end{aligned}$$

For future reference we record this *linear* equation satisfied by $v(\cdot,A)$:

$$v(x,A) = L(x,A(x)) + \rho \sum_{y\in\mathscr{X}} p(x,y,A(x))\, v(y,A) \;=\; Q(x,A(x);A), \quad \forall\, x\in\mathscr{X}. \tag{5.9}$$

The policy iteration algorithm is the following: The procedure for the value iteration algorithm is the following:

(*i*) start with any feedback action $A_0 : \mathscr{X} \mapsto \mathscr{A}$;

(*ii*) for $n = 1,2,\ldots$: suppose that A_n is computed, then construct A_{n+1} satisfying

$$A_{n+1}(x) \in \arg\min_{a\in\mathscr{A}} Q(x,a;A_n), \quad \forall\, x\in\mathscr{X},$$

(*iii*) stop when $A_{n+1} \equiv A_n$. Then, A_n is optimal and $v(\cdot,A_n)$ is the value function.

We note that by (5.9) and the construction of A_{n+1},

$$\begin{aligned} v(x,A_n) &= L(x,A_n(x)) + \rho \sum_{y\in\mathscr{X}} p(x,y,A_n(x))\, v(y,A_n) \\ &\geqslant \inf_{a\in\mathscr{A}} \left\{ L(x,a) + \rho \sum_{y\in\mathscr{X}} p(x,y,a)\, v(y,A_n) \right\} \\ &= L(x,A_{n+1}(x)) + \rho \sum_{y\in\mathscr{X}} p(x,y,A_{n+1}(x))\, v(y,A_n), \qquad \forall\, x\in\mathscr{X}. \end{aligned} \tag{5.10}$$

Hence, if $A_{n+1} \equiv A_n$, then $v(\cdot,A_n)$ solves the dynamic programming equation (5.7). As the value function is the unique solution to (5.7), this implies that when $A_{n+1} \equiv A_n$, their cost $v(\cdot,A_n)$ is the value function and they are optimal.

We illustrate this procedure again with Example 5.1 and provide a proof convergence. After the example, we prove in Theorem 5.4.3 that the above procedure stops after finitely many steps.

■ **Example 5.4 — Policy iteration for the Recycling Robot.** For Example 5.1, choose

$$A_0(h) = A_0(\ell) = w = \text{‘wait’}.$$

Then, by (5.9),

$$v(h,A_0) = 1 + \frac{1}{2}\, v(h,A_0), \quad v(\ell,A_0) = 1 + \frac{1}{2}\, v(\ell,A_0), \quad \Rightarrow \quad v(h,A_0) = v(\ell,A_0) = 2.$$

Next, we compute $A_1(h), A_1(\ell)$:

$$\begin{aligned} A_1(h) &= \arg\max \{ R(h,s,A_0)\,, R(h,w,A_0) \}, \\ A_1(\ell) &= \arg\max \{ R(\ell,r,A_0)\,, R(\ell,s,A_0)\,, R(\ell,w,A_0) \}, \end{aligned}$$

where,

$$\begin{aligned} R(h,w,A_0) &= 1 + \frac{1}{2}\, v(h,A_0) = 2, \quad R(h,s,A_0) = 2 + \frac{1}{4}\,[v(h,A_0) + v(\ell,\pi_0)] = 3 \\ R(\ell,r,A_0) &= 1 + \frac{1}{2}\, v(h,A_0) = 2, \quad R(\ell,w,A_0) = 1 + \frac{1}{2}\, v(\ell,A_0) = 2, \\ R(\ell,s,A_0) &= -\frac{3}{2} + \frac{1}{4}\,[v(h,A_0) + v(\ell,A_0)] = -\frac{1}{2}. \end{aligned}$$

Hence,

$$A_1(h) = s, \quad A_1(\ell) = w.$$

We compute $v(\cdot,A_1)$ using (5.9). The result is

$$v(h,A_1)=\frac{10}{3},\quad v(\ell,A_1)=2.$$

One can check that

$$A_2(h)=s=\arg\max\{R(h,s,A_1),R(h,w,A_1),$$

and

$$A_2(\ell)=w=\arg\max\{R(\ell,r,A_1),R(\ell,s,A_1),R(\ell,w,A_1)\}.$$

Hence, $A_2\equiv A_1$ and therefore, A_1 is optimal and the value is $v(h)=10/3$, $v(\ell)=2$.

■

We conclude this chapter with the general result on the convergence of the policy iteration algorithm.

Theorem 5.4.3 Assume $\mathscr{X}$ and $\mathscr{A}$ are finite. The sequence $v(\cdot,A_n)$ constructed by the policy iteration is monotone, and after a finite number steps, we have $v(\cdot,A_n)=v(\cdot,A_{n-1})$ and the algorithm stops.

Proof. Set

$$k_n:=\min_{x\in\mathscr{X}}[v(x,A_n)-v(x,A_{n+1})].$$

By (5.10),

$$v(x,A_n)\geqslant L(x,A_{n+1}(x))+\rho\sum_{y\in\mathscr{X}}p(x,y,A_{n+1}(x))\,v(y,A_n).$$

Also by (5.9) used for $A=A_{n+1}$,

$$v(x,A_{n+1})=L(x,A_{n+1}(x))+\rho\sum_{y\in\mathscr{X}}p(x,y,A_{n+1}(x)\,v(y,A_{n+1})$$

We subtract the above two to obtain,

$$[v(x,A_n)-v(x,A_{n+1})]\geqslant\rho\sum_{y\in\mathscr{X}}p(x,y,A_{n+1}(x))\,[v(y,A_n)-v(y,A_{n+1})]\geqslant\rho\,k_n,\qquad x\in\mathscr{X}.$$

By taking minimum over $x\in\mathscr{X}$, we conclude that $k_n\geqslant\rho\,k_n$. Since $\rho<1$, this implies that $k_n\geqslant 0$. hence, the sequence $v(\cdot,A_n)$ is decreasing in n. As there are finitely many different possible policies, this implies that in a finite number of steps, the algorithm stops. As argued before, when this happens, A_n is optimal. ■

