Feedback Systems and Reinforcement Learning

Sean Meyn

August 26, 2020

Welcome to the Simons Institute RL Bootcamp!

This draft manuscript was conceived in March 2020. During the spring semester I was giving my stochastic control course, for which the final weeks always focus on topics in RL. Throughout the semester I was thinking ahead to the Simons RL crash course, and a similar one I was supposed to present in Berlin before a pandemic altered my travel plans. A publisher living far away contacted me in May to see if I had any plans for a book: "*...someone mentioned that you were scheduled to deliver lectures on reinforcement learning...*" The idea of a book came up, and I decided this must be destiny. You can learn more about my motivation and goals in the introduction.

This draft manuscript is a product of handouts I've prepared over more than a decade, bits and pieces of papers and book chapters prepared over an even longer period, and the longest time ever confined to home without travel.

There were two themes in particular that I felt a need to spell out:

(i) There is an apparent paradox in Q-learning, considering the long history of convex analytic approaches to dynamic programming, as we will discuss in September lectures. In contrast, there is nothing convex about the Q-learning algorithms used today. Some would object that the least-squares techniques, and DQN are based on convex optimization (subject to linear function approximation). However, the root finding problems they seek to solve are highly non-linear and not well understood. The paradox was addressed in [142], through a convex re-formulation of Q-learning. This summer I set to work to make this more accessible, and show how the ideas are related to standard algorithms in RL.

(ii) Since I was a graduate student, I have enjoyed the idea of ODE methods for algorithm design—probably first inspired by the *ODE method* (a phrased coined by Ljung [134, 171]) in applications to adaptive control. We now know that ODE methods have amazing power in the design of optimization algorithms [193, 224, 181], and they offer enormous insight for algorithm design in general.

The book chapter [70] and thesis [67] make this case in the context of RL, with much of the theory built upon the *Newton-Raphson flow*. This exotic ODE is universally stable, and provides a new tool for algorithm design.

Much of this theory seems to be highly technical. In particular, [70, 67] assume significant background in the theory of stochastic processes, mainly as a vehicle to explain the theory of *stochastic approximation*: the engine behind the ODE method.

I wrote this book to lift the veil: many aspects of RL can be understood by a student with only a good grasp of calculus and matrix theory. In particular, there is nothing inherently *stochastic* about *stochastic approximation*, provided you are willing to work with sinusoids or other deterministic probing signals instead of stochastic processes.

This is my third book, and like the others the writing came with discoveries. While working on theme (i), my colleague Prashant Mehta and I found that *Convex Q-Learning* could be made much more practical by borrowing batch RL concepts that are currently popular. This led to the new work [143]—you will find text and equations from this paper scattered throughout Chapters 3 and 5.

Chapter 4 on ODE methods and *quasi-stochastic approximation* was to be built primarily on [24, 25]. Over the summer all of this material was generalized to create a complete theory of convergence and convergence rates for these algorithms, along with a better understanding

of their application to both *gradient free optimization* and *policy gradient* techniques for RL. The chapter is currently incomplete because a manuscript is in preparation—to be uploaded to arXiv in September.

A quote from [66] came to mind while thinking about this preface: "*...little more than a grab-bag of techniques which have been successfully applied to special situations and are therefore worth trying in sufficiently closely related settings".* The authors were referring to the theory of *Large Deviations*: a highly applicable, but highly abstract branch of mathematics. The same statement applies to the field of reinforcement learning, and control theory, and perhaps any successful engineering sub-discipline.

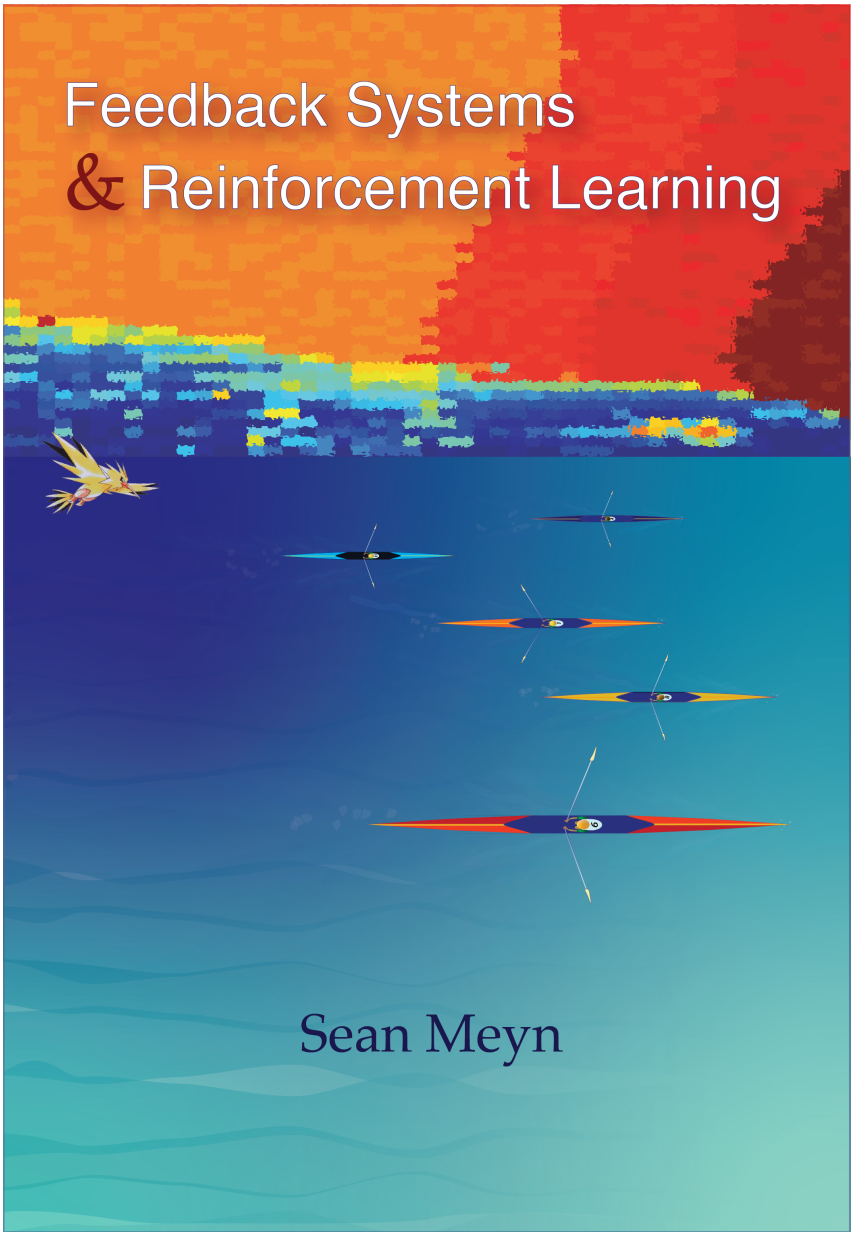
We need a big "grab-bag" because one technique cannot possible solve every control problem in fields ranging from healthcare to supply chains to autonomous vehicles. We need theory for understanding, but we also need reassurance that tricks from the grab-bag have been successfully applied in at least a few special situations. For this reason, any book on reinforcement learning will have its own biases and eccentricities, reflecting on those of the discipline and the author.

I hope this book will be a bit less eccentric when it is finished in 2021!

Sean Meyn — August 26, 2020

Prerequisites for the control-RL crash course on September 3rd:
Please watch at least hour-one of Richard Murray's crash course on control:
Feedback Control Theory: Architectures and Tools for Real-Time Decision Making
`https://simons.berkeley.edu/talks/murray-control-1`

Feedback Systems & Reinforcement Learning



Sean Meyn

Contents

Chapter 1

Introduction

To define *reinforcement learning* (RL) it is first necessary to define *automatic control*. Examples in your every day life may include the cruise control in your car, the thermostat in your air-conditioning, refrigerator and water heater, and the decision making rules in a modern clothes drier. There are sensors that gather data, a computer to take the data to understand the state of the "world" (is the car traveling at the right speed? Are the towels still damp?), and based on these measurements an algorithm powered by the computer spits out commands to adjust whatever needs to be adjusted: throttle, fan speed, heating coil current, or ... More exciting examples include space rockets, artificial organs, and microscopic robots to perform surgery.

The dream of RL is automatic control that is truly automatic: without any knowledge of physics or biology or medicine, an RL algorithm tunes itself to become a super controller: the smoothest ride into space, and the most expert micro-surgeon!

The dream is surely beyond reach in most applications, but recent success stories have inspired industry, scientists, and a new generation of students. DeepMind's AlphaGo set the world on fire following the defeat of European Go champion Fan Hui in 2015. On the news soon after was the astonishing sequel AlphaZero, which learns to play Chess and Go by "self play" without any help from experts [95, 184]. This is now considered old news, with new breakthroughs coming seemingly monthly [176].[1]

1.1 What You Can Find in Here

Reinforcement learning today has two foundations of equal importance:

1. Optimal control: the two most famous RL algorithms, TD- and Q-learning, are all about approximating the *value function* that is at the heart of optimal control.
2. Statistics and information theory (especially the topic of exploration, as in bandit theory — think of the annoying ads on YouTube, which are an example of Google's exploration: "will Diana click???") [175, 131]. Exploration in RL is a rapidly evolving field—it will surely generate many new books in the years to come.

The big focus of the book is the first foundation, emphasizing the geometry of optimal control, and why it should not be difficult to create reliable algorithms for learning. We will not ignore

[1] Dear generous colleague who is reading this intro: Given that I am writing this draft text in July 2020, I know that most of this essay will be tossed 12 months from now, when I send the finished book to the publishers!

the second foundation: motivation and successful heuristics will be explained without diving deeply into the theory. The reader will learn enough to begin experimenting with homemade computer code, and have a big library of options for algorithm design; before completing half of the book, I hope that a student will have a good understanding of why these algorithms are expected to be useful, and why they sometimes fail.

This is only possible through mastery of several fundamentals:

(i) The philosophical foundations of control design

(ii) The theory of optimal control

(iii) ODEs: stability and convergence. The ODE method: translation to algorithm.

(iv) Basics of machine learning (ML): function approximation & optimization theory.

Any reader who knows the author will wonder why the list is so short! The following topics are seen as fundamental in RL theory, and are fundamental to much of the author's research:

(i) Stochastic processes and Markov chains

(ii) Markov decision theory

(iii) Stochastic approximation and convergence of algorithms

Yes, we will get to all of this after Part 1. However, I want to make it clear that there is no need for probability theory to understand many of the important concepts in RL.

The first half of the book contains RL theory and design techniques without any probability pre-requisites. This means that we pretend that the world is purely deterministic until we see the word "Markov" in the second half of the book. Justification comes in part from the control foundations covered in Chapters 2 and 3. Do you think we modeled the probability of hitting a seagull when we designed a control system to go to the moon? The models used in traditional control design are often absurdly simple, but good enough to get insight on how to control a rocket or a pancreas.

Beyond this, once you understand RL techniques in this simplified setting, it does not take much work to extend the ideas to more complex settings.

Among the highlights of part 1 of the book are

• *Batch RL methods and convex Q-learning.* One of the founders of AlphaGo admits that extension of these techniques is not trivial: "This approach won't work in more ill-structured problems like natural-language understanding or robotics, where the state space is more complex and there isn't a clear objective function" [95].

There is no question that in applications to controlling building systems, the energy grid, robotic surgery, or autonomous vehicles, we need to think carefully about more structured learning and control architectures, designed so that we have a reliable outcome (hopefully with some guarantees on performance as well as the probability of disaster). The basic RL engine of AlphaZero is Deep Q-Learning (DQN) [155, 153, 152]; a "batch" Q-learning method that is easily explained, and offers great flexibility to allow the inclusion of "insights from experts". Completely new in this book are convex-analytic approaches to RL that have performance guarantees not possible with DQN.

• *ODE design.* The ODE (ordinary differential equation) method has been a workhorse for algorithm *analysis* since the introduction of the stochastic approximation technique of Robbins and Monro in the early 1950s [171]. In this book we flip the narrative, and start off in the continuous time domain. There is tremendous motivation for this point of view:

(i) We will see that an ODE is so much simpler to describe and analyze than the discrete-time noisy algorithm that is eventually implemented.

(ii) Remarkable discoveries in the optimization literature reinforce the value of this approach: first design an ODE with desirable properties, and then find a numerical analyst to implement this on a computer [181]. It is now known that the famous acceleration techniques of Polyak and Nesterov can be interpreted in this way.

(iii) Zap Q-learning will be one of many algorithms described in the book. It is a particular ODE design based on the Newton-Raphson flow introduced in the economics literature, and first analyzed by Smale in the 1970s [186]. The Zap ODE is universally stable and consistent, and hence when translated with care, provides new techniques for RL design [49, 71, 70, 67]. The power of Zap design is illustrated in Fig. 1.1 (created for a lecture at the 2018 Simons RTDM program)

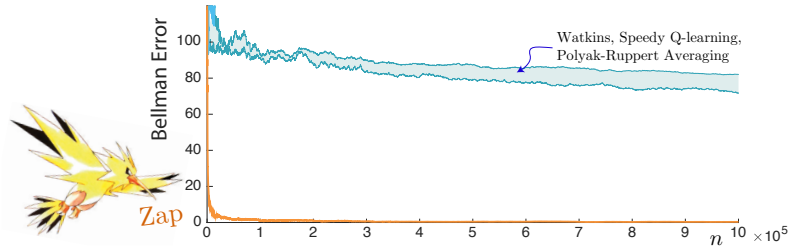


Figure 1.1: Maximum Bellman error $\{\overline{\mathcal{B}}_n : n \ge 0\}$ for various Q-learning algorithms.

• *Quasi-stochastic approximation.* The theory of "stochastic approximation" amounts to justifying a discrete time algorithm based on an ODE approximation. An understanding of the general theory requires substantial mathematical background.

The reader will be introduced to stochastic approximation, without any need to know the meaning of "stochastic". This is made possible by substituting mixtures of sinusoids for randomness, which is justified by an emerging science [133, 6, 30, 29, 52, 24, 25]. Not only is this more accessible, but the performance in application to *policy gradient* methods in RL is remarkable. Shown below is a figure copied from Chapter 4, comparing exploration using sinusoids vs. traditional random "i.i.d." exploration. The histograms were created based on 1000 independent experiments. The traditional approach labeled "K-W" requires additional training of many orders of magnitude when compared to QSA.

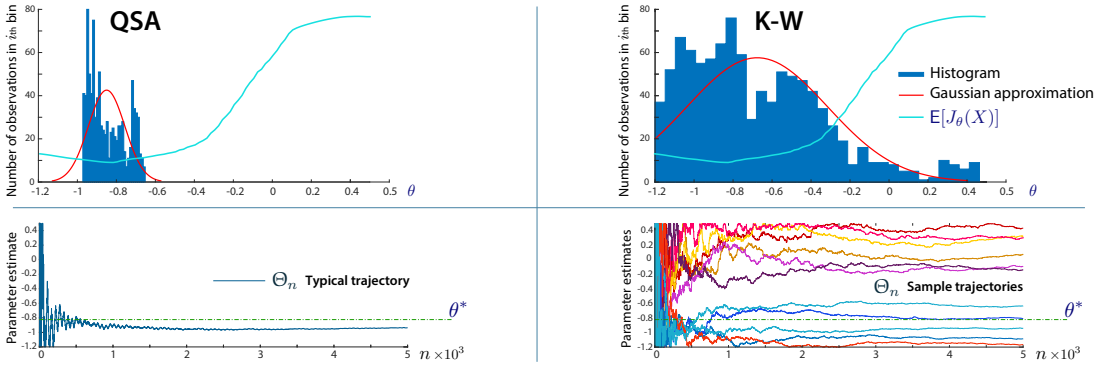


Figure: Error analysis for two policy-gradient algorithms for Mountain Car, using QSA and traditional randomized exploration.[2]

1.2 What's Missing?

The focus of the book is on control fundamentals that are most relevant to reinforcement learning, and a large collection of tools for RL algorithm design based on these fundamentals.

Important topics that will receive less attention:

(i) *Exploration.* This is a hot topic for research right now [131, 175, 164]. I feel this field is too big, and the state of the art is not at all mature (though I might be proven wrong before December! I look forward to learning more). An exception is within the area of bandit theory: we will survey the basics of bandits, which motivates much of the modern exploration theory for RL.

(ii) *Machine learning topics.* The book will explain the meaning of neural networks and kernels, but ask the reader to go elsewhere for details.

Sample complexity theory will be covered only briefly. Dear colleauges and participants at the 2020 Simons Institute RL program: There is no question that sample complexity theory is the bedrock of classical learning theory, and the theory of bandits. However, I personally believe that its value in RL is limited: bounds are typically loose, and to-date they offer little insight for algorithm design. For example, I don't see how today's finite-n bounds will help DeepMind create better algorithms for Go or Chess. I welcome counter-examples to my perception.

I also believe that the value of asymptotic statistics is under-appreciated. The best way to make this point is through images. Fig. 1.2 is taken from [73] (many similar plots can be found in the thesis of A. Devraj [67]).

The histograms show estimation error for a single parameter (one of many) using a particular implementation of *tabular Q-learning*. The integer N refers to the run-length of the algorithm, and the histograms were obtained from 1,000 independent experiments. The "theoretical density" is what you can obtain from asymptotic statistics theory for stochastic approximation. This density is easily estimated based on limited data. In particular, in this experiment, it is clear after $N = 10^4$ samples that we have a very good variance estimate. This can then be used to obtain approximate confidence bounds after we run to $N = 10^7$ (we know how far we have to run once we have a variance estimate).

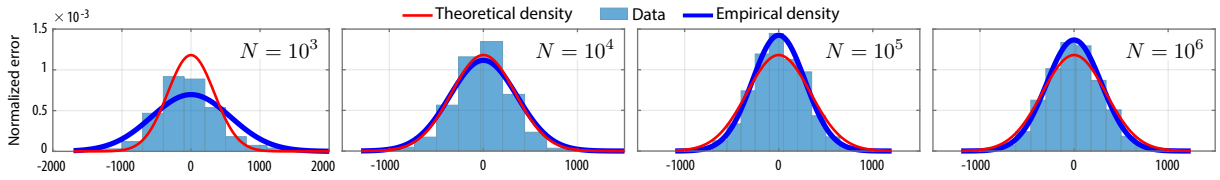


Figure 1.2: Comparison of Q-learning and Relative Q-learning algorithms for the stochastic shortest path problem of [71]. The relative Q-learning algorithm is unaffected by large discounting.

Asymptotic theory is developed for QSA at the close of Chapter 4. Something like finite-n bounds are also developed, but tight bounds are not possible for complex algorithms, even

[2] I hope to upload a paper to arXiv in September.

though the asymptotic theory is straightforward. The more traditional variance theory for SA—explaining the histograms in Fig. 1.2—is a focus of Chapter 9.

Variance theory for SA is used to decide run-lengths. It is also used for algorithm *design*: for example, adjust variables in an algorithm so that the asymptotic variance is minimized (Zap stochastic approximation is just one example). Returning to my audience at the Simons program: outside of the bandits literature, *I am not aware of work in the sample complexity literature that can offer similar value for algorithm design in reinforcement learning.*

1.3 Words of Thanks

1.4 Resources

Part I

Fundamentals Without Noise

Chapter 2

Control Crash Course

A single chapter can hardly do justice to the amazing universe of control theory and practice. The textbook [9] gives a very accessible introduction to the philosophy and practice of control, and is also full of history. I was fortunate to be at the Simons Institute at Berkeley, California, when one of the authors presented a two-part survey of ideas in the book.[3] These lectures are a great starting point if you are new to control systems, and will inspire many old-timers.

2.1 You Have a Control Problem

You surely have encountered control problems in your daily life. If you know how to drive, then you know what it is like to be part of a control system:

- $\boldsymbol{y}$ The *observations* (also called the "output") refers to the data you process in order to effectively maneuver the car through traffic: this includes your view of the streets and lights, and the sounds of angry drivers pleading with you to speed up.
- $\boldsymbol{u}$ You apply *inputs* to the system: steering wheel, brakes and gas pedal are continuously adjusted based on your observations.
- ϕ This symbol will be used to denote an algorithm that takes in the observations y and produces the response u. This mapping from y to u is known as a *policy* or *feedback law* (the Greek letter is pronounced "*fee*").
- *ff* You are not simply reacting to horns and lights and the lines on the road. You started off with a plan: get to the farmers market by 9am, while avoiding the traffic downtown due to the demonstration. This planning is an example of *feedforward* control. Planning is based on forecasts, so inevitably plans will change as you gather information en-route: traffic updates, or an invitation from a close friend to park your car and join the protest.

The feedforward component is typically defined with attention to a *reference signal* denoted $\boldsymbol{r}$. The primary control objective is the *tracking problem*: construct a policy so that some object of interest $z(k)$ is approximately equal to $r(k)$ for all $k \geq 0$ (in control courses, it is often assumed that $z = y$).

The yelling and bumps on the road are collectively known as *disturbances*. Along with the reference signal, partial measurements of disturbances and their forecasts are taken into account

[3] https://simons.berkeley.edu/talks/murray-control-1

in both the feedforward and feedback components of the control system. The final input is often defined as the sum of two components:

$$u(k) = u_{ff}(k) + u_{fb}(k) \tag{2.1}$$

where in the shopping problem, u_{ff} quantifies the results of planning before heading to the market (perhaps with updates every 20 minutes), and u_{fb} is the second-by-second operation of the automobile.

The dream of RL is to mimic and surpass the skill in which humans create an internal algorithm ϕ, and use it to skillfully navigate through complex and unpredictable environments.

Fig. 2.1 shows a block diagram typically used in model-based control design, and illustrates a few common design choices: there is a state to be estimated using an **observer**, with state estimates denoted $\widehat{x}$. The block denoted **trajectory generation** generates two signals: the feedforward component of the control, and also a reference x_{ref} that an internal state should track (the state is associated with the physical process). It is designed so that $x(k) = x_{\text{ref}}(k)$ for all k implies that the tracking problem is solved, and the solution is efficient. The **state feedback** is designed to achieve this ideal.

There is a larger "world state" labeled **environment** for which partial measurements are available, and forecasts of future events. Forecasts are of course important in the planning process that is part of trajectory generation.

Design of the three grey blocks is based on models of the process, the sensor noise n, disturbances (such as the "input disturbance" d indicated in the figure), and a model of the environment. The "Δ-feedback loop" is a standard way to quantify model uncertainty associated with the process to be controlled.

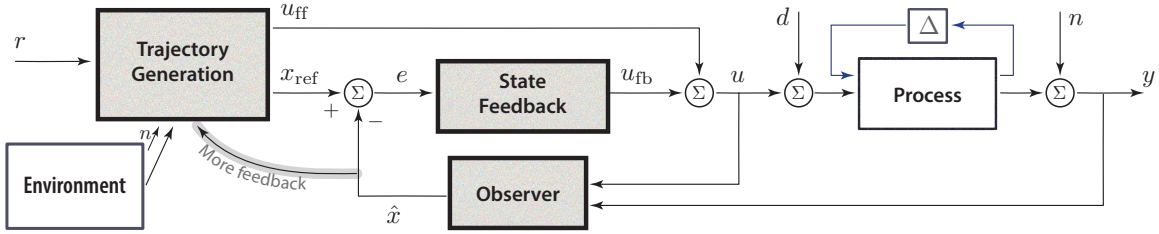


Figure 2.1: Control systems contain purely reactive feedback, as well as planning that is regularly updated. This represents two layers of feedback, differentiated in part by speed of response to new data.

2.2 What To Do About It?

The vast literature on control solutions is built upon a model of input-output behavior that is used to design the policy ϕ. Modeling and control design are each an art-form, with many possible solutions from vast statistics and control tool chests.

When we say *model*, we mean a sequence of mappings from inputs to outputs:

$$y(k) = \mathrm{G}_k(u(0), u(1), u(2), \ldots, u(k)), \qquad k \geq 0 \tag{2.2}$$

Each of the functions G_k may also depend on *exogenous variables* (outside of our control), such as the weather and traffic conditions. And here we come to one of the most vital principles of

control design: *the model must capture essential properties of the system to be controlled*, and simultaneously *simple enough to be useful*.

For example, aerospace engineers will create *absurdly simple models* for the design of flight control systems, and from this create a policy ϕ designed to work well for the model. Of course, they do not stop there! The next step is to create an entirely new model for validation, and simulate under a range of scenarios in order to answer a range of questions: what happens when the plane is full, empty, or flying through a thunderstorm? How does the control system perform after an engine detaches from a wing? If one of these tests fails, then the control engineer goes back to either improve the model, improve the policy, or *improve the airplane*. That's right: we may find that additional sensors to measure pitch angles, or more powerful motors to control ailerons, flaps or elevators.

Again, I am writing without any knowledge of aerospace engineering! I am describing general principles for anyone interested in control design:

1. Create a model for control.
2. Design the policy ϕ based on the model
3. Simulate based on a high-fidelity model, and then revisit steps 1 and 2.

The success of this approach has been tremendous, as can be seen in the history recounted in [9].

The LTI model The most successful class of *absurdly simple models* are linear and time invariant, *LTI systems*. To keep the discussion simple, assume that the input $u(k)$ and output $y(k)$ are scalar real numbers. The general scalar LTI system is defined by a sequence of scalars $\{b_i\}$ (the *impulse response*), and for a given input sequence $\boldsymbol{u}$, the model defines $y(k)$ as the sum

$$y(k) = \sum_{i=0}^{k} b_i u(k-i)\,, \qquad k \geq 0 \tag{2.3}$$

This is in fact too complex in many situations. In particular, where do we find the infinite number of scalars $\{b_i\}$? A more tractable sub-class of LTI models are auto-regressive moving-average (ARMA):

$$y(k) = -\sum_{i=1}^{N} a_i y(k-i) + \sum_{i=0}^{M} b_i u(k-i)\,, \qquad k \geq 0 \tag{2.4}$$

with scalar coefficients $\{a_i, b_i\}$, $N \geq 1$ and $M \geq 0$.

A linear input-output model motivates the design of a policy ϕ that has a similar linear form. A common design technique based on optimization will be described in the following chapters.

2.3 State Space Models

2.3.1 Sufficient statistic and the nonlinear state space model

In statistics, the term *sufficient statistic* is used to denote a quantity that summarizes all past observations. The *state* serves an analogous role in control theory.

A *state space model* requires the following ingredients: the *state space* X on which the state $\boldsymbol{x}$ evolves, an *input space* (or *action space*) denoted U on which the input $\boldsymbol{u}$ evolves. We may have additional constraints coupling the state and the input, which is modeled via

$$u(k) \in \mathsf{U}(x)\,, \qquad \text{when } x(k) = x \in \mathsf{X} \tag{2.5}$$

with $\mathsf{U}(x) \subseteq \mathsf{U}$ for each x. We might also want to model an observation process $\boldsymbol{y}$ evolving on a set Y. In the control theory literature it is common to assume that X, U, and Y are subsets of Euclidean space, while in operations research and reinforcement learning theory it is more common to assume these are finite sets. Whenever possible, in this book we prefer the control perspective so that we can more easily search for structure of control solutions: for example, is an optimal input a continuous function of the state?

Next we require two functions $\mathrm{F}\colon \mathsf{X} \times \mathsf{U} \to \mathsf{X}$ and $\mathrm{G}\colon \mathsf{X} \times \mathsf{U} \to \mathsf{Y}$ that define the state equations:

$$x(k+1) = \mathrm{F}(x(k), u(k)), \qquad x(0) = x_0 \tag{2.6a}$$

$$y(k) = \mathrm{G}(x(k), u(k)) \tag{2.6b}$$

An LTI model can often be transformed into a state space model in which the two functions F, G are linear in (x, u).

We might also allow F, G to depend upon the time variable k. It is argued in Section 3.3 that it is often more convenient to simply assume that the state $x(k)$ includes k as one component.

However, there is one example of a time-dependent model that highlights the role of state as a sufficient statistic: The general input-output model (2.2) always has a state space description, in which the state is the full history of inputs:

$$x(k+1) = [u(0), u(1), u(2), \dots, u(k)]^{\intercal} \tag{2.7}$$

We have $x(k+1) = \mathrm{F}_k(x(k), u(k))$, defined by concatenation; also, $y(k) = \mathrm{G}_k(x(k), u(k))$, where G_k is defined in (2.2). For this deterministic model in which the the input fully determines the output, (2.7) is called the (full) *history state* since it captures all observations up to time k. A practical state space model can be regarded as a compression of the complete observations.

In many cases we can construct a good policy via *state feedback*, $u(k) = \phi(x(k))$, for some $\phi\colon \mathbb{R}^n \to \mathbb{R}$, but the power of this approach is fully realized only if we are flexible in our definition of the state. We won't be using the full history state because of complexity; what's more, *the "full history" may not be nearly rich enough*!

2.3.2 State augmentation and learning

Tracking and disturbance rejection are two of the basic goals in control design. Here we provide a brief glimpse of two tricks used to simultaneously track the reference $\boldsymbol{r}$ while rejecting disturbances:

(i) The definition of state is not sacred: invent a state process that simplifies control design

(ii) Unknown quantities, including disturbances and even the state space model, can be learned based on input-output measurements.

For simplicity, suppose that the input and output are scalar valued, and $\mathsf{X} = \mathbb{R}^n$. The state evolution is also influenced by a scalar disturbance $\boldsymbol{d}$ that is outside of our control, which requires a modification of (2.6a):

$$x(k+1) = \mathrm{F}(x(k), u(k), d(k)) \tag{2.8}$$

The ultimate goal is to achieve these three objectives simultaneously:

(a) *Tracking*: with $e(k) = r(k) - y(k)$,

$$\limsup_{k\to\infty} |e(k)| = e_\infty\,, \qquad \textit{with } e_\infty = 0\textit{, or very small} \tag{2.9}$$

(b) *Disturbance rejection*: The error e_∞ is not highly sensitive to the disturbance $\boldsymbol{d}$.

(c) Tuned transient response (you probably know what kind of acceleration "feels right" when driving a car).

A common special case is when the reference and disturbance are assumed independent of time (e.g., driving at constant speed with a steady headwind). In this special case, suppose in addition that the disturbance is known. We might choose $u(k) = \phi(x(k), r(0), d(0))$, where the policy ϕ is designed for success: $e_\infty = 0$. Typically, ϕ is designed so that the state is also convergent: $x(k) \to x(\infty)$ as $k \to \infty$. The limiting values must satisfy

$$\begin{aligned} x(\infty) &= \mathrm{F}(x(\infty), u(\infty), d(0)) \\ u(\infty) &= \phi(x(\infty), r(0), d(0)) \end{aligned}$$

The outcome $e_\infty = 0$ is expressed as the final constraint:

$$r(0) = y(\infty) = \mathrm{G}(x(\infty), u(\infty))$$

This approach is thus dependent on an accurate model, as well as direct measurements of $\boldsymbol{d}$.

Suppose that instead of exact measurements of the disturbance, we have a state space model for $(\boldsymbol{r}, \boldsymbol{d})$:

$$z(k+1) = \mathrm{F}_{\mathrm{m}}(z(k)) \tag{2.10a}$$

$$y_m(k) = \mathrm{G}_{\mathrm{m}}(z(k)) \tag{2.10b}$$

with $y_m(k) = (r(k), d(k))^\intercal$, and $\boldsymbol{z}$ evolves on $\mathbb{R}^p$ for some integer p. The functions $\mathrm{F}_{\mathrm{m}} \colon \mathbb{R}^p \to \mathbb{R}^p$ and $\mathrm{G} \colon \mathbb{R}^p \to \mathbb{R}$ are assumed known.

Given the larger state space model (2.8, 2.10), we might opt for an *observer* based solution:

$$u(k) = \phi(x(k), r(k), \hat{d}(k))$$

where $\{\hat{d}(k)\}$ are estimates of the disturbance, based on input-output measurements up to time k (we might even replace $x(k)$ with $\hat{x}(k)$—sometimes we don't directly observe the state). Observer design makes up about 20% of a typical introductory course on state space control systems [1, 43].

A second option, called the Internal Model Principle, is to create a different state augmentation that is entirely observed. For the sake of illustration, consider again the case of constant

reference/disturbance. We have (2.10) in this case with $z(k) = y_m(k)$, and $\mathrm{F_m}$ is the identity function:

$$z(k+1) = z(k)$$

The state augmentation is performed based on this model: define for each k,

$$z_a(k+1) = z_a(k) + e(k+1)\,, \tag{2.11}$$

with error $\boldsymbol{e}$ defined above (2.9). We regard $(x(k), z_a(k))$ as the state for the purposes of control, and hence state feedback takes the form

$$u(k) = \phi(x(k), z_a(k)) \tag{2.12}$$

The control design (2.12) is an example of *integral* control, since $\boldsymbol{z}_a$ is the sum of errors (the discrete-time analog of integration).

Suppose that $z_a(k)$ converges to some finite limit $z_a(\infty)$, as $k \to \infty$; the value of the limit is irrelevant. This and eq. (2.11) imply perfect tracking:

$$\lim_{k\to\infty} e(k+1) = \lim_{k\to\infty} [z_a(k+1) - z_a(k)] = 0.$$

This conclusion is remarkable: we only require a very weak form of stability of (2.12) in order to obtain perfect tracking. The secret to success is a hidden element of "learning" that comes with integral control.

State augmentation has many other dimensions. If we have forecasts of significant disturbances, then it may be wise to make use of this data: forecasts can be used in the design of the feedforward component $u_{\mathit{ff}}(k)$ in the decomposition (2.1), or they may be used for state augmentation. Much more on these topics can be found in [9].

2.3.3 Linear state space model

When F and G are linear we obtain the linear state space model:

$$x(k+1) = Fx(k) + Gu(k), \qquad x(0) = x_0 \tag{2.13a}$$

$$y(k) = Hx(k) + Eu(k) \tag{2.13b}$$

where (F, G, H, E) are matrices of suitable dimension (in particular, F is $n \times n$ for an n-dimensional state space).

The state space model is not unique, in the sense that there are many choices for (F, G, H, E) that result in the same input-output behavior, even though the definition of the state process $\boldsymbol{x}$ will change depending on the model. And never forgot: *we may add additional components to $x(k)$ as a means to solve a control problem.*

Linear state feedback The linear model (2.13) is often constructed so that the goal is to keep $x(k)$ near the origin—the *regulation problem*; consider for example flight control, where we wish to maintain velocity and altitude at some constant values. We first normalize the problem so that these constant values are *zero*. It is then common to apply a linear control law:

$$u(k) = -Kx(k) \tag{2.14}$$

where K is called the *gain matrix*. To evaluate choice of gain, we tack-on something like a reference signal:

$$u(k) = -Kx(k) + v(k)$$

The *closed loop behavior* with new "input $\boldsymbol{v}$" has a similar state space description:

$$x(k+1) = (A - BK)x(k) + Bv(k), \qquad x(0) = x_0 \tag{2.15a}$$
$$y(k) = (C - DK)x(k) + Dv(k) \tag{2.15b}$$

The signal $v(k)$ appearing in (2.15a) is viewed as an "input disturbance". In the regulation problem, the goal is to investigate its impact on control performance.

Realization theory The ARMA model (2.4) admits an infinite number of distinct state space descriptions). Let's begin with the scalar auto-regressive model:

$$y(k) = -\sum_{i=1}^{N} a_i y(k-i) + u(k)\,, \qquad k \geq 0$$

which is (2.4), with $M = 0$ and $b_0 = 1$. We obtain the state space model (2.13) with $n = N$ by choosing $x(k) = (y(k), \dots, y(k-N+1))^\intercal$, and

$$A = \begin{bmatrix} -a_1 & -a_2 & -a_3 & \cdots & \cdots & -a_N \\ 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.16}$$

$C = [1\,,\ 0\,,\ 0\,,\ \cdots, 0]$, and $D = 0$.

This construction can be generalized: with $M = N$ in (2.4), we first define an intermediate process

$$z(k) = -\sum_{i=1}^{N} a_i z(k-i) + u(k)\,, \qquad k \geq 0$$

So that we arrive at a state space model with state space $x(k) = (z(k), \dots, z(k-N+1))^\intercal$ to describe the evolution of $\boldsymbol{z}$. We next use the assumption that $M = N$: setting $u(k) = z(k) = 0$ for $k < 0$, it is possible to show that

$$y(k) = \sum_{i=0}^{N} b_i z(k-i) = Cx(k) + Du(k)$$

where

$$C = [b_0\,,\ b_1\,,\ \cdots,\ b_{N-1}]\,, \qquad D = b_N \tag{2.17}$$

The state space description (2.16,2.17) is known as *controllable canonical form*. There are many other "canonical forms", with special properties you can learn about in a linear systems course [17, 125, 1, 43, 9].

2.3.4 A nod to Newton and Leibnitz

In many engineering applications it is best to start off in continuous time, with thanks to Newton and Leibnitz for bringing us calculus. It is convenient to suppress time dependency for the continuous time models. For example, provided there is no risk of ambiguity, we write u instead of u_t, and its derivative with respect to time is then denoted $\frac{d}{dt}u$.

The state space model in continuous time has the form

$$\tfrac{d}{dt}x = \mathrm{f}(x, u) \tag{2.18}$$

where x is the state evolving in $\mathbb{R}^n$, and u the input evolving in $\mathbb{R}^m$. The motion of a typical solution to a nonlinear state space model in $\mathbb{R}^2$ is illustrated in Fig. 2.2.

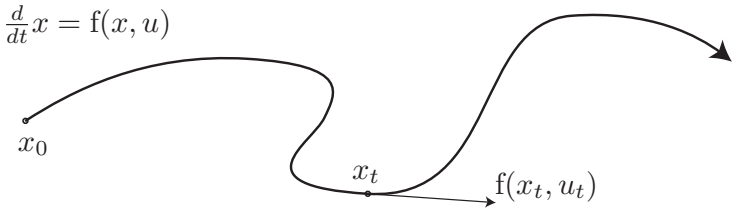


Figure 2.2: Trajectory of a nonlinear state space model in two dimensions: at any time t, the velocity $\frac{d}{dt}x_t$ is a function of the current state x_t and input u_t.

When the function f appearing in (2.18) is linear, then we obtain the linear state space model in continuous time. As in (2.13), this is accompanied by an observation process $\boldsymbol{y}$ evolving on $\mathbb{R}^p$:

$$\tfrac{d}{dt}x = Ax + Bu \tag{2.19a}$$
$$y = Cx + Du, \tag{2.19b}$$

and A, B, C, D are matrices of appropriate dimensions.

The geometry illustrated in Fig. 2.2 (that vector $\mathrm{f}(x_t, u_t)$ is tangent to the state trajectory) is surprisingly valuable in gaining intuition in control design. An example of application is Lyapunov theory and optimal control, where theory is more attractive in the continuous rather than discrete time domain

However, in the end we have to sample time to apply our control and learning algorithms. In this book we will opt for an Euler approximation. For sampling interval Δ, the discrete time approximation of (2.18) is of the form (2.6a), with $\mathrm{F}(x, u) = x + \Delta \mathrm{f}(x, u)$. For the linear model (2.19a) this leads to $F = I + \Delta A$.