5.5 Exercises

Problem 5.1 This is another knapsack problem. Suppose that a non-empty set

$$A=\{a_1=(c_1,w_1),\ldots,a_n=(a_n,w_n)\},$$

of items is given. Each item $a_i\in A$ has a value $c_i>0$ and a weight $w_i>0$. We are also given a weight limit of $W\geqslant 0$ and our goal is to choose a *portion*, x_i of each item so as to maximize the total value while remaining within the weight limit. Mathematically, set

$$\mathscr{Q}_n:=\{x=\{x_1,\ldots,x_n\}\in[0,1]^n\}.$$

For any set portions $x \in \mathscr{Q}_n$, let

$$v(x) := \sum_{i=1}^{n} x_i c_i$$

be its value, and let

$$w(x) := \sum_{i=1}^{n} x_i w_i$$

be its weight. Then, the problem is to maximize $v(x)$ over all $x \in \Sigma_{n-1}$ satisfying the weight limit $w(x) \leqslant W$.

For $k = 1, \ldots, n$ and $w \in [0, W]$, let $v_k(w)$ be the optimal value with weight w (it does not have to be an integer) and using the first k elements, i.e, using the set $A_k = \{x_1, \ldots, a_k\}$.

a. Compute $v_1(w)$.

b. Show that for any $k = 2, \ldots, n$,

$$v_k(w) = \max \left\{ v_{k-1}(w - xc_k) + xc_k \;\middle|\; 0 \leqslant x \leqslant \min(1, \frac{w}{w_k}) \right\}$$

c. Describe an algorithm to compute the original problem, which is $v_n(W)$, using the above recursion.

Problem 5.2 This is another variation of the knapsack problem. It is the same as the above problem but the portions are restricted to stay in the set

$$\Sigma_{n-1} := \{x = \{x_1, \ldots, x_n\} \in [0,1]^n : x_1 + \ldots + x_n = 1\}.$$

a. Compute $v_1(w)$.

b. Show that for any $k = 1, \ldots, n-1$,

$$v_k(w) = \max \left\{ L_{k_1}(x; w) \;\middle|\; 0 \leqslant x \leqslant \min(1, \frac{w}{w_k}) \right\}$$

where

$$L_{k_1}(x; w) = \begin{cases} c_k, & \text{if} \quad x = 1, \\ (1-x) v_k \left(\dfrac{w - x w_{k+1}}{1-x} \right) + x c_k, & \text{if} \quad x < 1. \end{cases}$$

c. Describe an algorithm to compute the original problem, which is $v_n(W)$, using the above recursion.

Problem 5.3 There are strictly ordered six players $p_6 > p_5 > \ldots > p_1$. They want to split 100 non-divisible units amongst them. This is done by voting on a proposal by the top ranked player. If it is voted up or if there is a tie, the proposal is accepted. If, on the other hand, if it is strictly voted down, the proposer is removed from the list (and gets 0) and the remaining highest ranked player makes the next proposal. This is voted on and so on until a split is agreed by the remaining players. What split should the first player propose?

We make several assumptions:

- Each player is rational and there are no cliques;
- In each round players would vote down the proposal if their optimal future share is greater or equal than the offer made in this round.

Solve this problem by *dynamic programming*.

Problem 5.4 Suppose that $\{\xi_1, \xi_2, \ldots\}$ is an i.i.d. sequence of random variables with values $\{0,4\}$ each having equal probability. For control variables $c_t \in [0,1]$, and initial data $X_0 = y \in \mathbb{R}$, let Y_t be the solution of

$$X_{t+1} = X_t + \ln(c_t) + \xi_{t+1}, \qquad t = 1,2,\ldots$$

We choose the controls $\mathbf{c} := (c_1, c_2, \ldots)$ so as to

$$\text{maximize } v(x,\mathbf{c}) := E\left[\sum_{t=0}^{\infty} \rho^t \left[\ln(1-c_t) + X_t\right] \middle| X_0 = x\right], \qquad yx \in \mathbb{R}.$$

Let,

$$v(x) := \min_{\mathbf{c}} v(x,\mathbf{c}).$$

a. Write down the dynamic programming equation for $v(y)$.

b. We postulate that the value function has the form

$$v(x) = \frac{x+b}{1-\rho},$$

for some constant b. Use this Ansatz and the dynamic programming equation to compute b in terms of ρ.

c. Show that the optimal feedback is the constant $c_t^* \equiv \rho$.

Remark. Above is related to an optimal consumption problem with logarithmic utility. Here $(1-c_t)$ is the fraction of wealth chosen for consumption and X_t is the natural logarithm of the wealth of the agent.

6. Control with Learning

In this chapter, we provide several popular learning algorithms. The ones that we discuss are based on dynamic programming, and on the value and policy iteration. The textbook *Reinforcement Learning: An Introduction* by Richard S. Sutton and Andrew G. Baroto is an excellent source for these. It also has many excellent examples.

6.1 Multi-armed bandit problems

In this classical problem, a decision maker should choose one arm between K alternatives at each time t. Once choice $a_t \in \{1,\dots,K\}$ is made, a random reward r_t is given. This random variable is independently chosen and it distribution depends only on a_t. There are several possible objective function for this problem, we consider here the maximisation of the long-term average of the returns, defined as follows:

$$J(\mathbf{a}) := \limsup_{T\to\infty} J_T(\mathbf{a}), \quad \text{with } J_T(\mathbf{a}) := \frac{1}{T}\sum_{t=1}^{T} r_t, \quad \text{for an action sequence } \mathbf{a} := (a_1, a_2, \dots).$$

Let $T \in \mathbb{N}^\star$, and fix an action sequence $\mathbf{a} := (a_1, a_2, \dots)$. For all possible alternative $i \in \{1,\dots,K\}$, we define the number of times the action i is chosen up to time T:

$$N_T(\mathbf{a}, i) := \sum_{t=1}^{T} \mathbb{1}_{a_t=i},$$

where $\mathbb{1}_{a_t=i}$ is the indicator function, *i.e.* it is one if $a_t = i$ and zero otherwise. We also define $\lambda_T(\mathbf{a}, i) := N_T(\mathbf{a}, i)/T$. It is clear that for all $T \in \mathbb{N}^\star$,

$$\sum_{i=1}^{K} \lambda_T(\mathbf{a}, i) = \frac{1}{T}\sum_{i=1}^{K} N_T(\mathbf{a}, i) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{K} \mathbb{1}_{a_t=i} = 1.$$

Finally, for all $i \in \{1,\dots,K\}$, define

$$J_T(\mathbf{a}, i) := \frac{1}{N_T(\mathbf{a}, i)} \sum_{t=1}^{T} r_t \mathbb{1}_{a_t=i}, \quad \text{if } N_T(\mathbf{a}, i) \neq 0,$$

and $J_T(\mathbf{a},i) = 0$ otherwise. This quantity represents the average of the rewards obtained when playing action i. Then, we can write

$$J_T(\mathbf{a}) := \frac{1}{T}\sum_{t=1}^{T} r_t = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{K} r_t \mathbb{1}_{a_t=i} = \frac{1}{T}\sum_{i=1}^{K} N_T(\mathbf{a},i) J_T(\mathbf{a},i) = \sum_{i=1}^{K} \lambda_T(\mathbf{a},i) J_T(\mathbf{a},i).$$

On the other hand, if the random variable $N_T(\mathbf{a},i)$ converges to infinity with probability one, then by the law of large number, we also have the following convergence with probability one:

$$\lim_{T\to\infty} J_T(\mathbf{a},i) = m(i) := \mathbb{E}[R_i],$$

where the random variable R_i represents the random reward given by arm i. Finally, let $a^\star$ be a maximizer of $m(\cdot)$, *i.e.* the arm with the best expected reward. Then,

$$\begin{aligned} J_T(\mathbf{a}) &= \sum_{i=1}^{K} \lambda_T(\mathbf{a},i) J_T(\mathbf{a},i) = \sum_{i=1}^{K} \lambda_T(\mathbf{a},i) m(i) + \sum_{i=1}^{K} \lambda_T(\mathbf{a},i)\big(J_T(\mathbf{a},i) - m(i)\big) \\ &\leqslant m(a^\star) + \sum_{i=1}^{K} \lambda_T(\mathbf{a},i)\big(J_T(\mathbf{a},i) - m(i)\big), \end{aligned}$$

and therefore,

$$J(\mathbf{a}) := \limsup_{T\to\infty} J_T(\mathbf{a}) \leqslant m(a^\star) + \limsup_{T\to\infty} \sum_{i=1}^{K} \lambda_T(\mathbf{a},i)\big(J_T(\mathbf{a},i) - m(i)\big).$$

As mentioned above, if $N_T(\mathbf{a},i)$ converges to infinity, then by the law of large number, $J_T(\mathbf{a},i) - m(i)$ converges to 0. Otherwise, if $N_T(\mathbf{a},i)$ does not converge to 0, since T goes to infinity, we necessarily have that $\lambda_T(i)$ converges to zero. Therefore, in both cases, the sum converges to 0 at the limit, and we can conclude that for all action sequence $\mathbf{a} := (a_1, a_2, \ldots)$, $J(\mathbf{a}) \leqslant m(a^\star)$, where $m(a^\star)$ is the expected reward of the best arm.

In other words, if we know the expected values $m(\cdot)$, we would simply choose $a^\star$ at every time and obtain the maximum average return. The problem is interesting when the distribution of the reward are unknown and has to be learned. In this case, it is clear that one has to randomly explore all alternatives in order to estimate the law of the reward, and maximize the total reward. This can be done through the use of randomized controls/actions, introduced in the following section.

6.1.1 Randomized policy

To define the notion of randomized actions or controls, we consider the general Markov Decision Processes framework, detailed in Chapter 5. In particular, the finite state and action spaces are respectively defined by $\mathscr{X} = \{x^{(1)}, \ldots, x^{(M)}\}$ and and $\mathscr{A} = \{a^{(1)}, \ldots, a^{(K)}\}$, for $M, K \in \mathbb{N}^\star$, and we denote by

$$p(x,y,a) = \mathbb{P}(x_{t+1} = y | x = x, a_t = a),$$

the transition probability from the state $x \in \mathscr{X}$ to a new state $y \in \mathscr{X}$ at time $t \geqslant 0$ when choosing action $a \in \mathscr{A}$.

A *randomized policy* is a map from the state space $\mathscr{X}$ into probabilities on $\mathscr{A}$. As $\mathscr{A}$ has K many elements, probabilities on $\mathscr{A}$ take values in the simplex

$$\Delta_K := \{p = (p_1, \ldots, p_K) \in [0,1]^K \text{ s.t. } p_1 + \ldots + p_K = 1\}.$$

One may also see a randomized policy π as a function of the state-action pair into the unit interval, *i.e.* $\pi : \mathscr{X} \times \mathscr{A} \longrightarrow [0,1]$. With this in mind, the randomized policy π defined the probability of choosing action $a \in \mathscr{A}$ when at state $x \in \mathscr{X}$, *i.e.* $\pi(x,a) = \mathbb{P}(a_t = a \mid x_t = x)$ for all $t \geqslant 0$.