2.4 Stability and Performance

In this section we consider the state space model (2.6a) in *closed loop*: a policy ϕ is chosen, so that $u(k) = \phi(x(k))$ for each k. Since the feedback law is fixed, the state then evolves as a state space model without control. With just a slight abuse of notation we write

$$x(k+1) = \mathrm{F}(x(k))\,, \qquad k \geq 0 \tag{2.20}$$

Our interest is in the long-run behavior of the state process; in particular, does it converge to an *equilibrium*? We also seek bounds on a particular performance metric called the *total cost*.

The following is assumed throughout:

$$\textit{The state space } \mathsf{X} \textit{ is equal to } \mathbb{R}^n\textit{, or a closed subset} \tag{2.21}$$

For example we allow the positive orthant, $\mathsf{X} = \mathbb{R}^n_+$. The restriction on the state space (2.21) is imposed so that any closed and bounded set $S \subset \mathsf{X}$ is necessarily a compact subset of X.

The definition of an equilibrium x^e is straightforward—it is a state at which the system is frozen:

$$x^e = \mathrm{F}(x^e) \tag{2.22}$$

The equilibrium will in fact be a part of the control design. Think of the cruise control in your car, in which "equilibrium" means traveling in a straight line at constant speed. The particular speed is something that you as the driver will choose. The control system then does the best it can to keep $x(k)$ near the desired x^e.

The performance metric is based on a function $c\colon \mathsf{X} \to \mathbb{R}_+$, interpreted as the "cost function under policy ϕ", to be considered in greater depth in Chapter 3. Now we arrive at a strange but ubiquitous definition: the total cost is a function of x, known as the *value function*. It is defined as the infinite sum,

$$J(x) = \sum_{k=0}^{\infty} c(x(k))\,, \qquad x(0) = x \in \mathsf{X} \tag{2.23}$$

It is assumed that $c(x^e) = 0$, and we will seek conditions ensuring that $x(k) \to x^e$ as $k \to \infty$, so there is some hope that J is finite valued. For the cruise control problem, the cost function is designed so that $c(x)$ is large if the state x corresponds to a speed that is far from desired.

Why is the controls community so excited about total cost? This metric for performance is not very intuitive, but there are compelling reasons for using it as a metric for control design:

(i) This performance metric is "forward looking". One might argue it is looking too far foward (who cares about infinity), but there is implicit "discounting" of the future since for a good policy we have $c(x(k)) \to 0$ quickly as $k \to \infty$.

(ii) Theory for total cost optimal control is often closely related to average cost optimal control—to be seen in Part 2 of the book.

(iii) If J is finite valued, then stability is typically guaranteed.

Benefit (iii) is the most abstract, but the most valuable aspect of this performance metric.

Section 2.4.1 is dedicated to stability theory and its relationship to value functions. A part of this theory is based on the "fixed policy" dynamic programming equation:

$$J(x) = c(x) + J(\mathrm{F}(x)) \tag{2.24}$$

This can be derived from the definition (2.23), written as

$$J(x) = c(x) + \sum_{k=0}^{\infty} c(x^+(k))$$

where $\boldsymbol{x}^+$ is the solution to (2.20), starting at $x^+(0) = \mathrm{F}(x)$.

2.4.1 Stability of equilibria

We first survey the three most common definitions of stability for a nonlinear state space model. The first and most basic is essentially a notion of continuity near the equilibrium x^e. Let $\mathcal{X}(k; x_0)$ denote the state at time k with initial condition x_0: this is simply $x(k)$, obtained recursively from (2.20), starting at $x(0) = x_0$. In particular, $\mathcal{X}(k; x^e) = x^e$ for all k since x^e is an equilibrium.

Definition 2.1. ***Stable in the sense of Lyapunov***

The equilibrium x^e is stable in the sense of Lyapunov if for all $\varepsilon > 0$, there exists $\delta > 0$ such that if $\|x_0 - x^e\| < \delta$, then

$$\|\mathcal{X}(k; x_0) - \mathcal{X}(k; x^e)\| < \varepsilon \qquad \text{for all } k \geq 0.$$

■

In words, if an initial condition is close to the equilibrium, then it will stay close forever. An illustration is provided in Fig. 2.3, with

$$B_r(x^e) = \{x \in \mathbb{R}^n : \|x - x^e\| < r\}, \qquad r > 0$$

This is a very weak notion of stability, since there is no guarantee that the state will ever approach a desired equilibrium. The next definitions impose convergence:

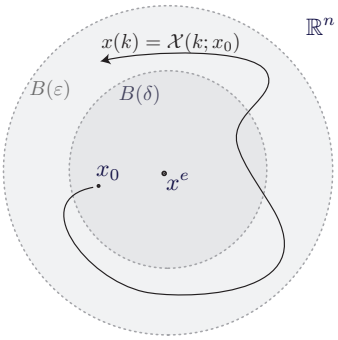


Figure 2.3: If $x_0 \in B(\delta)$, then $\mathcal{X}(k; x_0) \in B(\varepsilon)$ for all $k \geq 0$.

Definition 2.2. ***Asymptotic Stability***

An equilibrium x^e is said to be *asymptotically stable* if x^e is stable in the sense of Lyapunov and for some $\delta_0 > 0$, whenever $\|x_0 - x^e\| < \delta_0$,

$$\lim_{k \to \infty} \mathcal{X}(k; x_0) = x^e. \tag{2.25}$$

The set of x_0 for which the above limit holds is called the *region of attraction* for x^e.

The equilibrium is *globally asymptotically stable* if the region of attraction is all of X: that is, $\delta_0 = \infty$, and hence $x(k) \to x^e$ from any initial condition. ■

It is common to say that the state space model is globally asymptotically stable. That is, stress that this is a property of the system (2.20) rather than the equilibrium.

2.4.2 Lyapunov functions

The construction of a *Lyapunov function* V is the most common approach to establishing asymptotic stability, as well as bounds on a value function (and more general bounds on the state process). In broad generality, V is a function on X taking non-negative values, and the crucial property that makes it a Lyapunov function is that $V(x(k))$ is decreasing when $x(k)$ is large: this is formalized as a *drift inequality*. The Lyapunov function V is often regarded as a crude notion of "distance" to the "center of the state space".

For any scalar r, let $S_V(r)$ denote the *sublevel set*:

$$S_V(r) = \{x \in \mathsf{X} : V(x) \le r\} \tag{2.26}$$

In addition to non-negativity of V, we frequently assume that for each r, the set $S_V(r)$ takes on one of three forms: the empty set, $S_V(r) = \mathsf{X}$, or $S_V(r) \subset \mathsf{X}$ is bounded. Under this assumption, the function V is called *inf-compact*.

In most cases we find that $S_V(r) = \mathsf{X}$ is impossible, so that we arrive at the stronger *coercive* assumption:

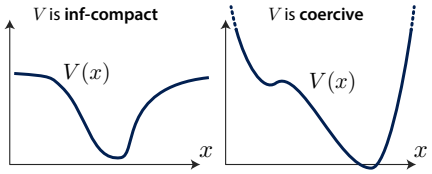


Figure 2.4: The function on the left is inf-compact (in this example, it is bounded on $\mathbb{R}$), and the function on the right is coercive.

$$\lim_{\|x\|\to\infty} V(x) = \infty \tag{2.27}$$

In this case, under our standing assumption (2.21), the closure of $S_V(r)$ is either empty or compact for each r.

Fig. 2.4 illustrates the two classes of functions with $\mathsf{X} = \mathbb{R}$. Two numerical examples:

(i) $V(x) = x^2$ is coercive since (2.27) holds.

(ii) $V(x) = x^2/(1+x^2)$ is inf-compact but not coercive: $S_V(r) = \mathbb{R}$ for $r \ge 1$, and $S_V(r) = [-a, a]$ (a bounded interval) for $0 \le r < 1$, with $a = \sqrt{r/(1-r)}$.

(iii) $V(x) = e^x$ is neither, since $S_V(r) = (-\infty, \log(r)]$, $r > 0$, is not a bounded subset of $\mathbb{R}$.

The value function J satisfies the intuitive properties of a Lyapunov function under mild conditions:

Lemma 2.1. *Suppose that the cost function c and the value function J defined in (2.23) are non-negative, and finite valued. Then,*

(i) *$J(x(k))$ is non-increasing, and $\lim_{k\to\infty} J(x(k)) = 0$ for each initial condition.*

(ii) *Suppose in addition that J is continuous, inf-compact, and vanishes only at x^e. Then, for each initial condition,*

$$\lim_{k\to\infty} x(k) = x^e$$

The proof is postponed to the end of the section, but we note here the first steps: the dynamic programming equation (2.24) implies that

$$J(x(k+1)) = J(\mathrm{F}(x(k))) = J(x(k)) - c(x(k)) \le J(x(k))\,, \qquad k \ge 0 \tag{2.28}$$

That is, $J(x(k))$ is non-increasing, so that $x(k) \in S_J(r)$ for each $k \ge 0$, with $r = J(x(0))$. The inf-compact assumption then implies that the state trajectory is "bottled-up" in the bounded set $S_J(r)$.

In the context of total-cost optimal control, the basic drift inequality considered in this book is *Poisson's inequality*: for non-negative functions $V, c\colon \mathsf{X} \to \mathbb{R}_+$, and a constant $\overline{\eta} \ge 0$,

$$V(\mathrm{F}(x)) \le V(x) - c(x) + \overline{\eta} \tag{2.29}$$

The reference to a great French mathematician is explained in the notes. Poisson's inequality is a relaxation of the dynamic programming equation (2.24) through the introduction of $\overline{\eta}$, as well as the inequality.

The inequality (2.29) is defined with attention to the dynamics (2.20), which together imply the family of bounds (similar to (2.28)):

$$V(x(k+1)) \le V(x(k)) - c(x(k)) + \overline{\eta}\,, \qquad k \ge 0$$

If $\overline{\eta} = 0$, it follows that the sequence $\{V(x(k) : k \ge 0\}$ is non-increasing. Under mild assumptions on V, we obtain a weak form of stability:

Proposition 2.2. *Suppose that (2.29) holds with $\overline{\eta} = 0$. Suppose moreover that V is continuous, inf-compact, and with a unique minimum at x^e. Then the equilibrium is stable in the sense of Lyapunov.*

Proof. From the definition of the sublevel sets we obtain

$$\bigcap \{S_V(r) : r > V(x^e)\} = S_V(r)\Big|_{r=V(x^e)} = \{x^e\}$$

The final equality follows from the assumption that x^e is the unique minimizer of V. The inf-compact assumption then implies the following inner and outer approximations: for each $\varepsilon > 0$, we can find $r > V(x^e)$ and $\delta < \varepsilon$ such that[4]

$$\{x \in \mathsf{X} : \|x - x^e\| < \delta\} \subset S_V(r) \subset \{x \in \mathsf{X} : \|x - x^e\| < \varepsilon\}$$

If $\|x_0 - x^e\| < \delta$, then $x_0 \in S_V(r)$, and hence $x(k) \in S_V(r)$ for all $k \ge 0$ since $V(x(k))$ is non-increasing. The final inclusion above then implies that $\|x(k) - x^e\| < \varepsilon$ for all k. Stability in the sense of Lyapunov follows. ∎

Bounds on the value function J are obtained by iteration: for example, the two bounds

$$V(x(2)) \le V(x(1)) - c(x(1)) + \overline{\eta}\,, \quad V(x(1)) \le V(x(0)) - c(x(0))$$

imply that $V(x(2)) \le V(x(0)) - c(x(0)) - c(x(1)) + 2\overline{\eta}$. We can of course go further:

Proposition 2.3. (Comparison Theorem) *Poisson's inequality (2.29) implies the following:*

(i) *For each $N \ge 1$ and $x = x(0)$,*

$$V(x(N)) + \sum_{k=0}^{N-1} c(x(k)) \le V(x) + N\overline{\eta} \tag{2.30}$$

(ii) *If $\overline{\eta} = 0$, then $J(x) \le V(x)$ for all x.*

(iii) *Suppose that $\overline{\eta} = 0$, and that V, c are continuous. Suppose moreover that c is inf-compact, and vanishes only at x^e. Then, the equilibrium is globally asymptotically stable.*

∎

[4]This conclusion requires a bit of topology: the characterization of compact sets in terms of "open coverings". If this is new to you, don't worry! In the future you might want to take a first year mathematical analysis course.

The proof is found below.

Prop. 2.3 raises a question: what if the Poisson's inequality is tight, so that the inequality in (2.29) is replaced by equality? Consider this ideal with $\overline{\eta} = 0$, and use the more suggestive notation $V = J^\circ$ for the Lyapunov function:

$$J^\circ(\mathrm{F}(x)) = J^\circ(x) - c(x) \tag{2.31}$$

If J° is non-negative valued, then we can take $V = J^\circ$ in Prop. 2.3 to obtain the upper bound $J(x) \le J^\circ(x)$ for all x. Equality requires further assumptions:

Proposition 2.4. *Suppose that (2.31) holds, along with the following assumptions:*

(i) *J is continuous, inf-compact, and vanishes only at x^e.*

(ii) *J° is continuous.*

Then, $J(x) = J^\circ(x) - J^\circ(x^e)$ for each x.

To establish this result we first require the lemma:

Proof of Lemma 2.1. Similar to Prop. 2.3, we iterate (2.24) to obtain for each $x = x(0)$ and each N,

$$J(x) = J(x(N)) + \sum_{k=0}^{N-1} c(x(k))$$

On taking limits we obtain

$$J(x) = \lim_{N\to\infty} \Big\{ J(x(N)) + \sum_{k=0}^{N-1} c(x(k)) \Big\} = \Big\{ \lim_{N\to\infty} J(x(N)) \Big\} + J(x)$$

which implies (i) under the assumption that $J(x)$ is finite.

As noted after the lemma, the inf-compact assumption in (ii) is imposed to ensure that the state trajectory evolves in a bounded set. Equation (2.28) implies that $x(k) \in S_J(r)$ for the particular value $r = J(x(0))$, and each $k \ge 0$. Suppose that $\{x(k_i) : i \ge 0\}$ is a convergent subsequence of the state trajectory, with limit x^∞. Then, by continuity of J,

$$J(x^\infty) = \lim_{i\to\infty} J(x(k_i)) = 0$$

The assumption that J vanishes only at x^e implies that $x^\infty = x^e$. Part (ii) follows, since every convergent subsequence reaches the same value x^e. ☐

Proof of Prop. 2.3. The bound (2.30) is established following the discussion preceding the proposition. In particular, (2.29) implies

$$V(x(k+1)) - V(x(k)) + c(x(k)) \le \overline{\eta} \tag{2.32}$$

Summing each side from $k = 0$ to $N - 1$ gives (i):

$$V(x(N)) - V(x(0)) + \sum_{k=0}^{N-1} c(x(k)) \le \overline{\eta} N$$

Part (ii) follows since $V(x(N)) \geq 0$ for each N, so that when $\overline{\eta} = 0$ we obtain from the preceding bound,

$$\sum_{k=0}^{N-1} c(x(k)) \leq V(x(0))$$

The proof of (iii) is identical to Lemma 2.1: part (ii) implies that $\lim_{k\to\infty} c(x(k)) = 0$, and the assumptions on c then imply that $x(k) \to x^e$ as $k \to \infty$.

It remains to show that x^e is stable in the sense of Lypapunov. To see this, first observe that with $\overline{\eta} = 0$, the bound (2.29) implies that $V \geq c$, so that V is also inf-compact. The bound (2.29), and conditions on c, $\overline{\eta}$, also imply that $V(x(k))$ is strictly decreasing when $x(k) \neq x^e$. Continuity of V implies that $V(x(k)) \downarrow V(x^e)$ for each $x(0)$, so that $V(x^e) < V(x(0))$ for all $x(0) \in \mathsf{X}$. Stability in the sense of Lypapunov then follows from Prop. 2.2. ☐

Proof of Prop. 2.4. The proof begins with iteration, as in Prop. 2.3:

$$J^\circ(x(N)) + \sum_{k=0}^{N-1} c(x(k)) = J^\circ(x)$$

Lemma 2.1 (ii) and continuity of J° implies that $J^\circ(x(N)) \to J^\circ(x^e)$ as $N \to \infty$, so that

$$J^\circ(x^e) + J(x) = J^\circ(x)$$

☐

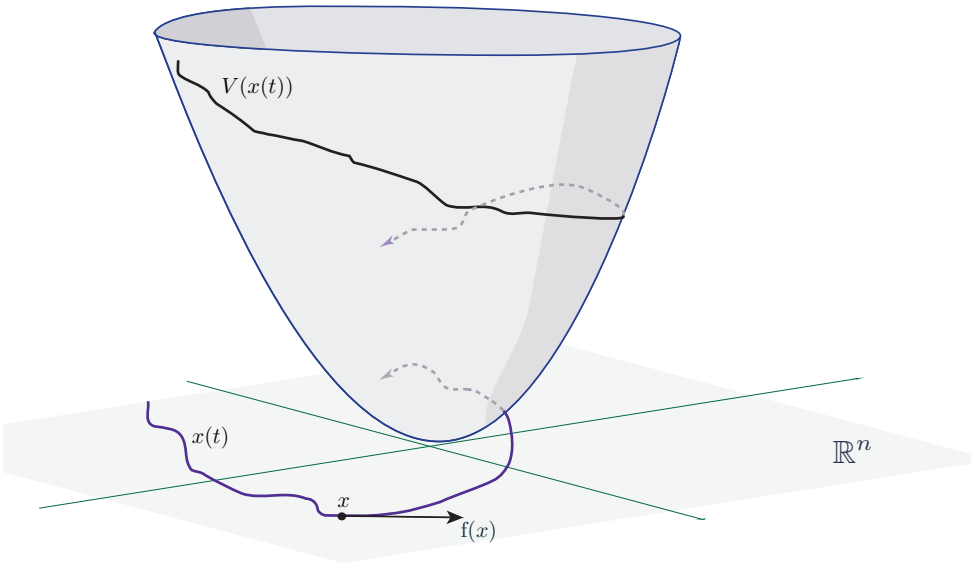


Figure 2.5: If V is a Lyapunov function, then $V(x_t)$ is non-increasing with time.

2.4.3 Geometry in continuous time

Let's briefly consider an analog of (2.20) in continuous time, with state evolving on $\mathsf{X} = \mathbb{R}^n$:

$$\tfrac{d}{dt} x_t = \mathrm{f}(x_t) \tag{2.33}$$

where $\mathrm{f}\colon \mathbb{R}^d \to \mathbb{R}^d$ is called the *vector field*. the use of subscripts is to help distinguish from models in discrete time. It is common to suppress the time index, writing instead $\frac{d}{dt}x = \mathrm{f}(x)$

We let $\mathcal{X}(t; x_0)$ denote the solution to (2.33) at time t, when we need to emphasize dependency on the initial condition $x_0 = x_0$. The definition of asymptotic stability of an equilibrium x^e is the same as Def. 2.1 for the state space model in discrete time, (2.20). The equilibrium is globally asymptotically stable if in addition,

$$\lim_{t\to\infty} \mathcal{X}(t; x_0) = x^e\,, \qquad \text{for all } x_0 \in \mathsf{X}$$

Verification of global asymptotic stability invites the following assumptions for a Lyapunov function, generalizing the theory in discrete time:

(i) V is non-negative valued, differentiable, and its derivative ∇V is continuous.

(ii) It is inf-compact (recall the definition below (2.26)).

(iii) For any solution $\boldsymbol{x}$, whenever $x_t \neq x^e$,

$$\tfrac{d}{dt}V(x_t) < 0. \tag{2.34}$$

Naturally, $\frac{d}{dt}V(x_t) = 0$ if $x_t = x^e$ since in this case $V(x_{t+s}) = V(x^e)$ for all $s \geq 0$.

Fig. 2.5 illustrates the meaning of the vector field f for the special case $\mathsf{X} = \mathbb{R}^2$, and the figure is intended to emphasize the fact that $V(x_t)$ is non-increasing when V is a Lyapunov function. The drift condition (iii) can be expressed in functional form,

$$\langle \nabla V(x), \mathrm{f}(x)\rangle < 0, \qquad x \neq x^e. \tag{2.35}$$

This is illustrated geometrically in Fig. 2.6.