Recall that previously, for a sequence of action $\mathbf{a} := (a_1, a_2, \ldots)$, the Markov Decision Process was characterized by the transition probabilities $p(x,y,a)$ for $(x,y,a) \in \mathscr{X} \times \mathscr{X} \times \mathscr{A}$. Now, the Markov Decision process x^π given a randomized policy π is actually a simple Markov process (not controlled) characterized by the following transition probabilities:

$$\begin{aligned} p_\pi(x,y) &:= \mathbb{P}(x_{t+1}^\pi = y \mid x_t^\pi = x) = \sum_{a \in \mathscr{A}} \mathbb{P}(x_{t+1}^\pi = y \mid x^\pi = x, a_t = a) \mathbb{P}(a_t = a \mid x^\pi = x) \\ &= \sum_{a \in \mathscr{A}} p(x,y,a)\pi(x,a). \end{aligned}$$

Note that here, we defined the randomized policy as a *stationary policy*, in the sense that it does not depend on time $t \in \mathbb{N}^\star$. In other words, we use the same randomization at any time of action. We could also define time-dependent randomized policy, which can for example be useful for the multi-armed bandit problem. Indeed, in this problem, the estimate of the rewards change over time thanks to exploration, and the optimal policy should take that into account (see section below).

6.1.2 Exploration and exploitation: ε-greedy policies

In this section, we use the concept of randomized actions in the armed bandit problem. More precisely here, we want first to estimate the expected rewards $m(\cdot)$ by *exploration*, and then *exploit* these estimates to maximize the average returns.

With this in mind, we first need to construct an estimation process for the expectation of each arm $i \in \{1, \ldots, K\}$. Let $T \in \mathbb{N}^\star$, and fix an action sequence $\mathbf{a} := (a_1, a_2, \ldots)$. At each time $T \in \mathbb{N}^\star$, we denote by $\widehat{m}_T(i)$ the estimated expected reward of arm i. With the above notations, we let $\widehat{m}_T(i) = 0$ when $N_T(\mathbf{a}, i) = 0$ (if we did not try arm i yet), and otherwise, as soon as $N_T(\mathbf{a}, i) > 0$, we set

$$\widehat{m}_T(i) := J_T(\mathbf{a}, i) := \frac{1}{N_T(\mathbf{a}, i)} \sum_{t=1}^{T} r_t \mathbb{1}_{a_t = i}.$$

Then, at each time $T \in \mathbb{N}^\star$, we compute $a_T^\star$ the smallest maximizer of the estimates $\widehat{m}_T(\cdot)$. Note that choosing the smallest maximizer is rather arbitrary, in order to ensure that $a_T^\star$ is uniquely defined. Other choices, such as the largest, would of course work exactly the same way.

We then choose a small parameter $\varepsilon > 0$ and define a randomized (time-dependent) policy π_T at each time $T \in \mathbb{N}^\star$ by:

$$\pi_T(i) = \begin{cases} \frac{\varepsilon}{K-1}, & \text{if } i \neq a_T^\star, \\ 1 - \varepsilon, & \text{if } i = a_T^\star. \end{cases}$$

This policy is in fact exactly the one we introduced in Chapter 1, when introducing the ε-greedy algorithm for the multi-armed bandit problem: with probability $1 - \varepsilon$, the "optimal" (so far) action $a^\star$ is chosen, and with probability ε, the algorithm explores the other $K - 1$ alternatives with uniform probability $\varepsilon/(K-1)$. One can remark that this randomized policy is time-dependent, but is independent of the current state, or equivalently of the choices made at previous states.

The following result states that this policy is almost optimal. In fact, this policy will be optimal if we let the parameter ε tend to 0 in time.

Theorem 6.1.1 The average reward obtained with the above ε-greedy algorithm satisfies the following inequality:

$$J(\pi) := \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t \geqslant (1-\varepsilon)m(a^\star) + \frac{\varepsilon}{K-1}\sum_{i\neq a^\star} m(i),$$

where $a^\star$ is the smallest maximizer of $m(\cdot)$.

This above type of policies are called ε-greedy. The general definition for a finite action space $\mathscr{A}$ with K element is given below.

Definition 6.1.1 Let $G : \mathscr{A} \mapsto \mathbb{R}$ be a ranking function, and let $a^\star$ be a uniquely defined maximizer of G, i.e.

$$G(a^\star) = \max_{a\in\mathscr{A}} G(a).$$

The ε-greedy (randomized) policy derived from the ranking function G is given by

$$\pi(a) = \begin{cases} \frac{\varepsilon}{K-1}, & \text{if } a \neq a^\star, \\ 1-\varepsilon, & \text{if } a = a^\star. \end{cases}$$

In the multi-armed bandit problem, the ranking function is defined as the current estimate $\widehat{m}_T(\cdot)$ of the reward expectation of each arm. In general, the Greedy policy *exploits* the ranking given by the function G, but still *explores* the environment by randomly trying other actions which are not ranked highly by G. The exploration is essential for learning, while the exploitation is of course essential to maximize the reward.

6.1.3 Estimation

We close this section by providing a recursive formula for the estimation function $\widehat{m}_T$ in the multi-armed bandit problem. Recall that for each arm $i \in \{1,\dots,k\}$, the estimation is defined at each time $T \in \mathbb{N}^\star$ by

$$\widehat{m}_T(i) := \frac{1}{N_T(\mathbf{a},i)}\sum_{t=1}^{T} r_t \mathbb{1}_{a_t=i},$$

if $N_T(\mathbf{a},i) \neq 0$, and 0 otherwise. If action i is chosen at time $T+1$, we can update this estimation by computing:

$$\begin{aligned}\widehat{m}_{T+1}(i) &:= \frac{1}{N_{T+1}(\mathbf{a},i)}\sum_{t=1}^{T+1} r_t \mathbb{1}_{a_t=i} = \frac{1}{N_T(\mathbf{a},i)+1}\left(r_{T+1} + \sum_{t=1}^{T} r_t \mathbb{1}_{a_t=i}\right) \\ &= \frac{1}{N_T(\mathbf{a},i)+1}\big(r_{T+1} + N_T(\mathbf{a},i)\widehat{m}_T(i)\big).\end{aligned}$$

Alternatively, we can write:

$$\widehat{m}_{T+1}(i) = \widehat{m}_T(i) + \frac{1}{N_T(\mathbf{a},i)+1}\big(r_{T+1} - \widehat{m}_T(i)\big) \tag{6.1}$$

In other words, the new estimate $\widehat{m}_{T+1}(i)$ is given by the sum of the previous estimate $\widehat{m}_T(i)$ and the learning rate (here $\frac{1}{N_T(\mathbf{a},i)+1}$) times the temporal difference (TD), defined as the difference between the target (here r_{T+1}) and the previous estimate. This notion of temporal difference will be further investigated in the next section. Note that the recursive formula (6.1) satisfied by the estimate share some similarities with the formula obtained in the Kalman filter procedure detailed in Chapter 4.

6.2 Algorithms

6.2.1 Temporal-difference learning – TD(0)

In this section, we introduce an algorithm which was precursor to many approaches in reinforcement learning, in particular to $\mathscr{Q}$-Learning. It is the first one among all Temporal Difference (TD) Learning algorithms and thus called TD(0).

We consider a MDP with finite state and action spaces, respectively denoted $\mathscr{X}$ and $\mathscr{A}$. Let π be a *stationary* (time-independent) randomized policy, and denote x^π be the corresponding homogenous Markov process. Recall that the transition probabilities of the Markov process x^π are given by:

$$p(x,y) := \mathbb{P}(x_{t+1}^\pi = y \mid x_t^\pi = x) = \sum_{a\in\mathscr{A}} p(x,y,a)\pi(x,a), \tag{6.2}$$

where

$$p(x,y,a) = \mathbb{P}(x_{t+1} = y | x = x, a_t = a),$$

is the "initial" transition probability from the state $x \in \mathscr{X}$ to a new state $y \in \mathscr{X}$ when choosing action $a \in \mathscr{A}$.

We then consider the following infinite-horizon discounted reward, when starting at state $x \in \mathscr{X}$ and using a randomized policy π:

$$v^\pi(x) := \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t r\left(x_t^\pi, x_{t+1}^\pi, a_t\right) \middle| x_0^\pi = x\right], \tag{6.3}$$

where $r(x,y,a)$ is the reward obtained when moving from the state $x \in \mathscr{X}$ to a new state $y \in \mathscr{X}$ by choosing action $a \in \mathscr{A}$.

As in the value iteration algorithm described in Chapter 5, the main point is to show that the value function defined by (6.3) is the fixed point of an operator to be characterized. With this in mind, we compute for any $x \in \mathscr{X}$:

$$\begin{aligned}
v^\pi(x) &= \mathbb{E}\left[\mathbb{E}\left[\sum_{t=0}^{\infty} \rho^t r\left(x_t^\pi, x_{t+1}^\pi, a_t\right) \middle| x_1^\pi\right] \middle| x_0^\pi = x\right] \\
&= \mathbb{E}\left[\rho^0 r\left(x_0^\pi, x_1^\pi, a_0\right) + \mathbb{E}\left[\sum_{t=1}^{\infty} \rho^t r\left(x_t^\pi, x_{t+1}^\pi, a_t\right) \middle| x_1^\pi\right] \middle| x_0^\pi = x\right] \\
&= \mathbb{E}\left[r\left(x, x_1^\pi, a_0\right) + \mathbb{E}\left[\sum_{t=0}^{\infty} \rho^{t+1} r\left(x_{t+1}^\pi, x_{t+2}^\pi, a_{t+1}\right) \middle| x_1^\pi\right] \middle| x_0^\pi = x\right] \\
&= \mathbb{E}\left[r\left(x, x_1^\pi, a_0\right) + \rho v^\pi(x_1^\pi) \mid x_0^\pi = x\right].
\end{aligned}$$

Remark that this is similar to a dynamic programming principle, except that here, we are not considering the value function, but only the total reward (not the maximization or minimization problem). We can continue by writing the previous expectation as follows, using (6.2):

$$v^\pi(x) = \sum_{y\in\mathscr{X}} p(x,y)\left(r\left(x,y,a_0\right) + \rho v^\pi(y)\right) = \sum_{y\in\mathscr{X}}\sum_{a\in\mathscr{A}} p(x,y,a)\pi(x,a)\left(r(x,y,a) + \rho v^\pi(y)\right). \tag{6.4}$$

Therefore, the function v^π can be seen as the fixed-point of the linear operator $\mathscr{T}_\pi$, *i.e.* $v^\pi = \mathscr{T}_\pi v^\pi$, where $\mathscr{T}_\pi$ is defined for all functions $u : \mathscr{X} \mapsto \mathbb{R}$ as follows:

$$\mathscr{T}_\pi u(x) := \sum_{y\in\mathscr{X}}\sum_{a\in\mathscr{A}} p(x,y,a)\pi(x,a)\left(r(x,y,a) + \rho u(y)\right),\ x \in \mathscr{X}.$$

When all functions are known, this linear equation can be solved easily, as the linear map $\mathscr{T}_\pi$ is a contraction (see Chapter 5). However, in a learning environment one want to learn or estimate v^π from the observed rewards. More precisely, we assume that we do not know the reward function $r(x,y,a_0)$, we can only observe the reward obtained through time. Therefore, given the transition probability matrix, a fixed randomized policy π, and the observed reward, we want to estimate v^π.

The procedure of the TD(0) algorithm to estimate v^π is the following.

TD(0) algorithm to estimate v^π.

1. Initialization: arbitrarily choose $v_0^\pi : \mathscr{X} \mapsto \mathbb{R}$ and $x_0 \in \mathscr{X}$.
2. For $n = 0, 1, \ldots$, suppose that v_n^π and x_n are known (constructed at the previous step).
   (a) Choose an action a_n according to the (fixed) randomized policy $\pi(x_n, \cdot)$.
   (b) Choose a new state x_{n+1} according to the (known) transition probability $p(x_n, \cdot, a_n)$.
   (c) Given a sequence of learning rate $\alpha := (\alpha_1, \alpha_2, \ldots)$, update the estimate for v_n^π by:

$$v_{n+1}^\pi(x_n) = v_n^\pi(x_n) + \alpha_n\big(r(x_n, x_{n+1}, a_n) + \rho v_n^\pi(x_{n+1}) - v_n^\pi(x_n)\big). \tag{6.5}$$

3. Stop the recursion when the difference $|v_{n+1}^\pi - v_n^\pi|$ is sufficiently small. In this case, v_{n+1}^π is an approximation of v^π.