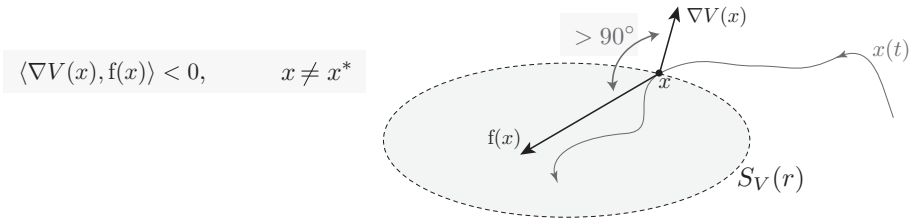


Figure 2.6: Geometric interpretations of a Lyapunov drift condition: the gradient $\nabla V(x)$ is orthogonal to the level set $\{y : V(y) = V(x)\}$, which is the boundary of the set $S_V(r)$ shown, with $r = V(x)$ for the value of x shown.

Proposition 2.5. *If a Lyapunov function exists* (satisfying (i)–(iii) above)*, then the equilibrium x^e is globally asymptotically stable.* □

Prop. 2.5 is a partial extension of Prop. 2.3 to the continuous time model. A full extension requires a version of Poisson's inequality. Suppose that $c\colon \mathbb{R}^n \to \mathbb{R}_+$ is continuous, $V\colon \mathbb{R}^n \to \mathbb{R}_+$ is continuously differentiable, and $\overline{\eta} \geq 0$ is a constant, jointly satisfying

$$\langle \nabla V(x), \mathrm{f}(x)\rangle \leq -c(x) + \overline{\eta}, \qquad x \in \mathsf{X} \tag{2.36}$$

An application of the chain rule implies that this is a continuous time version of Poisson's inequality (2.29):

$$\tfrac{d}{dt}V(x_t) \le -c(x_t) + \overline{\eta}\,, \qquad t \ge 0.$$

And with a bit more work, the following conclusion:

Proposition 2.6. *If (2.36) holds for non-negative $c, V, \overline{\eta}$, then for*

$$V(x_T) + \int_0^T c(x_t)\,dt \le V(x) + T\overline{\eta}, \qquad x_0 = x \in \mathsf{X}\,, \quad T > 0$$

If $\overline{\eta} = 0$ then the total cost is finite:

$$\int_0^\infty c(x_t)\,dt \le V(x), \qquad x_0 = x \in \mathsf{X}. \tag{2.37}$$

Proof. For any $T > 0$, we obtain by the fundamental theorem of calculus,

$$\begin{aligned} -V(x_0) \le V(x_T) - V(x_0) &= \int_0^T \left(\tfrac{d}{dt}V(x_t)\right) dt \\ &\le T\overline{\eta} - \int_0^T c(x_t), \qquad\qquad T \ge 0. \end{aligned}$$

If $\overline{\eta} = 0$, then the bound (2.37) follows on letting $T \to \infty$. ∎

Converse Theorems We have seen this implication:

$$\text{Existence of Lyapunov function} \implies \textit{Stability and/or performance bound}$$

where the nature of stability depends on the nature of the Lyapunov function bound. What about a converse? That is, if the system is stable, can we infer that a Lyapunov function exists?

Assume moreover that the total cost is finite:

$$J(x) = \int_0^\infty c(x_t)\,dt, \qquad x_0 = x$$

with arbitrary initial condition. If J is differentiable, then we obtain a solution to (2.34) using $V = J$:

Proposition 2.7. *If J is finite valued, then for each initial condition x_0 and each t,*

$$\tfrac{d}{dt}J(x_t) = -c(x_t) \tag{2.38}$$

If J is continuously differentiable, the Lyapunov bound (2.34) follows:

$$\nabla J(x) \cdot \mathrm{f}(x) = -c(x)$$

Proof. We have a simple version of Bellman's principle: for any $T > 0$,

$$J(x_0) = \int_0^T c(x_r)\,dr + J(x_T)$$

For $t \geq 0$, $\delta > 0$ given, apply this equation with $T = t + \delta$ and $T = t$:

$$J(x_0) = \int_0^{t+\delta} c(x_r)\, dr + J(x_{t+\delta})$$
$$J(x_0) = \int_0^{t} c(x_r)\, dr + J(x_t)$$

On subtracting, and then dividing by δ, this gives

$$0 = \frac{1}{\delta} \int_t^{t+\delta} c(x_r)\, dr + \frac{1}{\delta}\big(J(x_{t+\delta}) - J(x_t)\big)$$

Letting $\delta \downarrow 0$, the left hand side converges to $c(x_t)$ because $c\colon \mathbb{R}^n \to \mathbb{R}$ is continuous, and the right hand side converges to the derivative of $J(x_t)$ with respect to time, which establishes (2.38). The final conclusion follows from the chain rule. □

2.4.4 Linear state space models

If the dynamics in (2.20) are linear, with $x(k) \in \mathsf{X} = \mathbb{R}^n$, then

$$x(k+1) = Fx(k)\,, \qquad k \geq 0 \tag{2.39}$$

for an $n \times n$ matrix F. Iterating this gives

$$x(k) = F^k x\,, \qquad k \geq 0\,,\ x(0) = x$$

Suppose that the cost is also quadratic, $c(x) = x^\intercal D x$ for a symmetric and positive definite matrix D. It follows that $c(x(k))$ is a quadratic function of $x(0)$ for each k:

$$c(x(k)) = (F^k x)^\intercal D F^k x$$

Hence the value function J defined in (2.23) is also quadratic:

$$J(x) = x^\intercal \Big[\sum_{k=0}^{\infty} (F^k)^\intercal D F^k\Big] x\,, \qquad x(0) = x \in \mathsf{X}$$

That is, $J(x) = x^\intercal M x$, where M is the matrix within the brackets. It satisfies a linear fixed point equation reminiscent of the dynamic programming equation:

$$M = D + F^\intercal M F \tag{2.40}$$

This equation is known as the (discrete-time) *Lyapunov equation*. It admits a solution if all of the eigenvalues of F satisfy $|\lambda(F)| < 1$. .

The continuous time setting is similar. Consider the linear ODE

$$\tfrac{d}{dt} x = Ax \tag{2.41}$$

whose solution is the matrix exponential:

$$x_t = e^{At} x(0)\,, \qquad e^{At} = \sum_{m=0}^{\infty} \frac{1}{m!} t^m A^m \tag{2.42}$$

Consequently, $x_t \to 0$ as $t \to \infty$ from each initial condition if and only if A is *Hurwitz*: each eigenvalue of A has strictly negative real part.

The solution to (2.38) is obtained with a quadratic $J(x) = x^\intercal M^c x$, where the matrix M^c can be found through a bit of linear algebra and calculus. The value function is non-negative, so we may assume M^c is positive semi-definite (hence in particular, symmetric: $M^c = M^{c\intercal}$). Symmetry implies,

$$\tfrac{d}{dt} J(x_t) = 2x_t^\intercal M^c A x_t = x_t^\intercal [M^c A + A^\intercal M^c] x_t$$

and from (2.38) this gives

$$x_t^\intercal [M^c A + A^\intercal M^c] x_t = -c(x_t) = -x_t^\intercal D x_t$$

This must hold for each t and each $x(0)$, giving the Lyapunov equation in continuous time:

$$M^c A + A^\intercal M^c + D \tag{2.43}$$

Example: Euler approximation for the LTI model We consider the linear model in continuous time (2.41). If we sample at regular intervals, with sampling interval Δ, then we obtain a model of the form (2.39), with F equal to the matrix exponential $e^{\Delta A}$:

$$x(\Delta(k+1)) = Fx(\Delta k)$$

An Euler approximation of (2.41) is obtained through the approximation $F = I + \Delta A$, which can be regarded as the approximation of $e^{\Delta A}$: keeping only the first two terms in the Taylor series (2.42). The state $x(k)$ in (2.39) is only approximation of the solution to the ODE (2.41) at time $t = \Delta k$ with this chose of F.

A particular two dimensional example is $A = \left(\begin{smallmatrix} -0.2, & 1 \\ -1, & -0.2 \end{smallmatrix}\right)$. The matrix is Hurwitz, with two eigenvalues $\lambda(A) \approx -0.15 \pm j$. With sampling interval $\Delta = 0.02$, we find that $F = I + \Delta A$ also has two eigenvalues:

$$\lambda(F) = 1 + \Delta\lambda(A) \approx 0.997 \pm 0.02j\,,$$

The eigenvalues satisfy $|\lambda(F)| < 1$, so we see that stability of the discrete-time approximation is inherited from the continuous-time model.

The Matlab command `M = dlyap(F,eye(2))` returns a solution to the Lyapunov equation (2.40) with $D = I$ (the identity matrix):

$$M = \begin{bmatrix} 178.04 & 8.767 \\ 8.767 & 180.49 \end{bmatrix}$$

The fact that F has complex eigenvalues implies that the state process will exhibit rotational motion. A sample paths of $\boldsymbol{x}$ is shown on the left hand side of Fig. 2.7. The trajectory spirals towards the origin, and is intuitively "stable". The plot on the right is a simulation of the linear model subject to a "white noise" disturbance:

$$X(k+1) = FX(k) + N(k+1), \qquad k \ge 0 \tag{2.44}$$

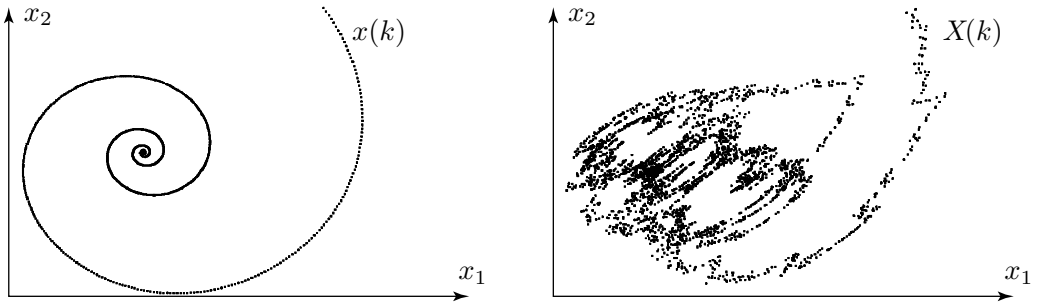


Figure 2.7: At left is a sample path of the deterministic linear model (2.39). At right is a sample path from the linear model with disturbance, (2.44).

Example: frictionless pendulum The *frictionless pendulum* illustrated on the left hand side of Fig. 2.8 is a favorite example in physics and undergraduate control courses. In is based on several simplifying assumptions:

- There is no friction or air resistance
- The rod on which the bob swings is rigid, and without mass
- The bob has mass, but zero volume
- Motion occurs only in two dimensions
- The gravitational field is uniform
- "F = MA" (apply classical mechanics, subject to the foregoing)

A nonlinear state space model is obtained in which x_1 is the angular position θ, and x_2 its derivative:

$$\tfrac{d}{dt}x = \mathrm{f}(x) = \begin{bmatrix} x_2 \\ -\frac{g}{L}\sin(x_1) \end{bmatrix}. \tag{2.45}$$

Shown on the right hand side of Fig. 2.8 are sample trajectories of x_t, and two equilibria.

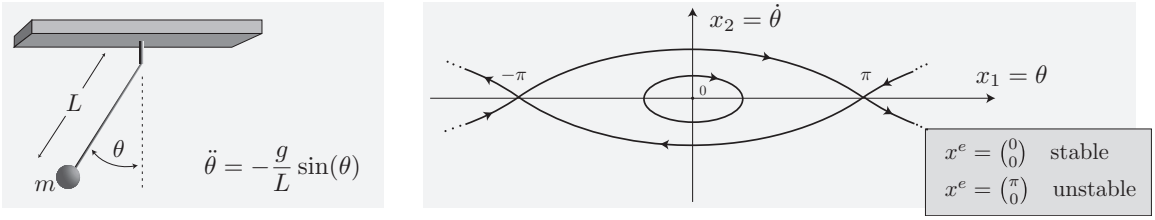


Figure 2.8: Frictionless pendulum: stable and unstable equilibria for the state space model.

An inspection of state trajectories shown on the right hand side of Fig. 2.8 reveals that the equilibrium $x^e = \binom{\pi}{0}$ is not stable in any sense, which agrees with physical intuition (the pendulum is sitting upright in this case). Trajectories which begin near the equilibrium $x^e = 0$ will remain near this equilibrium thereafter.

In fact, the origin is stable in the sense of Lyapunov. To see this, consider a Lyapunov function defined as the sum of potential and kinetic energy:

$$V(x) = \mathrm{PE} + \mathrm{KE} = mgL[1 - \cos(x_1)] + \tfrac{1}{2}mL^2x_2^2$$

The first term is potential energy relative to the equilibrium height, and the second is the classical "KE $= \frac{1}{2}mv^2$" formula for kinetic energy. It is not surprising that V is minimized at $x^e = 0$.

We have $\nabla V(x) = mL^2[(g/L)\sin(x_1)\,,\, x_2]^\intercal$, and

$$\nabla V(x) \cdot \mathrm{f}(x) = mL^2\{(g/L)\sin(x_1) \cdot x_2 - x_2 \cdot (g/L)\sin(x_1) = 0$$

This means that $\frac{d}{dt}V(x_t) = 0$, and hence $V(x_t)$ does not depend on time. For example, the periodic orbit shown in Fig. 2.8 evolves in a level set of V:

$$\frac{g}{L}[1 - \cos(x_1(t))] + \tfrac{1}{2}x_2(t)^2 = \text{const.}$$

From this it is not difficult to show that the origin is stable in the sense of Lyapunov.

Linearization: Using the first-order Taylor series approximation $\sin(\theta) \approx \theta$, the state space equation for the pendulum can be approximated by the LTI model (2.41): $\frac{d}{dt}x = Ax$, with

$$A = \begin{bmatrix} 0 & 1 \\ -g/L & 0 \end{bmatrix}. \tag{2.46}$$

The eigenvalues of A are obtained on solving the quadratic equation

$$0 = \det(I\lambda - A) = \det\Big(\begin{bmatrix} \lambda & -1 \\ g/L & \lambda \end{bmatrix}\Big) = \lambda^2 + g/L \Longrightarrow \lambda = \pm\sqrt{g/L}\, j$$

The matrix is not Hurwitz, since $\mathrm{Re}\,(\lambda(A)) = 0$ for each of the two eigenvalues. Hence the linear approximation, or *linearization*, cannot predict that the equilibrium is stable in the sense of Lyapunov, but the complex eigenvalues are consistent with the periodic behavior of the pendulum.

2.5 A Glance Ahead: From Control Theory to RL

Here is a definition from Wikipedia, as seen on July 2020: *Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.* Here is a translation of some of the key terms:

- **Machine learning** (ML) refers to prediction based on sampled data.
- **Take actions** ≡ feedback. That is, the choice of $u(k)$ for each k based on observations.
- **Software agent** ≡ policy ϕ. This is where the machine learning comes in: the creation of ϕ is based on a large amount of training data collected in "the environment".
- **Cumulative reward** ≡ negative of the sum of cost, such as (2.23), but with the inclusion of the input:

$$\text{Cumulative reward} = -\sum_k c(x(k), u(x(k)))$$

An emphasis in the academic community is truly model-free RL, and most of the theory builds on the optimal control concepts reviewed in the next chapter. Some of the main ideas can be exposed right here.

What follows is background on how RL algorithms are currently formulated. Think hard about alternatives — remember, the field remains young!

2.5.1 Actors and critics

The *actor-critic* algorithm of reinforcement learning is specifically designed within the context of stochastic control, so this is a topic for part 2 of the book. The origins of the terms are worth explaining here. We are given a parameterized family of policies $\{\phi^\vartheta : \vartheta \in \mathbb{R}^d\}$, which play the role of actors. For each ϑ we (or our "software agents") can observe the state process $\boldsymbol{x}$ (or perhaps only an observation process $\boldsymbol{y}$), under the policy $u(k) = \phi^\vartheta(x(k))$. The ideal critic then computes exactly the associated value function J_ϑ, but in realistic situations we have only an estimate.

Since in this book we are minimizing cost rather than maximizing reward, the output of an actor-critic algorithm is the minimum

$$\vartheta^\star = \min_\vartheta \langle \mu, J_\vartheta \rangle \tag{2.47}$$

where $\mu \geq 0$ serves as a state weighting. For a discrete state space this is a sum

$$\langle \mu, J_\vartheta \rangle = \sum_i J_\vartheta(x^i)\mu(x^i)$$

where $\mu(x^i)$ is relatively large for "important states". Methods to solve the optimization problem (2.47) are explored in Section 4.6, using an approach known as *gradient free optimization*. These algorithms are intended to approximate the true gradient descent algorithms of optimization surveyed in Section 4.4.

This is an example of ML: optimizing a complex objective function over a large function class for the purposes of prediction or classification (in this case we are predicting the best policy). A very short introduction to ML can be found in Section 5.1.

2.5.2 Temporal differences

Where do we find a critic? That is, how can we estimate a value function without a model? One answer lies in the sample path representation of the fixed policy Bellman equation, previously announced in (2.28). For any ϑ we have

$$J_\vartheta(x(k)) = c(x(k), u(k)) + J_\vartheta(x(k+1)), \qquad k \geq 0, \quad u(k) = \phi^\vartheta(x(k))$$

We might seek an approximation $\widehat{J}$ for which this identity is well approximated. This motivates the *temporal difference* (TD) sequence commonly used in RL algorithms:

$$\mathcal{D}_{k+1}(\widehat{J}) := -\widehat{J}(x(k)) + \widehat{J}(x(k+1)) + c(x(k), u(k)), \qquad k \geq 0, \quad u(k) = \phi^\vartheta(x(k)) \tag{2.48}$$

After collecting N observations, we obtain the mean-square loss:

$$\mathcal{E}^\varepsilon(\widehat{J}) = \frac{1}{N}\sum_{k=0}^{N-1} \big[\mathcal{D}_{k+1}(\widehat{J})\big]^2 \tag{2.49}$$

We are then faced with another machine learning problem: minimize this objective function over all $\widehat{J}$ in a given class (for example, this is where neural networks frequently play a star

role). It is often easy to solve because $\mathcal{D}_{k+1}(\widehat{J})$ depends linearly on its argument: for any two approximations, and any scalars a, b,

$$\mathcal{D}_{k+1}(a\widehat{J}^1 + b\widehat{J}^2) = a\mathcal{D}_{k+1}(\widehat{J}^1) + b\mathcal{D}_{k+1}(\widehat{J}^2)$$

It follows that $\mathcal{E}^\varepsilon(\widehat{J})$ is a quadratic "functional" (a function whose domain is a set of functions).

It seems we are all done! If we can make (2.49) nearly zero, then we have a good estimate of a value function. Beyond its application to actor-critic methods, there are TD- and Q-learning techniques, designed to minimize (2.49) or a surrogate, that are part of a bigger RL toolbox.

Unfortunately, you have to drop the optimism: *we are not done.*

2.5.3 Bandits and exploration

Suppose that our policy is pretty good. Maybe not optimal in any sense, but $x(k) \to x^e$, $u(k) \to u^e$ rapidly as $k \to \infty$, where the limit is a desirable state-input pair (say, (x^e, u^e) minimizes the one-step cost $c(x, u)$). We typically then have continuity:

$$\lim_{k\to\infty} \big[-\widehat{J}(x(k+1)) + \widehat{J}(x(k)) - c(x(k), u(k))\big] = -\widehat{J}(x^e) + \widehat{J}(x^e) - c(x^e, u^e) = c(x^e, u^e) \quad (2.50)$$

It follows that we aren't observing very much via the temporal difference (2.48). If N is very large then $\mathcal{E}^\varepsilon(\widehat{J}) \approx c(x^e, u^e)^2$. This essentially destroys any hope for a reliable estimate of the value function. Expressed another way: a good policy does not lead to sufficient exploration of the state space.