As mentioned in the previous description of the TD(0) algorithm, we need to choose the sequence of learning rate $\alpha := (\alpha_1, \alpha_2, \ldots)$. This sequence must satisfy the following properties, for all $n \in \mathbb{N}^\star$:

$$\alpha_n \in (0,1), \sum_n \alpha_n = \infty, \text{ and } \sum_n \alpha_n^2 < \infty. \tag{6.6}$$

Remark that the above updating equation has exactly same form as the updating rule (6.1) in the multi-armed bandit problem. Indeed, here we have:

$$\text{New Estimate} = v_{n+1}^\pi(x_n), \quad \text{Old Estimate} = v_n^\pi(x_n), \quad \text{Learning rate} =: \alpha_n,$$

and

$$\text{Temporal Difference} = r(x_n, x_{n+1}, a_n) + \rho v_n(x_{n+1}) - v_n(x_n).$$

Finally, one could use the usual supremum norm for the notion of 'small', *i.e.* stopping if

$$\|v_{n+1}^\pi - v_n^\pi\| := \max_{y \in \mathscr{X}} \{|v_{n+1}^\pi(y) - v_n^\pi(y)|\}$$

is smaller than a pre-determined threshold.

■ **Example 6.1 — Evaluating Recycling Robot.** Consider the recycling robot problem, and assume that the expected number of can that can be collected in each mode is unknown, *i.e.* the rewards r_w, r_s are unknown. We suggest the following randomized policy:

$$\pi(h,s) = \frac{3}{4}, \quad \pi(h,w) = \frac{1}{4}, \quad \pi(\ell,s) = \frac{2}{5}, \quad \pi(\ell,w) = \frac{2}{5}, \quad \pi(\ell,r) = \frac{1}{5}.$$

One can evaluate the reward obtained through this policy by using the TD(0) algorithm. A python code will be provided on Canvas. ■

Theorem 6.2.1 Suppose that the Markov process x^π visits each state with non-zero probability (i.e., it is recurrent) and the learning rate satisfies (6.6). Then, the sequence of estimating functions v_n^π converges to v^π.

The proof of this theorem is omitted here, but we will prove convergence in the following very specific examples to provide some insight.

■ **Example 6.2** Suppose that the rewards are deterministic and constant equal to $r_\star \in \mathbb{R}$. In particular, they are independent of time, state and action. In this case, it is clear that the total reward v^π is equal to $r_\star/(1-\rho)$. In this case, we can easily show that regardless the initialization v_0^π, the deterministic sequence v_n^π converges.

To show that, we define $w_n := v_n^\pi - r_\star/(1-\rho)$ for all n, where v_n^π is constructed recursively by the TD(0) algorithm through equation (6.5). We want to show that the sequence w_n converges to 0. First, using that v_n^π is constant in (6.5), we obtain:

$$v_{n+1}^\pi = v_n^\pi + \alpha_n\big(r_\star + \rho v_n^\pi - v_n^\pi\big) = \beta_n v_n^\pi + \alpha_n r_\star, \quad \text{where} \quad \beta_n = 1 - \alpha_n(1-\rho).$$

Replacing in the definition of w_n, we then obtain:

$$\begin{aligned} w_{n+1} := v_{n+1}^\pi - \frac{r_\star}{1-\rho} &= \beta_n v_n^\pi + \alpha_n r_\star - \frac{r_\star}{1-\rho} = \beta_n v_n^\pi + \big(\alpha_n(1-\rho) - 1\big)\frac{r_\star}{1-\rho} = \beta_n v_n^\pi - \beta_n \frac{r_\star}{1-\rho} \\ &= \beta_n w_n. \end{aligned}$$

By mathematical induction, one can therefore prove that $w_{n+1} = \beta_n\beta_{n-1}\cdots\beta_0 w_0$. Since for all n, $\beta_n \in (0,1)$, this shows that w_n converges to zero. ■

■ **Example 6.3** Let us now consider a slightly more interesting case, where the rewards r_t are an i.i.d. sequence with $\mathbb{E}[r_t] = r_\star$. Again, it is clear that v^π is scalar (independent of the state) and is also equal to $r_\star/(1-\rho)$. To prove Theorem 6.2.1 in this case, we define:

$$m_n := \mathbb{E}[v_n^\pi],\ \sigma_n^2 := \mathbb{V}\mathrm{ar}(v_n^\pi), \qquad \text{and} \qquad r_\star := \mathbb{E}[r_t],\ \sigma_\star^2 := var(r_t).$$

Then, using (6.5) and taking expectation, we have:

$$m_{n+1} = m_n + \alpha_n\big(r_\star + \rho\, m_n - m_n\big).$$

This is exactly the sequence we studied in the previous example. Therefore, m_n converges to $r_\star/(1-\rho)$. Moreover, using the fact that v_n^π is independent of r_n in the equation $v_{n+1}^\pi = \beta_n v_n^\pi + \alpha_n r_n$, one can compute the variance:

$$\sigma_{n+1}^2 := \mathbb{V}\mathrm{ar}(v_{n+1}^\pi) = \mathbb{V}\mathrm{ar}\big(\beta_n v_n^\pi + \alpha_n r_n\big) = \beta_n^2\sigma_n^2 + \alpha_n^2\sigma_\star^2.$$

Set

$$B_n := \prod_{k=1}^{n} \beta_k^2.$$

Then, for any $n \leqslant m$,

$$\sigma_{n+m+1}^2 = \left(\sum_{k=0}^{m} \frac{B_{n+m+1}}{B_{n+k}}\alpha_{n+k}^2\right)\sigma_\star^2 + \frac{B_{n+m+1}}{B_n}\sigma_n^2.$$