There are many ways to introduce exploration. We can for example adapt our criterion as follows: denote by $\mathcal{E}^\varepsilon(\widehat{J}; x)$ the mean-square loss obtained with $x(0) = x$. Rather than take a very long run, perform many shorter runs, from many ($M > 1$) initial conditions. The loss function to be minimized is the average

$$L(\widehat{J}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{E}^\varepsilon(\widehat{J}; x^i) \quad (2.51)$$

The most efficient way to choose the samples $\{x^i\}$ is a topic of research. It is a question similar to how to choose μ in (2.47).

Another approach is to let the input do the exploring. The policy is modified slightly through the introduction of "noise":

$$u(k) = \widetilde{\phi}(x(k), \xi(k))$$

For example, $\{\xi(k)\}$ might be a scalar signal, defined as a mixture of sinusoids. The noisy policy is defined so that

(i) $\widetilde{\phi}(x(k), \xi(k)) \approx \phi^\vartheta(x(k))$ for "most k"

(ii) The state process "explores". In particular, the policy is design to avoid convergence of $(x(k), u(k))$ to any limiting value.

This is a crude approach, since by changing the input process, the associated value function also changes. More sensible approaches are contained in Chapters 4 and 5: Q-learning and "off policy SARSA" might be designed around an exploratory policy like this one, but these algorithms are carefully designed to avoid bias from exploration.

The best way to explore is a topic of research, and is mature only within a very special setting: multi-armed bandits. The term "bandit" refers to slot machines: you put money in the machine, pull an arm, and hope that more money pops out. A more rational application is in the advertisement industry, in which an "arm" is an advertisement (which costs money), and the advertiser hopes money will pop out as the ads encourage sales. There is a great history of heuristics and science to create successful algorithms to maximize profit, based only on noisy observations of the performance of candidate ads ([131] is a great reference on the theory of bandits). It is here that the "exploration/exploitation" tradeoff is most clearly seen: you have to accept some loss of revenue through exploration in order to learn the best strategy, and then "exploit" as you gain confidence in your estimates.

The situation is much more complex in control applications: imagine that for each state $x(k)$, there is a multi-armed bandit. "Pulling arm a" at time k means choosing $u(k) = a \in \mathsf{U}$. Concepts from bandit theory have lead to heuristics to best balance the exploration/exploitation tradeoffs arising in RL [175]. This is an exciting direction for future research.

2.6 How Can We Ignore Noise?

If you have never heard the term "random variable", then you can skip this section without concern.

work in progress

2.7 Examples

What follows are toy examples which will be useful for applying the methods to be developed over the course of this book. The models are presented in continuous time because of the elegance of calculus and classical mechanics.

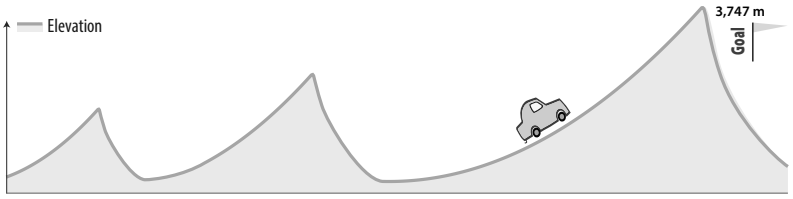


Figure 2.9: Mountain Car

2.7.1 Mountain Car

The goal is to drive a car with a very weak engine to the very top of a very high mountain.

A simple example is illustrated in Fig. 2.9, in which the two dimensional state space is position and velocity:

$$x_t = (z_t, v_t)^\intercal \in \mathsf{X} = [z^{\min}, z^{\text{goal}}] \times [-\overline{v}, \overline{v}]$$

Where $z^{\min}$ is a lower limit for the position z_t, and the target state is z^{goal}. The velocity v_t is bounded in magnitude by $\overline{v} > 0$. The input u is the throttle position (which is negative when

the car is in reverse). Within the RL literature, this example was introduced in the dissertation [156], and has since become a favorite basic example [185, 194].

What makes this problem interesting is that the engine is so weak, that it is impossible to reach the hill directly from some initial conditions. In particular, it may seem sensible to hit the throttle, and head towards the goal at maximum speed. In some cases, the car will stall before reaching the goal. A successful policy will sometimes put the car in reverse, and travel at maximal speed away from the goal to reach a higher elevation to the left. Several cycles back and forth may be required to reach the goal.

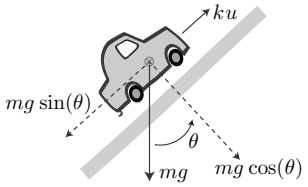


Figure 2.10: Two forces on the Mountain Car

A continuous-time model can be constructed based on the two forces on the car, illustrated in Fig. 2.10. To obtain a simple model, we need to be careful with our notion of distance: $z^{\text{goal}} - z_t$ denotes the *path distance* along the road to the goal, which is not the same as the distance along the *x-axis* in Fig. 2.9. Subject to this convention, Newton's law gives

$$ma = m\frac{d^2}{dt^2}z = -mg\sin(\theta) + ku$$

With state $x = (z, v)^\intercal$, we arrive at the two dimensional state space model,

$$\begin{aligned}\tfrac{d}{dt}x_1 &= x_2\\ \tfrac{d}{dt}x_2 &= \frac{k}{m}u - g\sin(\theta(x_1))\end{aligned} \tag{2.52}$$

where $\theta(x_1)$ is the road grade at $z = x_1$.

An examination of the potential energy $\mathcal{U}$ tells us from which states we can reach the goal without control (setting $u = 0$ in (2.52)). The potential energy is proportional to elevation, and it can be computed by integrating the negative of force, $-F(z)$. For the control-free model we have $-F(z) = g\sin(\theta(z))$, and hence

$$\mathcal{U}(z) = \mathcal{U}(z^{\min}) + \int_{z^{\min}}^{z} g\sin(\theta(z))\,dz \tag{2.53}$$

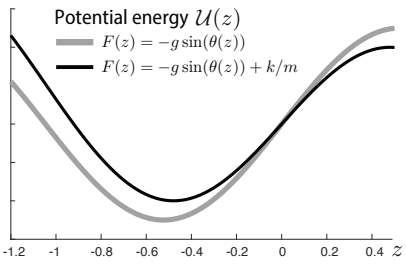


Figure 2.11: Potential energy for Mountain Car.

The version of this model adopted in [194, Ch. 10] uses these numerical values:

$$k/m = 10^{-3}\,, \quad g = 2.5 \times 10^{-3}\,, \quad \theta(z) = \pi + 3z$$

In this case (2.53) gives $\mathcal{U}(z) = \mathcal{U}(z^{\min}) + \frac{1}{3}g\sin(3z)$. Fig. 2.11 shows the potential energy as a function of z on the interval $[z^{\min}, z^{\text{goal}}]$. It has a unique maximum at z^{goal}, which implies that it is necessary to apply external force to reach the goal for any initial condition satisfying $z(0) < z^{\text{goal}}$ and $v(0) \le 0$.

Is the goal reachable? We again examine potential energy. Consider the force as a function of z with $u(k) = 1$ for all k. We obtain $-F(z) = g\sin(\theta(z)) - kz/m$, and the resulting potential energy as a function of z is transformed as shown in Fig. 2.11. We now

have $\mathcal{U}(z^{\min}) > z^{\text{goal}}$, so from $z(0) = z^{\min}$ we will reach the goal with this open-loop control law.

This state space model leaves much out of the navigation problem—namely, wind, traffic, and pot-holes. Also missing from (2.52) are hard constraints on position and velocity assumed at the start. A discrete time model is adopted in [194, Ch. 10] of the form (2.6a): using the notation $x(k) = (z(k), v(k))^\intercal$,

$$z(k+1) = [\![z(k) + v(k)]\!]_1 \tag{2.54a}$$

$$v(k+1) = [\![v(k) + 10^{-3}u(k) - 2.5 \times 10^{-3}\cos(3z(k))]\!]_2 \tag{2.54b}$$

which corresponds to $\theta(z) = \pi + 3z$. The brackets denote projection of the values of $z(k+1)$ to the interval $[z^{\min}, z^{\text{goal}}]$, and $v(k+1)$ to the interval $[-\overline{v}, \overline{v}]$. In numerical experiments, we take

$$z^{\min} = -1.2, \;\; z^{\text{goal}} = 0.5, \;\; \text{and} \;\; \overline{v} = 7 \times 10^{-2}. \tag{2.54c}$$

The state space is also trimmed, so that $v = 0$ whenever $z = z^{\text{goal}}$ (the car is parked once it reaches its target).

Here is an aggressive policy that will get you to the top: whatever direction you are going, accelerate in that direction at maximum rate (provided this is feasible):

$$u(k) = \begin{cases} \text{sign}(v(k)) & z^{\text{goal}} < z(k) < z^{\text{goal}} \\ 0 & z(k) = z^{\text{goal}} \\ 1 & z(k) = z^{\min} \end{cases} \tag{2.55}$$

On considering the evolution of the velocity state (2.54b), you can see this is not a great policy: imagine what happens when $z(0) = z^{\min}$ and $v(0) = -\overline{v}$. It will take many time steps k for the velocity $v(k)$ to become positive.

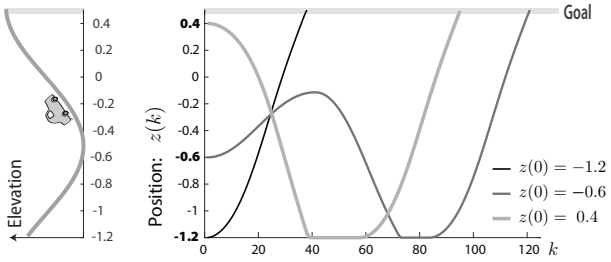


Figure 2.12: Position as a function of time for Mountain Car, from three different initial conditions, using the policy (2.55).

Fig. 2.12 shows results using this policy, from three different initial conditions. The plot starting from $z(0) = 0.4$ tells an unfortunate story: the car moves toward the undesirable "Western hill" with maximal acceleration, and on arrival to the limit $z^{\min} = -1.2$ (just before $k = 40$), the velocity is at its minimum $v(k) = -\overline{v}$. The plot shows that the position remains at $z^{\min}$ for over 20 time steps, since it takes that much time for the velocity to become positive. Better policies will be investigated in Section 4.7, designed to slow down in advance to avoid this delay.

2.7.2 MagBall

The magnetically suspended metal ball illustrated in Fig. 2.13 will be used to illustrate several of the important modeling concepts. In particular, how to transform a set of nonlinear differential equations into a state space model, and how to approximate this by a linear state space model of the form (2.19). Further details from a control systems perspective may be found in the lecture notes [17].

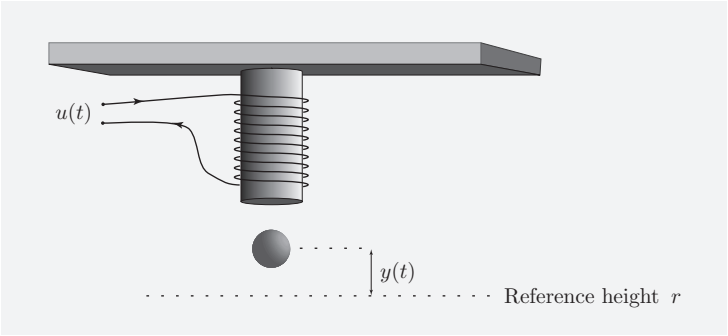


Figure 2.13: Magnetically Suspended Ball

The input u is the current applied to an electro-magnet, and the output y is the distance between the center of the ball and some reference height. Since positive and negative inputs are indistinguishable at the output of this system, it follows that this cannot be a linear system. The upward force due to the current input is approximately proportional to u^2/y^2, and hence from Newton's law for translational motion we have

$$ma \;=\; m\tfrac{d^2}{dt^2}y \;=\; mg - c\frac{u^2}{y^2},$$

where g is the gravitational constant and c is some constant depending on the physical properties of the magnet and ball.

Control design goal: maintain the distance to the magnet at some reference value r. As a first step, we obtain a state space model.

This input-output model can be converted to state space form to obtain something similar to the controllable canonical form description of the ARMA model in (2.16,2.17): using $x_1 = y$ and $x_2 = \frac{d}{dt}y$,

$$\tfrac{d}{dt}x_1 = x_2, \qquad \tfrac{d}{dt}x_2 = g - \frac{c}{m}\frac{u^2}{x_1^2}$$

where the latter equation follows from the formula $\frac{d}{dt}x_2 = \frac{d^2}{dt^2}y$. This pair of equations defines a two-dimensional state space model of the form (2.18):

$$\tfrac{d}{dt}x_1 = x_2 = \mathrm{f}_1(x_1, x_2, u) \tag{2.56a}$$

$$\tfrac{d}{dt}x_2 = g - \frac{c}{m}\frac{u^2}{x_1^2} = \mathrm{f}_2(x_1, x_2, u) \tag{2.56b}$$

It is nonlinear, since f_2 is a nonlinear function of x, and also the state space is constrained: $\mathsf{X} = \{x \in \mathbb{R}^2 : x_1 \geq 0\}$.

Suppose that a fixed current u^e, say positive, is applied, and that the state x^e is an equilibrium:

$$\mathrm{f}(x^e, u^e) = 0.$$

From the definition of f_1 in (2.56a) we must have $x_2^e = 0$, and setting $\mathrm{f}_2(x^e, u^e)$ equal to zero in (2.56b) gives

$$x_1^e = \sqrt{\frac{c}{mg}}u^e \tag{2.57}$$

The negative solution is ignored, since x_1 is restricted to be positive.

If we are *very* successful with our control design, and $x_t = r$ for all t, then we must have

$$u_t = u^e(r)\,, \quad t \geq 0$$

with $u^e(r) = r\sqrt{mg/c}$: the solution to (2.57) with $x^e = r$. Of course, we don't expect that this "open loop" approach will be successful! If we are realistically successful, so that $x_t \approx r$ for all t (perhaps after a transient), then we should expect that $u_t \approx u^e(r)$ as well. The design of a feedback law to achieve this goal is often obtained through an approximate linear model, called a *linearization*.

Linearization about an equilibrium state The linearization is defined exactly as in the frictionless pendulum (2.45). Assume that the signals $x_1(t)$, $x_2(t)$ and u_t remain close to the fixed point (x_1^e, x_2^e, u^e), and write

$$\begin{aligned} x_1(t) &= x_1^e + \delta x_1(t) \\ x_2(t) &= x_2^e + \delta x_2(t) \\ u(t) &= u^e + \delta u(t), \end{aligned}$$

where $\delta x_1(t)$, $\delta x_2(t)$, and δu_t are small-amplitude signals. From the state equations (2.56) we then have

$$\begin{aligned} \tfrac{d}{dt}\delta x_1 &= x_2^e + \delta x_2(t) = \delta x_2(t) \\ \tfrac{d}{dt}\delta x_2 &= \mathrm{f}_2(x_1^e + \delta x_1, x_2^e + \delta x_2, u^e + \delta u) \end{aligned}$$

Applying a first-order Taylor series expansion to the right hand side of the second equation above gives

$$\begin{aligned} \tfrac{d}{dt}\delta x_2 &= \mathrm{f}_2(x_1^e, x_2^e, u^e) + \frac{\partial \mathrm{f}_2}{\partial x_1}\Big|_{(x_1^e, x_2^e, u^e)} \delta x_1 + \frac{\partial \mathrm{f}_2}{\partial x_2}\Big|_{(x_1^e, x_2^e, u^e)} \delta x_2 \\ &\quad + \frac{\partial \mathrm{f}_2}{\partial u}\Big|_{(x_1^e, x_2^e, u^e)} \delta u + d \end{aligned}$$

The final term d represents the error in the Taylor series approximation. After computing partial derivatives we obtain the formulae

$$\begin{aligned} \tfrac{d}{dt}\delta x_1 &= \delta x_2. \\ \tfrac{d}{dt}\delta x_2 &= 2\frac{c}{m}\frac{u^{e2}}{(x_1^e)^3}\delta x_1 - \frac{2c}{m}\frac{u^e}{(x_1^e)^2}\delta u + d \end{aligned}$$

On denoting

$$\alpha = 2\frac{c}{m}\frac{u^{e2}}{(x_1^e)^3}, \qquad \beta = -2\frac{c}{m}\frac{u^e}{(x_1^e)^2}.$$

we obtain a linear state space model with disturbance:

$$\begin{aligned} \tfrac{d}{dt}\delta x &= \begin{bmatrix} 0 & 1 \\ \alpha & 0 \end{bmatrix} \delta x + \begin{bmatrix} 0 \\ \beta \end{bmatrix} \delta u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d \\ \delta y &= \delta x_1\,, \end{aligned} \tag{2.58}$$

There is a hidden approximation in (2.58), since d is in fact a nonlinear function of (x, u). In control design this approximation is taken one step further by setting $d \equiv 0$, to obtain the linear model (2.19). This may not be a useful model for simulations, but often leads to effective control solutions.

2.7.3 CartPole

The next example has a long history within the control systems literature [150, 190, 8], and was introduced to the RL literature in early research of Barto and Sutton [15]. It is today a popular test example on `openai.com`. A history from the perspective of control education can be found in [227], which provides the dynamic equations with state $x = (z, \dot{z}, \theta, \dot{\theta})$, where z is the horizontal position of the cart, and the angle θ is as shown in Fig. 2.14.

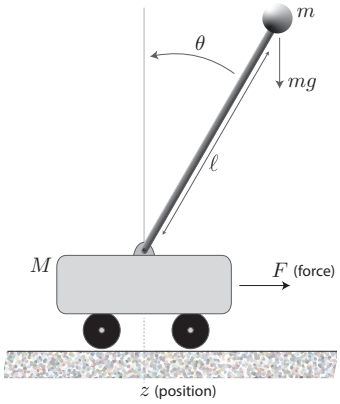


Figure 2.14: CartPole

The goal in this example is regulation: keep $\theta = 0$ while the cart is moving at some desired speed, or some desired fixed position. The aforementioned references describe several successful strategies to swing the pendulum up to a desired position without excessive energy. A normalized model used in [227] is given by

$$\begin{aligned} \tfrac{d}{dt} z &= \tfrac{d}{dt} x_1 = x_2 \\ \tfrac{d}{dt} x_2 &= u \\ \tfrac{d}{dt} \theta &= \tfrac{d}{dt} x_3 = x_4 \\ \tfrac{d}{dt} x_4 &= \sin(x_3) - u \cos(x_3) \end{aligned} \tag{2.59}$$

The state equations are easily linearized near the equilibrium $u^e = 0$ and $x^e = (z^e, 0, 0, 0)^\intercal$ for any z^e: using the first order Taylor series approximations $\sin(x_3) \approx x_3$ and $\cos(x_3) \approx 1$, we obtain as in the derivation of (2.58)

$$\begin{aligned} \tfrac{d}{dt} \delta x_1 &= \delta x_2 \\ \tfrac{d}{dt} \delta x_2 &= u \\ \tfrac{d}{dt} \delta x_3 &= \delta x_4 \\ \tfrac{d}{dt} \delta x_4 &= \delta x_3 - u + d \end{aligned} \tag{2.60}$$

Ignoring the "disturbance" (error term) d, the ODE (2.60) is a version of the state space model (2.19) with

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

The matrix A is not Hurwitz, with eigenvalues at ± 1 and repeated eigenvalues at 0. This was anticipated at the start: it is unlikely that the pendulum will remain upright with a constant "open loop" input, $u_t \equiv 0$.

The linear model is of great value for insight, and designing a linear feedback law to keep the system near the equilibrium:

$$\delta u = -K \delta x$$

Methods to obtain the 4×1 matrix K through optimal control techniques will be investigated later in the book, but there are far better sources to learn about control design for linear systems [17, 125, 1, 43, 9].

In conclusion, we know what to do locally, but the linearization provides no insight whatsoever on how to swing the pendulum up to the desired vertical position. The robotics community has developed ingenious specialized techniques for classes of nonlinear control problems that include CartPole as a special case [190, 8, 227, 47]. A goal of current research is to marry existing control approaches with model free techniques from RL to obtain reliable control designs in more complex settings.

2.7.4 Pendubot and Acrobot

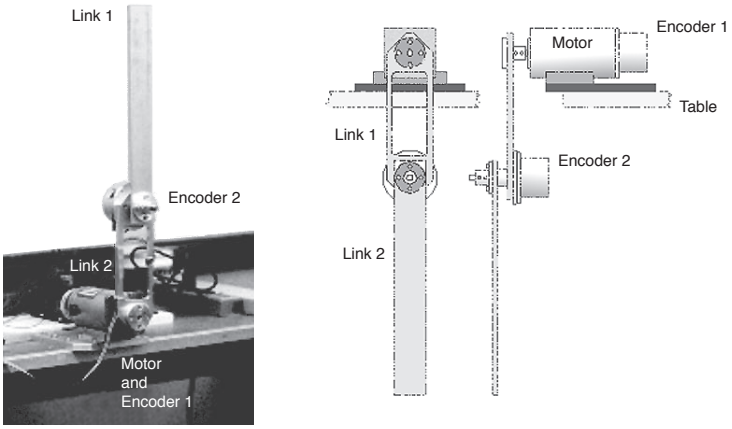
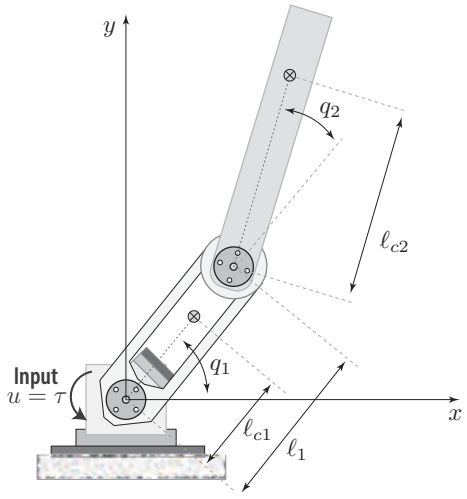


Figure 2.15: The Illinois Pendubot and Sutton's Acrobot

Fig. 2.15 shows a photograph of the *Pendubot* as it appeared in the robotics laboratory at the University of Illinois in the 1990s [189, 17], and a sketch indicating its component parts. It is similar to Sutton's Acrobot [198], which is another example that is currently popular on `openai.com`. The control objective is similar to CartPole: starting from any initial condition, swing the Pendubot up to a desired equilibrium, without excessive energy.

The value of this example is explained in the introduction of [189], where they compare to *CartPole*, and a variation of Furuta [86]:

> "The balancing problem for the Pendubot may be solved by linearizing the equations of motion about an operating point and designing a linear state feedback controller, very similar to the classical cart-pole problem ... One very interesting distinction of the Pendubot over both the classical cart-pole system and Furuta's system is the continuum of balancing positions. This feature of the Pendubot is pedagogically useful in several ways, to show students how the Taylor series linearization is operating point dependent and for teaching controller switching and gain scheduling. Students can also easily understand physically how the linearized system becomes uncontrollable at $q_1 = 0, \pm\pi$." (referring to the first and third illustrations shown in Fig. 2.16, with q_1, q_2 joint angles shown in Fig. 2.17).



$$
\begin{aligned}
d_{11} &= m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}\cos(q_2)) + I_1 + I_2 \\
d_{22} &= m_2\ell_{c2}^2 + I_2 \\
d_{12} &= d_{21} = m_2(\ell_{c2}^2 + \ell_1\ell_{c2}\cos(q_2)) + I_2 \\
h_1 &= -m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_2^2 - 2m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_2\dot{q}_1 \\
h_2 &= m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_1^2 \\
\phi_1 &= (m_1\ell_{c1} + m_2\ell_1)g\cos(q_1) + m_2\ell_{c2}g\cos(q_1 + q_2) \\
\phi_2 &= m_2\ell_{c2}g\cos(q_1 + q_2)
\end{aligned}
$$

Figure 2.17: Coordinate description of the Pendubot: ℓ_1 is the length of the first link, and ℓ_{c1}, ℓ_{c2} are the distances to the center of mass of the respective links. The variables q_1, q_2 are joint angles of the respective links, and the input is the torque applied to the lower joint.

The Pendubot consists of two rigid aluminum links: link 1 is directly coupled to the shaft of a DC motor mounted to the end of a table. Link 1 also includes the bearing housing for the second joint. Two optical encoders provide position measurements: one is attached at the elbow joint and the other is attached to the motor. Note that no motor is directly connected to link 2—this makes vertical control of the system, as shown in the photograph, extremely difficult!

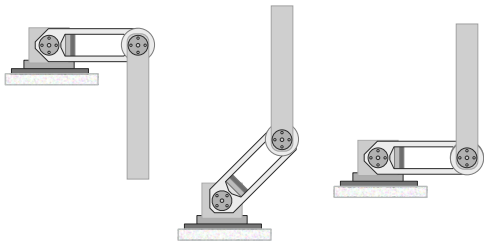


Figure 2.16: A continuum of different equilibrium positions for the Pendubot.

The system dynamics be derived using the so-called Euler-Lagrange equations found in robotics textbooks [191]:

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = \tau \tag{2.61a}$$
$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = 0 \tag{2.61b}$$

where the variables can be deduced from Fig. 2.17. Consequently, this model may be written in state space form, $\dot{x} = \mathrm{f}(x, u)$, where $x = (q_1, q_2, \dot{q}_1, \dot{q}_2)'$, and f is defined from the above equations.

This model admits various equilibria: for example, when $u^e = \tau^e = 0$, the vertical downward position $x^e = (-\pi/2, 0, 0, 0)$ is an equilibrium, as illustrated on the right hand side of Fig. 2.15. Three other possibilities are shown in Fig. 2.16, each with $\tau^e \neq 0$.

A fifth equilibrium is obtained in the upright vertical position, with $\tau^e = 0$ and $x^e = (+\pi/2, 0, 0, 0)$. It is clear from the photograph shown on the left hand side of Fig. 2.15 that the upright equilibrium is strongly unstable in the sense that with $\tau = 0$, it is unlikely that the physical system will remain at rest. Nevertheless, the velocity vector vanishes, $\mathrm{f}(x^e, 0) = 0$, so by definition the upright position is an equilibrium when $\tau = 0$.

Although complex, we may again linearize these equations about the vertical equilibrium. With the input u equal to the applied torque, and the output y equal to the lower link angle, the resulting state space model is defined by the following set of matrices in the 1990's vintage system described in [189]:

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ 51.9243 & 0 & -13.9700 & 0 \\ 0 & 0 & 0 & 1.0000 \\ -52.8376 & 068.4187 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 15.9549 \\ 0 \\ -29.3596 \end{bmatrix} \tag{2.62}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \qquad D = 0.$$

Postscripts For those students who have had a course in undergraduate control systems, the corresponding transfer function has the general form

$$P(s) = k \frac{(s-\gamma)(s+\gamma)}{(s-\alpha)(s+\alpha)(s-\beta)(s+\beta)},$$

with $k > 0$ and $0 < \alpha < \gamma < \beta$. The variable "$s$" corresponds to differentiation. Writing

$$P(s) = k \frac{s^2 - \gamma^2}{s^4 - 2(\alpha^2 + \beta^2)s^2 + \alpha^2\beta^2}$$

the transfer function notation $Y(s) = P(s)U(s)$ denotes the ODE model:

$$\frac{d^4}{dt^4}\delta y - 2(\alpha^2 + \beta^2)\frac{d^2}{dt^2}\delta y + \alpha^2\beta^2\delta y = k[\frac{d^2}{dt^2}u - \gamma^2 u]$$

The roots of the denominator of $P(s)$ are $\{\pm\alpha, \pm\beta\}$, which correspond with the eigenvalues of A. The positive eigenvalues mean that A is not Hurwitz. The fact that $P(s_0) = 0$ for the positive value $s_0 = \gamma$ implies more bad news (a topic far beyond the scope of this book, but the impact of zeros in the right-half plane is worth reading about in basic texts, such as [125, 1, 43, 9]).

2.7.5 Cooperative Rowing

In a sculling boat, each rower has two oars or 'sculls', one on each side of the boat. The control system discussed here concerns coordination of N *individual scullers* (meaning just one rower per boat) that are part of a single team. You can see 5 of N teammates on the left hand side of Fig. 2.18. The team objective is to maintain constant velocity towards a target (let's say, the island of Kaua'i), and also maintain "social distance" between boats.

A state space model might be formulated as follows. Let $z^i(t)$ denote the distance to Kaua'i at time t, and $u_i(t)$ the force exerted by the rower at time t. Taking into account the fact that drag increases with speed, and applying once more Newton's law $F = MA$, results in the following possible dynamical equations

$$\tfrac{d^2}{dt^2} z^i = -a_i \tfrac{d}{dt} z^i + b_i u_i + d^i$$

in which $\{a_i, b_i\}$ are positive scalars, and the disturbance $\{d^i(t)\}$ is left un-modeled. If we ignore the disturbance (for the purposes of control design), we can pose the rowing game as an LQR

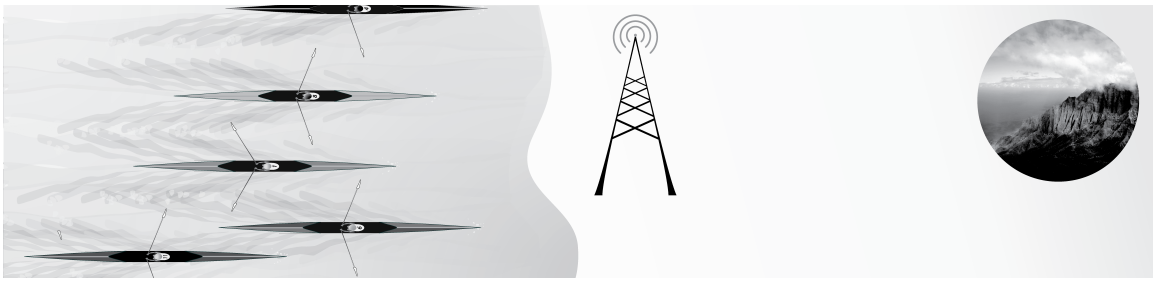


Figure 2.18: Cooperative rowing with partial information.

optimal control problem: a topic covered in Sections 3.1 and 3.6. We will see that this will result in a policy of the form

$$u^i = K^i x + b^i$$

where x is the $2N$-dimensional vector of positions and velocities for all the rowers, K^i is a $2N$-dimensional row vector, and b^i is a scalar that depends upon the tracking goal. Implementation of this policy requires that each rower know the position and velocity of every other rower at each time. Let's think about how the rowers might cooperate without so much data.

Imagine that each rower only views the nearest neighbors to the left and right. This breaks the team of size N into sub-teams of size three that coordinate individually. Unfortunately, if N is large, it is known that this distributed control technique can lead to large oscillations in the positions of the boats with respect to the distant island (the social distancing part is not a problem).

A more robust strategy is obtained with just a bit of global information: assume that at each time t, rower i has access to two scalar observations: her own distance to Kaua'i $z^i(t)$, and the average of all rowers:

$$\bar{z}(t) = \frac{1}{N}\sum_{j=1}^{N} z^j(t)$$

The policy of the ith rower is assumed to be a function of this data. One possibility is to *pretend* that $x^i(t) = (z^i(t), \bar{z}(t))^\intercal$ evolves according to a state space model of the form (2.18), in which case it is appropriate to search for a state feedback policy $u^i(t) = \phi^i(z^i(t), \bar{z}(t))$.

Before fixing the architecture of the policy it is essential to consider the goals. Since we have assumed that social distancing is managed through an independent control mechanism, there remain only two:

$$z^i(t) \approx \bar{z}(t)\,, \qquad \tfrac{d}{dt} z^i(t) \approx v^{\text{ref}} \qquad \textit{for all large } t$$

Based on the discussion in Section 2.3.2 we might obtain better coordination through the introduction of a third variable, defined as the integral of the position error:

$$z^i_a(t) = z^i_a(0) + \int_0^t [z^i(r) - \bar{z}(r)]\, dr$$

Or, to keep things bounded, a discounted approximation:

$$z^i_a(t) = z^i_a(0) + \int_0^t e^{\lambda(t-r)}[z^i(r) - \bar{z}(r)]\, dr$$

with $\lambda > 0$. Once we have made our choice, we then search for a policy defined as a function of the three variables, $u^i(t) = \phi^i(z^i(t), \bar{z}(t), z^i_a(t))$.

However, *do not forget that this is a game!* The "best" choice of ϕ^i will depend upon the choice of ϕ^j for all $j \neq i$. We will experiment with "best response" schemes designed to learn a collection of policies $\{\phi^i : 1 \leq i \leq N\}$ that work well for all. Best response is also behind the RL training in AlphaZero [184], even though Go and chess are obviously *not* cooperative games!

2.8 Exercises

2.1 Something dumb here. state space analysis

2.2 Something dumb here. integral control

Linear algebra:

2.3 Let A be an $n \times n$ matrix, and suppose that the infinite sum exists

$$U = I + A + A^2 + A^3 + \cdots$$

where I denotes the identity matrix. Verify that U is the inverse of the matrix $I - A$

2.4 Two square matrices A and $\bar{A}$ are called *similar* if there is an invertible matrix M such that

$$A = M^{-1}\bar{A}M$$

Obtain the following for two similar matrices A and $\bar{A}$.

(a) Show that A^m is similar to $\bar{A}^m$ for any $m \geq 1$, where the superscript "m" denotes matrix product,

$$A^1 = A, \qquad A^m = A(A^{m-1}), \qquad m \geq 1.$$

(b) Show that v is an eigenvector for A if and only if Mv is an eigenvector for $\bar{A}$.

(c) Suppose that $\bar{A}$ is diagonal ($\bar{A}_{ij} = 0$ if $i \neq j$). Suppose moreover that $|\bar{A}_{ii}| < 1$ for each i. Conclude that $I - A$ admits an inverse by combining Exercise 2.3 and Prob. 2.4 (i).

2.5 *Matrix exponential.* Compute e^{At} for all t for the 2×2 matrix

$$A = aI + bJ\,, \qquad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The notation is intended to be suggestive: $J^2 = -I$.

It is not difficult to obtain a formula for A^m for each m, as required in the definition (2.42). With $a < 0$ and $b \neq 0$, describe the solution to $\frac{d}{dt}x = Ax$ with non-zero initial condition.

Control systems in continuous time For simulating an ODE you might try `ode45` in Matlab; there are many imitations available for use with Python.

2.6 Consider the state space model $\dot{x} = Ax + Bu$; $y = Cx$, where A is similar to a diagonal matrix. In this case we may write

$$\Lambda = V^{-1}AV$$

where Λ is a diagonal matrix, with each $\Lambda(i,i)$ an eigenvalue of A, V is a matrix whose columns are eigenvectors.

(a) Obtain a state space model for $\overline{x} = V^{-1}x$, of the form $\dot{\overline{x}} = \bar{A}\overline{x} + \bar{B}u$; $y = \bar{C}\overline{x}$, by finding repesentations for $(\bar{A}, \bar{B}, \bar{C})$. Comment on why this state space representation is called *modal form*.

The remainder of the problem is numerical, using

$$A = \begin{bmatrix} 8 & -7 & -2 \\ 8 & -10 & -4 \\ -4 & 5 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad C = [1\ 0\ 0]$$

You are encouraged to use Matlab or Python.

(b) Find the eigenvalues and eigenvectors of A, and verify that the matrix $\Lambda = V^{-1}AV$ is indeed diagonal when V is the matrix of eigenvectors.

(c) Obtain the modal-form state space model.

2.7 For the system $\dot{x} = Ax$ with $A = \begin{bmatrix} -1 & 4 \\ 0 & -1 \end{bmatrix}$, show that $V(x) = \|x\|^2 = x_1^2 + x_2^2$ is *not* a Lyapunov function. Find a quadratic function V that is.

2.8 (*Foster's Criterion*) Suppose that $\dot{x} = f(x)$ is a nonlinear state space model on $\mathbb{R}^n$. Assume also that there is a C^1 function $V\colon \mathbb{R}^n \to \mathbb{R}_+$, and a set S such that,

$$\langle \nabla V(\theta), \mathrm{f}(\theta) \rangle \le -1, \qquad \theta \in S^c \tag{2.63}$$

(a) Show that $T_K(x) \le V(x)$ for $x \in \mathbb{R}^n$, where

$$T_K(x) = \min\{t \ge 0 : x_t \in K\}, \qquad x_0 = x \in \mathbb{R}^n.$$

(b) In the special case of a stable linear system ($f(x) = Ax$, with A Hurwitz), show that a solution to (2.63) is given by $V(x) = \log(1 + x^T P x)$ for some matrix $P > 0$, and with $S = \{x : \|x\| \le k\}$ for some scalar k.

(c) Find an explicit V, S for $A = \begin{bmatrix} -1 & 4 \\ 0 & -1 \end{bmatrix}$

Note: Foster devised a version of this stability criterion for countable state space Markov chains 70 years ago

2.9 Consider the nonlinear state space model on the real line,

$$\tfrac{d}{dt}x = \mathrm{f}(x) = \frac{1 - e^x}{1 + e^x}$$

(a) Sketch f as a function of x, and from this plot explain why $x^e = 0$ is an equilibrium, and this equilibrium is globally asymptotically stable.

(b) Find a solution to the Poisson inequality (2.36): $\langle \nabla V, f \rangle \le -c + \bar{\eta}$, with $c(x) = x^2$ and $\bar{\eta} < \infty$. You might try a polynomial, or a log of a polynomial of $|x|$. See if you can find a solution with $\bar{\eta} = 0$.

(c) Find a solution V to Foster's criterion (2.63), with $S = [-k, k]$ for some $k > 0$. Also, discuss why $T_S(x)$ is not finite valued using $S = \{0\}$ (that is, $k = 0$).

2.10 Suppose that one wants to minimize a C^1 function $V\colon \mathbb{R}^n \to \mathbb{R}_+$. A necessary condition for a point $x^\star \in \mathbb{R}^n$ to be a minimum is that it be a *stationary point*: $\nabla V(x^\star) = 0$.

Consider the steepest descent algorithm $\dot{x} = -\nabla V(x)$. Find conditions on the function V to ensure that a given stationary point $x^\star$ will be asymptotically stable for this equation. *One approach*: find conditions under which the function V is a Lyapunov function for this state space model.

2.11 You are given a nonlinear input-output system which satisfies the nonlinear differential equation:

$$\ddot{y}(t) = y^2(u - y) + 2\dot{u}$$

(a) Obtain a two-dimensional nonlinear state-space representation with output y, input u, and states $x_1 = y$ and $x_2 = \dot{y} - 2u$.

(b) Linearize this system of equations around its equilibrium output trajectory when $u(\cdot) \equiv 0$, and write it in state space form.

(c) *For those of you with background in classical control:* Find the transfer function for the linear system obtained in (b).

2.12 Consider the nonlinear state space model on the real line,

$$\tfrac{d}{dt}x = \mathrm{f}(x) = -x^3$$

(a) Sketch f as a function of x, and from this plot explain why $x^e = 0$ is an equilibrium, and this equilibrium is globally asymptotically stable.