We can show as in the previous example that

$$\lim_{n\to\infty}\lim_{m\to\infty}\left(\frac{B_{n+m+1}}{B_n}\right) = 0, \qquad \Rightarrow \qquad \lim_{t\to\infty}\sigma_t^2 = \lim_{n\to\infty}\lim_{m\to\infty}\left(\sum_{k=0}^{m}\frac{B_{n+m+1}}{B_{n+k}}\alpha_{n+k}^2\right)\sigma_\star^2$$

Finally, since $\beta_n < 1$ for each n, and α^2 is assumed to be summable,

$$\lim_{n\to\infty}\lim_{m\to\infty}\left(\sum_{k=0}^{m}\frac{B_{n+m+1}}{B_{n+k}}\alpha_{n+k}^2\right)\leqslant\lim_{n\to\infty}\lim_{m\to\infty}\left(\sum_{k=0}^{m}\alpha_{n+k}^2\right)=0.$$

■

6.2.2 $\mathscr{Q}$-learning: off-policy TD algorithm

The $\mathscr{Q}$-learning algorithm, also called the *off-policy TD algorithm*, is similar to the TD(0) algorithm discussed above, but the goal is now to compute the value function

$$v^\star=\inf_\pi v^\pi,$$

i.e. the minimal cost for the optimal randomized policy, whereas the TD(0) algorithm only computes the cost v^π for a given policy.

More importantly, the $\mathscr{Q}$-algorithm computes a function, called the $\mathscr{Q}$ function of the state-action pair. This function represents the current estimate of the "quality" of using a given action at a given state. Therefore, with this $\mathscr{Q}$ function, the decision maker does not have to wait until the end of the process to evaluate the decisions. Therefore, from the learning perspective, this algorithm is much more than just an algorithm to compute the value function.

More precisely, once a $\mathscr{Q}$ function (or $\mathscr{Q}$ table as it is sometimes called) is given, at the state x the controller can decide to exploit this function by choosing the 'best' action equal to the minimizer of $Q(x,\cdot)$, or may explore the environment by randomly choosing the action. Therefore, having a function of the state-action pair like $\mathscr{Q}$ is fundamentally more efficient than having a function of only the state, like the standard approximation of the value function.

The goal of the algorithm is to compute the following function, for all $x\in\mathscr{X}$ and $a\in\mathscr{A}$:

$$Q^\star(x,a):=\mathbb{E}\big[r(x,x_1,a)+\rho v^\star(x_1)\mid x_0=x,a_0=a\big]=\sum_{y\in\mathscr{X}}\big(r(x,y,a)+\rho v^\star(y)\big)p(x,y,a).$$

Indeed, by the dynamic programming principle (see Chapter 5), we have:

$$v^\star(x)=\inf_{a\in\mathscr{A}}Q^\star(x,a),$$

and we can therefore rewrite $Q^\star$ defined above as the solution to the following fixed-point equation:

$$Q^\star(x,a)=\sum_{y\in\mathscr{X}}\Big(r(x,y,a)+\rho\inf_{b\in\mathscr{A}}Q^\star(y,b)\Big)p(x,y,a).$$

There are now many versions of this algorithm, but we outline here the original algorithm of Watkins.

In this algorithm, we use a stationary randomized policy π, also independent of state. This policy can be chosen arbitrarily but has to satisfy $\pi(a)>0$ for all $a\in\mathscr{A}$, in order to explore all possible actions. The procedure of the algorithm is the following.

Off-policy TD algorithm.

1. Initialization: arbitrarily choose a $\mathcal{Q}$-matrix $Q_0 : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{R}$ and $x_0 \in \mathcal{X}$.
2. For $n = 0, 1, \ldots$, suppose that Q_n and x_n are known (constructed at the previous step or observed).
   (a) Choose an action a_n according to the randomized policy π.
   (b) Choose a new state x_{n+1} according to the (known) transition probability $p(x_n, \cdot, a_n)$.
   (c) Given a sequence of learning rate $\alpha := (\alpha_1, \alpha_2, \ldots)$, update the estimate for Q_n by:

$$Q_{n+1}(x_n, a_n) = Q_n(x_n, a_n) + \alpha_n \Big(r(x_n, x_{n+1}, a_n) + \rho \inf_{b \in \mathcal{A}} Q_n(x_{n+1}, b) - Q_n(x_n, a_n) \Big).$$

3. Stop the recursion when the difference $|Q_{n+1} - Q_n|$ is sufficiently small. In this case, Q_{n+1} is an approximation of $Q^\star$, and the approximation of the value function is given by

$$\widehat{v}(x) = \inf_{a \in \mathcal{A}} Q_{n+1}(x, a).$$

■ **Example 6.4 — Learning recycling Robot.** As in Example 6.1, consider the recycling robot problem, and assume that the expected number of can that can be collected in each mode is unknown, *i.e.* the rewards r_w, r_s are unknown. To find the optimal randomized policy, one can implement the $\mathcal{Q}$-algorithm. In the python code provided in Canvas, the rewards are simulated from a Poisson distribution. ■

Theorem 6.2.2 Suppose that the learning rate α_n satisfies (6.6) and $\pi(a) > 0$ for every $a \in \mathcal{A}$. Then, the sequence Q_n computed by the $\mathcal{Q}$-algorithm converges to the optimal quality function $Q^\star$.

A first proof of convergence was given by Watkins and Dayan in 1992. One may also want to consult the general technical proof given in a paper by John N. Tsitsiklis entitled *Asynchronous Stochastic Approximation and $\mathcal{Q}$-Learning* of 1994.