(b) Find a solution to the Poisson inequality (2.36) with $c(x) = x^2$: $\langle \nabla V, f \rangle \le -c + \bar{\eta}$ with $\bar{\eta} < \infty$. You might try a polynomial, or a log of a polynomial in of $|x|$. See if you can find a solution with $\overline{\eta} = 0$.

2.13 Consider the Van der Pol oscillator, described by the pair of equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -(1 - x_1^2)x_2 - x_1. \end{aligned} \tag{2.64}$$

(a) Obtain a linear approximate model $\frac{d}{dt}\delta x = A\delta x$ around the unique equilibrium $x^e = 0$.
(b) Verify that A is Hurwitz, and obtain a quadratic Lyapunov function V for the linear model.
(c) Show that V is also a Lyapunov function for (2.64) on the set $S_V(r)$ defined in (2.26), for some $r > 0$. That is, show that the drift inequality (2.34) holds whenever $x_t \in S_V(r)$.

Conclude that the set $S_V(r) \subset \Omega \equiv$ *the region of attraction for* x^e.

(d) *Can we find the entire region of attraction?* Take a box around the origin $B = \{x : -m \le x_1 \le m,\ -m \le x_2 \le m\}$ for some integer m (definitely larger than 1, but less than 10 will suffice). Choose N values $\{x^i\} \subset B$ (say, $N = 10^3$), and simulate the ODE for each i, with $x(0) = x^i$, to test to see if $x_t \in S_V(r)$ for some $t < \infty$, and hence $x^i \in \Omega$.

Why does entry to Ω *guarantee that* $x(0)$ *is in the region of asymptotic stability*?

2.14 (Integral control design). The temperature T in an electric furnace is governed by the linear state equation

$$\tfrac{d}{dt}T = u + w$$

where u is the control (voltage) and w is a constant disturbance due to heat losses. It is not directly observed. It is desired to regulate the temperature to a steady-state value prescribed by the set-point $T = T_0$, where T_0 is your comfort temperature. The following should be solved by hand:
(a) Design a state-plus-integral feedback controller to *guarantee* that $T(t) \to T_0$ as $t \to \infty$, for any constant w. The closed loop poles should have natural frequency $\omega_n \approx 1$ (that is, the eigenvalues of the closed loop state space model should satisfy $|\lambda| \approx 1$.)

(b) To what value does the control u_t converge as $t \to \infty$? Has the controller "learned" w?

2.15 (Linear control design for MagBall). Obtain a linear control design for (2.58), of the form

$$\delta u = -K\delta x = -K_1\delta x_1 - K_2\delta x_2$$

Base your solution on the linearization (2.58), making sure that your solution results in $A - BK$ Hurwitz. Simulate as in Exercise 2.13 to estimate the region of attraction.

2.16 (Feedback linearization for MagBall). For systems with simple nonlinearities, there is a "brute-force" approach to obtain a linear model. For MagBall we may view $\mathbf{v} = u^2/x_1^2$ as an input, from which we obtain a linear system via (2.56):

$$\begin{aligned}\tfrac{d}{dt}x_1 &= x_2\\ \tfrac{d}{dt}x_2 &= g - \frac{c}{m}\mathbf{v}\end{aligned}$$

(a) As in the previous exercise, obtain a control law $v = -K\delta x$, where K_1 and K_2 are parameters chosen for stability and good transient response.

(b) The policy is then given by

$$u = x_1\sqrt{K_1\delta x_1 + K_2\delta x_2} \tag{2.65}$$

Simulate, and estimate the region of attraction.

Note: See [121] for a survey on feedback linearization — a topic that has far more depth than is obvious from this example.

2.9 Notes

Work in progress—comments welcome

The notion of "state" is flexible in both control theory [9] and reinforcement learning [194]. The motivation is the same in each field: for the purposes of on-line decision making, replace the full history of observations at time k by some finite dimensional "sufficient statistic" x_k. One constraint that arises in RL is that the state process must be directly observable; in particular, the belief state that arises in partially observed MDPs requires the (model based) nonlinear filter, and is hence not directly useful for model-free RL. In practice, the "RL state" is specified as some compression of the full history of observations. The reader is referred to [194, Section 17.3] for further discussion.

Textbook treatments on Lyapunov theory can be found in [27] (nonlinear) and [1, 125] (linear). The ECE Department at the University of Illinois had a great course on state space methods—the lecture notes are now available online [17]. The first section of [104] contains a brief crash-course on Lyapunov theory, written in the style of this book, and with applications to reinforcement learning.

Poisson's inequality (2.29) is far removed (roughly two centuries) from the celebrated equation introduced by mathematician Siméon Poisson. The motivation back then was potential theory, as defined in theoretical physics. About one century later, Poisson's equation arose as a central player in studying the evolution of the density of of Brownian motion (a particular Markov process). The terminology *Poisson inequality* and *Poisson equation* is today applied to any Markov chain, with *generator* playing the role of the Laplacian. The generator takes any function $h\colon \mathsf{X} \to \mathbb{R}$ to a new function denoted $\mathcal{A}h$. In particular, the deterministic state space model (2.20) can be regarded as a Markov chain [149], and the associated generator is defined as

$$\mathcal{A}h\,(x) = h(\mathrm{F}(x)) - h(x)$$

In this notation, (2.29) becomes $\mathcal{A}V \le -c + \overline{\eta}$.

Chapter 3

Optimal Control

To begin, we recall some notation: $x(k)$ is the state at time k, which evolves in a state space X; $u(k)$ is input at time k, which evolves in the input (or action) space U (the sets X and U may be Euclidean space, a finite set, or something more exotic). There may also be an output $\boldsymbol{y}$, but this is usually ignored in this chapter. The input and state are related through the dynamical system (2.6a):

$$x(k+1) = \mathrm{F}(x(k), u(k)) \tag{3.1}$$

where $\mathrm{F} \colon \mathsf{X} \times \mathsf{U} \to \mathsf{X}$.

This chapter concerns the design of a state feedback policy $u(k) = \phi(x(k))$ based on optimization. The design of ϕ is based on a cost function c, which is a scalar-valued function of (x, u). We assume throughout that it takes on non-negative values:

$$c \colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}_+$$

The chapter surveys optimization for control systems without assuming any background on the theory of optimization. The term "convex" appears occasionally, but this concept is not essential for understanding any of the material. Optimization theory and the role of convexity will be important later in the book. A brief survey can be found in Section 4.4.

3.1 Value Function for Total Cost

The total cost J associated with a particular control input $\boldsymbol{u} := u_{[0,\infty)}$ is defined by the sum

$$J(\boldsymbol{u}) = \sum_{k=0}^{\infty} c(x(k), u(k))$$

The *value function* is defined to be the minimum over all inputs, and is a function of the initial condition:

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c(x(k), u(k)) \,, \quad x(0) = x \in \mathsf{X} \,. \tag{3.2}$$

The goal of optimal control is to find an optimizing input sequence, and in the process we usually need to compute the value function $J^\star$. We settle for an approximation in the majority of cases.

Why should we care? It is rare in our every-day lives that we think about solving a decision problem over an infinite horizon. It is favored in the control theory literature because an optimal policy often comes with stability guarantees: Thm. 3.1 implies this identity for the optimal input-output process:

$$J^\star(x^\star(k)) = c(x^\star(k), u^\star(k)) + J^\star(x^\star(k+1))$$

which is a version of Poisson's inequality (2.29) with $\overline{\eta} = 0$ (and inequality replaced by equality). Mild conditions laid out in Prop. 2.3 then imply that x^e is globally asymptotically stable under the optimal policy.

What's more, once you understand the total cost formulation, other standard optimal control objectives can be treated as special cases. This is explained in Section 3.3.

Under our assumption that c is non-negative, the value function is also non-negative. Below are minimal assumptions to ensure that $J^\star$ is finite:

(i) There is a target state x^e that is an equilibrium for some input u^e:

$$\mathrm{F}(x^e, u^e) = 0$$

(ii) The cost function c is non-negative, and vanishes at this equilibrium, $c(x^e, u^e) = 0$.

(iii) For any initial condition x_0, there is an input sequence $\boldsymbol{u}_0$ and a time T^0 such that with this initial condition and this input we have $x(T^0) = x^e$

Condition (iii) is a weak form of *controllability*. Under these three assumptions it follows that $J^\star(x) < \infty$ for each x.

The Linear Quadratic Regulator problem refers to optimal control synthesis based on the linear system model (2.13), with quadratic cost:

$$c(x, u) = x^\intercal S x + u^\intercal R u \tag{3.3}$$

It is always assumed that $S \geq 0$ (positive semi-definite) and $R > 0$ (positive definite). If there is one policy for which $J^\star$ is finite valued, then the value function is quadratic, $J^\star(x) = x^\intercal M^\star x$ where $M^\star \geq 0$. The optimal policy is obtained by linear state feedback: $\phi^\star(x) = -K^\star x$ for a matrix $K^\star$ that is a function of $M^\star$ and other system parameters. A bit more on this special case is contained in Section 3.6, where it will be clear why we impose the strict inequality $R > 0$.

3.2 Bellman Equation

To derive the Bellman equation, let x be an arbitrary initial state, and let k_m be an intermediate time, $0 < k_m < \infty$. We regard $J^\star(x(k_m))$ as the *cost to go* at time k_m: This is the optimal total cost over the remaining life-time of the optimal state-trajectory.

Based on this interpretation we obtain,

$$\begin{aligned} J^\star(x) &= \min_{u_{[0,\infty)}} \Big[\sum_{k=0}^{k_m-1} c(x(k), u(k)) + \sum_{k=k_m}^{\infty} c(x(k), u(k)) \Big] \\ &= \min_{u_{[0,k_m]}} \Big[\sum_{k=0}^{k_m-1} c(x(k), u(k)) + \underbrace{\min_{u_{[k_m,\infty)}} \Big(\sum_{k=k_m}^{\infty} c(x(k), u(k)) \Big)}_{J^\star(x(k_m))} \Big] \end{aligned}$$

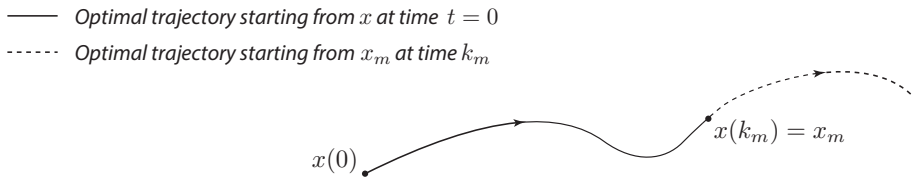


Figure 3.1: If a better control existed on $[k_m, \infty)$, we would have chosen it.

which gives the functional "fixed point equation":

$$J^\star(x) = \min_{u_{[0,k_m-1]}} \Big[\sum_{k=0}^{k_m-1} c(x(k), u(k)) + J^\star(x(k_m)) \Big] \tag{3.4a}$$

As a consequence, the optimal control over the whole interval has the property illustrated in Fig. 3.1: If the optimal trajectory passes through the state x_m at time $x(k_m)$ using the control $\boldsymbol{u}^\star = u_{[0,\infty)}$, then the control $u^\star_{[k_m,\infty)}$ must be optimal for the system starting at x_m at time k_m. If a better $u^\star$ existed on $[k_m, \infty)$, we would have chosen it. This concept is called the *principle of optimality*.

Analysis in continuous time proceeds by letting $k_m \downarrow 0$ to obtain a partial differential equation. Theory is far simpler in discrete time: we set $k_m = 1$ to obtain the following celebrated result.

Theorem 3.1. *If the value function $J^\star$ is finite valued, then it satisfies*

$$J^\star(x) = \min_u \big\{ c(x,u) + J^\star(\mathrm{F}(x,u)) \big\} \tag{3.5}$$

□

Equation (3.5) is often interpreted as a *fixed point equation* in the unknown "variable" $J^\star$. It goes by the name *Bellman equation* or *dynamic programming equation*: the two terms are used interchangeably in this book.

The function of two variables within the minimum in (3.5) is the "Q-function" of reinforcement learning:

$$Q^\star(x,u) := c(x,u) + J^\star(\mathrm{F}(x,u)) \tag{3.6}$$

so that the Bellman equation is equivalent to

$$J^\star(x) = \min_u Q^\star(x,u) \tag{3.7}$$

An important consequence of Thm. 3.1 is that the optimal control can be written in state feedback form, $x^\star(k) = \phi^\star(x^\star(k))$. The feedback law is any minimizer of the Q-function:

$$\phi^\star(x) \in \arg\min_u Q^\star(x,u)\,, \qquad x \in \mathsf{X} \tag{3.8}$$

Another important property is the fixed point equation,

$$Q^\star(x,u) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u)) \tag{3.9}$$

where we denote $\underline{Q}(x) = \min_u Q(x,u)$, $x \in \mathsf{X}$, for any function Q. Eqn. (3.9) is obtained by eliminating $J^\star$ in (3.6) via the identity (3.7). Applications of this dynamic programming equation will appear throughout the book, starting in Section 3.7.

The term *dynamic programming* refers to recursive algorithms designed to obtain the solution to a Bellman equation. However, dynamic programming is only practical when X is finite, or the system has special structure (such as for linear state space models and quadratic cost). The two most popular algorithms are value iteration and policy iteration.

3.2.1 Value iteration

Given an initial approximation V^0 for $V^\star$ appearing in (3.5), a sequence of approximations is defined recursively via

$$V^{n+1}(x) = \min_u \big\{ c(x,u) + V^n(\mathrm{F}(x,u)) \big\}\,, \qquad x \in \mathsf{X},\ n \ge 0 \tag{3.10}$$

Recursions like this to solve fixed point equations are generally known as *successive approximation*. In Exercise 3.3 you will establish the following interpretation:

$$V^{n+1}(x) = \min_{u_{[0,n]}} \Big\{ \sum_{k=0}^{n} c(x(k),u(k)) + V^0(x(n+1)) \Big\}\,, \quad x(0) = x \in \mathsf{X}\,. \tag{3.11}$$

The VIA is convergent under very general conditions: for each x

$$\lim_{n\to\infty} [V^n(x) - V^n(x^e)] = J^\star(x)$$

Here is the simplest result of this kind:

Proposition 3.2. *Suppose that the state space X and input space U are finite, and $J^\star$ is finite valued. Suppose moreover that $c(x,u) > 0$ for $(x,u) \neq (x^e,u^e)$. Then, there is $n_0 \ge 1$ such that for any initialization V^0,*

$$V_n(x) = J^\star(x) - V^0(x^e)\,, \quad x \in \mathsf{X}, \quad n \ge n_0$$

Proof. Let $\phi^\star$ be an optimal policy, and let $n_0 \ge 1$ denote a value such that $(x^\star(k), u^\star(k)) = (x^e, u^e)$ for $k \ge n_0$. Such an integer exists because $J^\star$ is finite valued.

We have from (3.11)

$$V^n(x) \le \sum_{k=0}^{n-1} c(x(k),u(k)) + V^0(x(n))\,, \textit{when } u(k) = \phi^\star(x(k)) \textit{ for each } k, \quad x(0) = x \in \mathsf{X}\,.$$

The right hand side is precisely $J^\star(x) - V^0(x^e)$ for $n \ge n_0$. ∎

In completely general situations the algorithm generates stabilizing policies, subject to an assumption on the initial value function. Denote for $n \ge 0$,

$$g_n(x) = V_{n+1}(x) - V_n(x)\,, \qquad \overline{\eta}_n = \sup_x g_n(x)$$

and define a policy at stage n of the algorithm:

$$\phi^n(x) \in \arg\min_u \big\{c(x,u) + V^n(\mathrm{F}(x,u))\big\}\,, \qquad x \in \mathsf{X},\ n \geq 0$$

The crucial assumption on V^0 is a finite bound, which is interpreted as Lyapunov bound in the proof of Prop. 3.3 that follows: the function V^0 is non-negative, and satisfies for some $\overline{\eta} \geq 0$,

$$\min_u \big\{c(x,u) + V^0(\mathrm{F}(x,u))\big\} \leq V^0(x) + \overline{\eta}\,, \qquad x \in \mathsf{X} \tag{3.12}$$

The conclusions of Prop. 3.3 are most interesting when $\overline{\eta} = 0$, so that for each n:

$$\big\{c(x,u) + V^n(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^n(x)} \leq V^n(x)$$

The following bound then follows from Prop. 2.3,

$$J^n(x) \leq V^n(x)\,, \qquad x \in \mathsf{X}\,,$$

where J^n is the total cost using policy ϕ^n.

Proposition 3.3. *Suppose that (3.12) holds, with V^0 non-negative. That is, there is a policy ϕ^{-1} for which V^0 serves as a Lyapunov function:*

$$\big\{c(x,u) + V^0(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^{-1}(x)} \leq V^0(x) + \overline{\eta}\,, \qquad x \in \mathsf{X}$$

Then a similar bound holds for each n:

$$\big\{c(x,u) + V^n(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^n(x)} \leq V^n(x) + \overline{\eta}_n\,, \qquad x \in \mathsf{X}$$

Moreover, the upper bounds are finite and non-increasing:

$$\overline{\eta} \geq \overline{\eta}_0 \geq \overline{\eta}_1 \geq \cdots$$

Proof. The Lyapunov bound follows from adding and subtracting terms in (3.10):

$$\big\{c(x,u) + V^n(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^n(x)} = V^{n+1}(x) = V^n + g_n \leq V^n + \overline{\eta}_n$$

It remains to obtain bounds on $\{\overline{\eta}_n\}$. First observe that

$$V^1(x) \leq \big\{c(x,u) + V^0(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^{-1}(x)} \leq V^0(x) + \overline{\eta}$$

from which we conclude that $g_1(x) \leq \overline{\eta}$ for all x, and hence also $\overline{\eta}_1 \leq \overline{\eta}$.

The next steps are similar: for $n \geq 1$,

$$V^{n+1}(x) \leq \big\{c(x,u) + V^n(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^{n-1}(x)}$$
$$V^n(x) = \big\{c(x,u) + V^{n-1}(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^{n-1}(x)}$$

Hence on subtracting,

$$g_{n+1}(x) = V^{n+1}(x) - V^n(x) \leq \big\{V^n(\mathrm{F}(x,u)) - V^{n-1}(\mathrm{F}(x,u))\big\}\Big|_{u=\phi^{n-1}(x)} \leq \overline{\eta}_{n-1}$$

□

3.2.2 Policy Improvement

The Policy Improvement Algorithm (PIA) starts with an initial policy ϕ_0, and updates recursively as follows:

For policy ϕ_n, the associated cost is computed:

$$J^n(x) = \sum_{k=0}^{\infty} c(x(k), u(k)) \,, \qquad u(j) = \phi_n(x(j)) \;\; \textit{for each } j \,, \qquad x(0) = x \in \mathsf{X} \tag{3.13}$$

This solves the fixed-policy Bellman equation,

$$J^n(x) = \big\{ c(x,u) + J^n(\mathrm{F}(x,u)) \big\}\Big|_{u=\phi_n(x)} \tag{3.14}$$

The $(n+1)$th iteration of PIA is completed with the policy improvement step:

$$\phi_{n+1}(x) = \arg\min_u \{ c(x,u) + J^n(\mathrm{F}(x,u)) \} \tag{3.15}$$

The proof of the following is similar to the proof of Prop. 3.3. The fact that the value functions are non-increasing is again an application of Prop. 2.3.

Proposition 3.4. *Suppose that ϕ^0 is stabilizing, in the sense that J^0 is finite valued. Then for each $n \ge 0$,*

$$\big\{ c(x,u) + J^n(\mathrm{F}(x,u)) \big\}\Big|_{u=\phi^{n+1}(x)} \le J^n(x) \,, \qquad x \in \mathsf{X}$$

Consequently, the value functions are non-increasing:

$$J^0(x) \ge J^1(x) \ge J^2(x) \ge \cdots$$

□

3.2.3 Perron-Frobenius theory – a gentle introduction*

One step in the PIA requires a subroutine: how to solve the fixed policy dynamic programming equation (3.14)? This is a question that may have been posed in Chapter 2 where we introduced the fixed policy value function J defined in (2.23). The purpose here is to present an efficient approach to computing the value function when the state space is finite; this is also a prelude to theory for Markov chains as well as spectral graph theory that arises in MDPs and ML.