In a slightly revised version of the $\mathcal{Q}$-algorithm, one does not use an arbitrary policy π but chooses a_n using the ε-greedy policy based on $Q_n(x_n, \cdot)$. Below is the outline of this version:

1. Initialisation: arbitrarily choose a $\mathcal{Q}$-matrix $Q_0 : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{R}$ and $x_0 \in \mathcal{X}$.
2. For $n = 0, 1, \ldots$, suppose that Q_n and x_n are known (constructed at the previous step or observed).
   (a) Let π_n be the ε-greedy policy based on $Q_n(x_n, \cdot)$.
   (b) Choose an action a_n according to the randomized policy π_n.
   (c) Choose a new state x_{n+1} according to the (known) transition probability $p(x_n, \cdot, a_n)$.
   (d) Given a sequence of learning rate $\alpha := (\alpha_1, \alpha_2, \ldots)$, update the estimate for Q_n by:

$$Q_{n+1}(x_n, a_n) = Q_n(x_n, a_n) + \alpha_n \Big(r(x_n, x_{n+1}, a_n) + \rho \inf_{b \in \mathcal{A}} Q_n(x_{n+1}, b) - Q_n(x_n, a_n) \Big).$$

3. Stop the recursion when the difference $|Q_{n+1} - Q_n|$ is sufficiently small. In this case, Q_{n+1} is an approximation of $Q^\star$, and the approximation of the value function is given by $\widehat{v}(x) = \inf_{a \in \mathcal{A}} Q_{n+1}(x, a)$.

6.2.3 SARSA: On-policy TD algorithm

This algorithm is inspired by both the $\mathcal{Q}$-algorithm and the policy iteration method. The name comes from the quintuple " **S**tate x_n, **A**ction a_n, **R**eward r_n, **S**tate $\widehat{x}_n$ and **A**ction $\widehat{a}_n$" that is used in the algorithm in the n-step. It is also called on-line TD algorithm. The goal is again to compute the optimal quality function $Q^\star$. Below is the outline of the algorithm:

On-policy TD algorithm.

1. Initialization: arbitrarily choose a $\mathscr{Q}$-matrix $Q_0 : \mathscr{X} \times \mathscr{A} \mapsto \mathbb{R}$ and $x_0 \in \mathscr{X}$.
2. For $n = 0, 1, \ldots$, suppose that Q_n and x_n are known (constructed at the previous step or observed).
   (a) Let π_n be the ε-greedy policy based on $Q_n(x_n, \cdot)$.
   (b) Choose an action a_n according to the randomiszed policy π_n.
   (c) Choose a state x_{n+1} according to the (known) transition probability $p(x_n, \cdot, a_n)$.
   (d) Let $\widehat{\pi}_n$ be the ε-greedy policy based on $Q_n(x_{n+1}, \cdot)$.
   (e) Choose another action $\widehat{a}_n$ according to the randomized policy $\widehat{\pi}_n$.
   (f) Given a sequence of learning rate $\alpha := (\alpha_1, \alpha_2, \ldots)$, update the estimate for Q_n by:

$$Q_{n+1}(x_n, a_n) = Q_n(x_n, a_n) + \alpha_n \big(r(x_n, x_{n+1}, a_n) + \rho Q_n(x_{n+1}, \widehat{a}_n) - Q_n(x_n, a_n) \big).$$

3. Stop the recursion when the difference $|Q_{n+1} - Q_n|$ is sufficiently small. In this case, Q_{n+1} is an approximation of $Q^\star$, and the approximation of the value function is given by

$$\widehat{v}(x) = \inf_{a \in \mathscr{A}} Q_{n+1}(x, a).$$

6.3 Exercises

Problem 6.1 Consider the armed bandit problem with M arms. For $a = 1, \ldots, M$, the reward distributions are independent and for each arm a they are exponential with mean a, i.e., the probability distribution functions (pdf) of the reward from arm a is

$$f_{R_a}(x) = \frac{1}{a} e^{-x/a}, \qquad x \geqslant 0.$$

Let $\mathbf{a} = (a_1, a_2, \ldots)$ with $a_t \in \{1, \ldots, M\}$ be an action sequence. As in the lectures, let the *pay-off* be given by,

$$J(\mathbf{a}) := \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t.$$

a. Suppose that there are 4 arms (i.e., $M = 4$) and further assume that that we *know the reward distributions*. What is the optimal action sequence? and what is the largest possible J?

b. Suppose that there are two arms (i.e., $M = 2$) and we *first try each arm once* with rewards R_1 and R_2. We *then only use the arm with larger reward* (when equal, we use arm one, i.e, arm one is used if and only if $R_1 \geqslant R_2$). Let J be the resulting random pay-off. Describe the distribution of J.

c. Suppose that there are 4 arms (i.e., $M = 4$) and we use the ε-greedy algorithm with $\varepsilon = 15\%$. Outline the algorithm and what is the resulting J value?
(*Simply state the result and give some reasoning but do not prove it.*)

Problem 6.2 Consider the armed bandit problem with $K = 3$ arms. Let R_a is random reward if arm a is chosen and set $q(a) = \mathbb{E}[R_a]$. Suppose

$$q(1) = 1, \quad q(2) = 2, \quad q(3) = 10.$$

Consider the ε-greedy algorithm with $\varepsilon = 0.2$.

The purpose of this exercise is to show that this greedy algorithm will result a nearly-optimal average reward as T goes to infinity. Let a_t be the action chosen at time t and r_t be the resulting random reward. Let $N_T(a)$, $\widehat{q}_T(a)$ be as above.

a. Show that

$$\liminf_{T\to\infty} \frac{1}{T} N_T(a) \geqslant \frac{\varepsilon}{2} = 0.1, \qquad a = 1,2,3.$$

b. With the help of part a, show that

$$\lim_{T\to\infty} \widehat{q}_T(a) = q(a), \qquad a = 1,2,3.$$

c. Let $p(1) = p(2) = 0.1$, $p(3) = 0.8$. Show that

$$\lim_{T\to\infty} \frac{1}{T} N_T(a) = p(a), \qquad a = 1,2,3.$$

d. Show that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} r_t = \frac{\varepsilon}{2}$$

f. . Run a simulation with $T = 1,000$ or more to test the above result. You may also experiment sending ε to zero (in a controlled manner) in the simulations to see how you can achieve the theoretical maximum of $q(3) = 10$.