Let's return for a moment to the setting of Section 2.4, where we considered the state space model without control:

$$x(k+1) = \mathrm{F}(x(k)) \,, \qquad k \ge 0$$

and associated value function (2.23), recalled here:

$$J(x) = \sum_{k=0}^{\infty} c(x(k)) \,, \qquad x(0) = x \in \mathsf{X}$$

The value function satisfies a version of (3.14), which in equation (2.24) is expressed in the simpler form

$$J(x) = c(x) + J(\mathrm{F}(x)) \,, \qquad x \in \mathsf{X} \tag{3.16}$$

Much of Perron-Frobenius theory concerns calculation of fixed-point equations involving matrices. To apply this theory, we need a matrix. Assume the state space is finite, and to simplify notation suppose that the state space is a sequence of positive integers: $\mathsf{X} = \{1, 2, 3, \ldots, N\}$ for some $N > 1$. Assume that $x^e = N$, which satisfies $\mathrm{F}(N) = N$ by the equilibrium property. Assume also that $c(N) = 0$, and that the value function is finite valued.

Definite an $N \times N$ *transition matrix* P, based on the dynamical system as follows: $P(i, j) = 0$ or 1 for each i and j, and $P(i, j) = 1$ means that $j = \mathrm{F}(i)$. Consequently, the ith row of P has exactly one non-zero element. In particular, $P(N, N) = 1$ characterizes the equilibrium property. With this notation, we have a new way of thinking about the fixed policy dynamic programming equation:

$$J(i) = c(i) + \sum_{j=1}^{N} P(i, j)J(j) \tag{3.17}$$

Now, dear reader: *please accept a new way of thinking about the notation:*

$$\vec{J} = \vec{c} + P\vec{J} \tag{3.18}$$

I hope the notation is clear: P is an $N \times N$ matrix, $\vec{J}$ is an N-dimensional column vector whose ith element is $J(i)$, and the definition of $\vec{c}$ is analogous. I am pleading with you here, because I know from experience that young graduate students feel uncomfortable going from (3.16) to (3.18).

At first glance, it seems clear that we can solve this equation by inversion:

$$\vec{J} = [I - P]^{-1}\vec{c}$$

The problem however is that $I - P$ is never invertible, since P always has an eigenvalue $\lambda = 1$: whenever $v \in \mathbb{R}^N$ has constant entries ($v(i) = v(1)$ for all i), we have

$$Pv = v$$

You might try a pseudo inverse. In Matlab, this is computed using the command

```
J = (I-P)\c
```

But if you do this, you may not understand what is going on behind Matlab's curtain. And how do you know if you have obtained the boundary constraint $J(x^e) = 0$?

Here is the ingenious idea of Perron and Frobenius: choose two vectors $s, \nu \in \mathbb{R}^N$ with non-negative entries, and satisfying,

$$P(i, j) \geq s(i)\nu(j) \qquad 1 \leq i, j \leq N \tag{3.19}$$

This is called a *minorization condition*. Letting $s \otimes \nu$ denote the "outer product" of these two vectors, this is equivalently expressed

$$P(i, j) \geq [s \otimes \nu](i, j) \qquad 1 \leq i, j \leq N$$

We then play with the fixed point equation:

$$\vec{c} = [I - P]\vec{J} = [I - (P - [s \otimes \nu])]\vec{J} + [s \otimes \nu]\vec{J}$$

and note that the final term is just a constant times s, represented as a column vector:

$$[s \otimes \nu]\vec{J} = \delta s\,, \qquad \delta = \sum_j \nu(j)J(j)$$

Under very mild conditions we can now invert: denote

$$Z = \sum_{n=0}^{\infty} (P - [s \otimes \nu])^n \tag{3.20}$$

with $(P - [s \otimes \nu])^k$ the kth power of the difference for $k \geq 1$, and with $(P - [s \otimes \nu])^0 = I$ (the identity matrix).

Here is where the minorization condition comes in: the matrix $(P - [s \otimes \nu])^k$ has non-negative entries for each $k \geq 0$, so that the infinite sum is always meaningful. If it is finite valued, then $Z = [I - (P - [s \otimes \nu])]^{-1}$, and consequently

$$\vec{J} = Z\vec{c} - \delta Z s$$

To find δ you must apply the boundary condition for J:

$$0 = J(N) = \sum_k Z(N,k)c(k) - \delta \sum_k Z(N,k)s(k)$$

and then obtain δ by division.

Alternatively, think harder about your choice of ν! Here is a simple consequence of the Perron-Frobenius construction:

Proposition 3.5. *Consider the state space model with $\mathsf{X} = \{1, 2, 3, \ldots, N\}$. Suppose that $c\colon \mathsf{X} \to \mathbb{R}_+$ vanishes only at the state N, and suppose that the total cost J is finite valued. Define the matrix Z using $s = \nu = e^N$* (the Nth basis vector in $\mathbb{R}^N$).

Then, $\vec{J} = Z\vec{c}$. That is, for each $k \in \mathsf{X}$,

$$J(k) = \sum_{j=1}^{N} Z(k,j)c(j) = \sum_{n=0}^{\infty}\sum_{j=1}^{N} (P - [s \otimes \nu])^n(k,j)c(j)$$

Proof. The minorization condition holds because $P(N,N) = 1 = [s \otimes \nu](N,N)$, and $[s \otimes \nu](i,j) = 0$ for all other i, j. Applying the boundary constraint $J(N) = 0$ gives

$$\delta = \sum_j \nu(j)J(j) = J(N) = 0$$

To see that Z is finite valued we establish an interpretation for each term in the sum: for $j < N$,

$$(P - [s \otimes \nu])^n(i,j) = \mathbf{1}\{x(n) = j\}\,, \qquad \text{when } x(0) = i$$

The left hand size is zero for $j = N$ and $k \geq 1$. Letting $n_0 \geq 1$ denote an integer for which $x(n) = N$ for $n \geq n_0$, it follows that Z can be expressed as a finite sum:

$$Z = \sum_{n=0}^{n_0} (P - [s \otimes \nu])^n$$

□

3.3 Variations

The total cost problem (3.2) is the standard in the control literature, and opens the door to many other possibilities.

Discounted cost A more popular objective within the operations research literature is the *discounted-cost* problem:

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} \gamma^k c(x(k), u(k)) \,, \quad x(0) = x \in \mathsf{X} \,. \tag{3.21}$$

with $\gamma \in (0, 1)$ the *discount factor*. The Q-function becomes $Q^\star(x, u) := c(x, u) + \gamma J^\star(\mathrm{F}(x, u))$, and the Bellman equation has the same form (3.7).

Shortest path problem Given a subset $A \subset \mathsf{X}$, define

$$\tau_A = \min\{k \geq 1 : x(k) \in A\}$$

The discounted shortest path problem (SPP) is defined to be the minimal discounted cost incurred before reaching the set A:

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\tau_A - 1} \gamma^k c(x(k), u(k)) \tag{3.22}$$

For the purposes of unifying the control techniques that follow, it is useful to recast this as an instance of the total cost problem (3.2). This requires the definition of a new state process $\boldsymbol{x}^A$ with dynamics F^A, and a new cost function c^A defined as follows:

(i) The modified state dynamics:

$$\mathrm{F}^A(x, u) = \begin{cases} \mathrm{F}(x, u) & x \in A^c \\ x & x \in A \end{cases}$$

so that $x^A(k+1) = x^A(k)$ if $x^A(k) \in A$ (called a *graveyard set* for the control system).

(ii) Modified cost function:

$$c^A(x, u) = \begin{cases} c(x, u) & x \in A^c \\ 0 & x \in A \end{cases}$$

From these definitions it follows that the value function (3.22) can be expressed

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} \gamma^k c^A(x^A(k), u(k)) \,, \qquad x \in A^c$$

Alternatively, we can obtain a dynamic programming equation by writing

$$J^\star(x) = \min_{\boldsymbol{u}} \Big\{ c(x, u(0)) + \sum_{k=1}^{\tau_A - 1} \gamma^k c(x(k), u(k)) \Big\}$$

with the understanding that $\sum_1^0 = 0$. The upper limit in the sum is equal to 0 when $\tau_A = 1$; equivalently, $x(1) \in A$. Hence,

$$\begin{aligned} J^\star(x) &= \min_{u(0)} \Big\{ c(x,u(0)) + \gamma \mathbf{1}\{x(1) \in A^c\} \min_{u_{[1,\infty]}} \sum_{k=1}^{\tau_A - 1} \gamma^{k-1} c(x(k),u(k)) \Big\} \\ &= \min_{u(0)} \Big\{ c(x,u(0)) + \gamma \mathbf{1}\{x(1) \in A^c\} J^\star(x(1)) \Big\}, \qquad x(1) = \mathrm{F}(x,u(0)) \\ &= \min_{u} \Big\{ c(x,u) + \gamma \mathbf{1}\{\mathrm{F}(x,u) \in A^c\} J^\star(\mathrm{F}(x,u)) \Big\} \end{aligned} \tag{3.23}$$

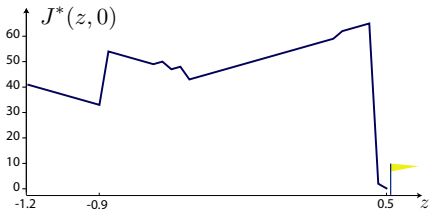


Figure 3.2: Value function for Mountain Car, for states $x = (z,0)^\intercal$ (the initial velocity is zero).

Mountain Car Recall the Mountain Car example introduced in Section 2.7.1. The control objective is to reach the goal in minimal time, but can also be case as a total cost optimal control problem without discounting. Let $c(x,u) = 1$ for all x, u with $x \neq z^{\text{goal}}$, and $c(z^{\text{goal}}, u) \equiv 0$. Given that the car is parked upon reaching the goal, it is natural to modify the dynamics to impose this constraint: $\mathrm{F}(z^{\text{goal}}, u) = z^{\text{goal}}$ for any u. The optimal total cost (3.2) is finite for each initial condition, and the Bellman equation (3.5) becomes

$$J^\star(x) = 1 + \min_u \big\{ J^\star(\mathrm{F}(x,u)) \big\}, \quad x_1 < z^{\text{goal}}$$

and with $J^\star(z^{\text{goal}}, x_2) = 0$ for any value of x_2.

Fig. 3.2 shows the value function for initial conditions corresponding to the car starting at rest (corresponding to $v = x_2 = 0$). The total cost is relatively low with initial condition $x_0 = (z,0)$ with $z \leq -0.9$ because the car can reach the goal without stalling.

Finite horizon Your choice of discount factor is based on how concerned you are with the distant future. Motivation is similar for the *finite horizon* formulation: fix a horizon $\mathcal{T} \geq 1$, and denote

$$J^\star(x) = \min_{u_{[0,\mathcal{T}]}} \sum_{k=0}^{\mathcal{T}} c(x(k),u(k)), \quad x(0) = x \in \mathsf{X}. \tag{3.24}$$

This can be interpreted as the total cost problem (3.2), following two modifications of the state description and the cost function, similar to the SPP:

(i) Enlarge the state process to $x^{\mathcal{T}}(k) = (x(k), \tau(k))$, where the second component is "time" plus an offset:

$$\tau(k) = \tau(0) + k, \qquad k \geq 0$$

(ii) Extend the definition of the cost function as follows:

$$c^{\mathcal{T}}((x,\tau),u) = \begin{cases} c(x,u) & \tau \leq \mathcal{T} \\ 0 & \tau > \mathcal{T} \end{cases}$$

That is, $c^{\mathcal{T}}((x,\tau),u) = c(x,u)\mathbf{1}\{\tau \le \mathcal{T}\}$ for all x, τ, u.

If these definitions are clear to you, then you understand that we have succeeded in the transformation:

$$J^\star(x) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c^{\mathcal{T}}(x^{\mathcal{T}}(k), u(k))\,, \quad x^{\mathcal{T}}(0) = (x,\tau)\,,\ \tau = 0 \tag{3.25}$$

However, to write down the Bellman equation it is necessary to consider all values of τ (at least values $\tau \le \mathcal{T}$), and not just the desired value $\tau = 0$. Letting $J^\star(x,\tau)$ denote the right hand side of (3.25) for arbitrary values of $\tau \ge 0$, the Bellman equation (3.5) becomes

$$J^\star(x,\tau) = \min_u \big\{ c(x,u)\mathbf{1}\{\tau \le \mathcal{T}\} + J^\star(\mathrm{F}(x,u), \tau+1) \big\} \tag{3.26}$$

The similarity with (3.10) is explored in Exercise 3.3.

Based on (3.25) and the definition of $c^{\mathcal{T}}$, we know that $J^\star(x,\tau) \equiv 0$ for $\tau > \mathcal{T}$. This is considered a *boundary condition* for the recursion (3.26), which is put to work as follows: first, since $J^\star(x, \mathcal{T}+1) \equiv 0$,

$$J^\star(x,\mathcal{T}) = \underline{c}(x) := \min_u c(x,u)$$

Applying (3.26) once more gives,

$$J^\star(x,\mathcal{T}-1) = \min_u \big\{ c(x,u) + \underline{c}(\mathrm{F}(x,u)) \big\}$$

If the state space X is finite then these steps can be repeated until we obtain the value function $J^\star(\,\cdot\,, 0)$.

What about the policy? It is again obtained via (3.26), but the optimal input depends on the extended state, based on the policy

$$\phi^\star(x,\tau) = \arg\min_u \big\{ c(x,u) + J^\star(\mathrm{F}(x,u), \tau+1) \big\}\,, \qquad \tau \le \mathcal{T}$$

This means that the feedback is no longer time-homogeneous:[5]

$$u^\star(k) = \phi^\star(x^\star(k), \tau^\star(k)) = \phi^\star(x^\star(k), k) \tag{3.27}$$

Model predictive control Perhaps the most successful control technique in manufacturing and building operations is model predictive control (MPC). This is a slight variant of (3.27) to obtain a *stationary* policy:

$$u(k) = \phi^{\mathrm{MPC}}(x^\star(k)) = \phi^\star(x(k), \mathcal{T}) \tag{3.28}$$

However, MPC is never presented as state feedback because the policy ϕ^{MPC} is not computed and stored in memory. Rather, for each k, when the state $x = x(k)$ is observed, the finite horizon optimization is performed to obtain the value $u = \phi^\star(x, \mathcal{T})$. That is, the policy is only evaluated for those states that are observed [140, 141].

[5] Substituting $\tau^\star(k) = k$ is justified because we cannot control time!

Nevertheless, the general optimal control theory provides techniques to ensure that the total cost associated with (3.28) is finite. Denote for $x \in \mathsf{X}$,

$$J^{\text{MPC}}(x) = \sum_{k=0}^{\infty} c(x(k), u(k))$$

subject to $x(0) = x$, and $u(k) = \phi^{\text{MPC}}(x(k))$ for all k. The following result is a corollary to Prop. 3.3, after recognizing that $J^\star(x; \mathcal{T}) = V^{\mathcal{T}}(x)$.

Proposition 3.6. *Consider the policy (3.28), obtained with modified objective:*

$$J^\star(x; \mathcal{T}) = \min_{u_{[0,\mathcal{T}]}} \sum_{k=0}^{\mathcal{T}-1} c(x(k), u(k)) + V_0(x(\mathcal{T})), \quad x(0) = x \in \mathsf{X}, \tag{3.29}$$

where $V_0 \colon \mathsf{X} \to \mathbb{R}_+$ satisfies (3.12) with $\overline{\eta} = 0$:

$$\min_u \big\{ c(x,u) + V^0(\mathrm{F}(x,u)) \big\} \le V^0(x), \qquad x \in \mathsf{X}$$

Then, the total cost under ϕ^{MPC} is everywhere finite, and admits the bound

$$J^{\text{MPC}}(x) \le J^\star(x; \mathcal{T})$$

3.4 Inverse Dynamic Programming

An alternative to dynamic programming is to change the problem: in optimal control we are given c, and then face the (often daunting) task of computing $J^\star$. Why not reverse the problem? Given any function J, find a cost function c_J so that (3.5) is satisfied? To proceed in a way that respects our original goal, we introduce the *Bellman error*:

$$\mathcal{B}(x) = -J(x) + \min_u [c(x,u) + J(\mathrm{F}(x,u))] \tag{3.30}$$

This is precisely the error in the Bellman equation (3.5). Based on this we obtain a solution to a Bellman equation, with modified cost function:

$$J(x) = \min_u [c_J(x,u) + J(\mathrm{F}(x,u))] \tag{3.31a}$$

$$c_J(x,u) = c(x,u) - \mathcal{B}(x) \tag{3.31b}$$

The minimizer in (3.31a) defines a policy, denoted $\phi^J(x)$.

This procedure is known as *Inverse Dynamic Programming*. It is is one formulation of the *control-Lyapunov function* approach to control design. Minimizing the Bellman error is a goal of many approaches to reinforcement learning. Motivation is provided in the following:

Proposition 3.7. *Suppose that the following hold:*

(i) *J is non-negative, continuous, and vanishes only at x^e.*

(ii) *The function $d_J(x) = c(x, \phi^J(x))$ is also non-negative, continuous, and vanishes only at x^e. Moreover, it is inf-compact.*

(iii) *There is a constant* ϱ *satisfying* $0 \le \varrho < 1$, *and*

$$\mathcal{B}(x) = c(x,u) - c_J(x,u) \ge -\varrho c(x,u) \qquad \textit{for all } x, u$$

Let J^{ϕ^J} *denote* the value function under the policy ϕ^J:

$$J^{\phi^J}(x) = \sum_{k=0}^{\infty} c(x(k), u(k)), \qquad x(0) = x, \quad u(k) = \phi^J(x(k)) \quad \textit{for all } k$$

Then, the performance of ϕ^J *admits the following bounds:*

$$J^\star(x) \le J^{\phi^J}(x) \le (1+\varrho) J^\star(x)$$

The proof of Prop. 3.7 requires a deeper look at the dynamic programming equation (3.5), whose solution is typically unique. The following is an extension of Prop. 2.4:

Proposition 3.8. *Suppose that the value function* $J^\star$ *is finite valued, and the optimal policy* $\phi^\star$ *is stabilizing, in the sense that* $x^\star(k) \to x^e$ *as* $k \to \infty$ *for any initial condition.*

Suppose that $J \colon \mathsf{X} \to \mathbb{R}_+$ *is continuous, vanishes only at* x^e, *and solves*

$$J(x) \le \min_u \big\{ c(x,u) + J(\mathrm{F}(x,u)) \big\}, \quad x \in \mathsf{X} \tag{3.32}$$

Then $J = J^\star$.

Proof. Let $\phi^J(x)$ denote a minimizer of (3.32), and denote by $H(x)$ the value function with this policy:

$$H(x) = \sum_{k=0}^{\infty} c(x(k), u(k)), \qquad x(0) = x, \quad u(k) = \phi^J(x(k)) \quad \textit{for all } k$$

We have the bound $H \le J$ by the Comparison Theorem, Prop. 2.3. We can also establish the following bound by induction on $\mathcal{T}$:

$$\begin{aligned} J(x) &\le \min_{u_{[0,\mathcal{T}]}} \Big\{ \sum_{k=0}^{\mathcal{T}-1} c(x(k), u(k)) + J(x(\mathcal{T})) \Big\} \\ &\le \sum_{k=0}^{\mathcal{T}-1} c(x^\star(k), u^\star(k)) + J(x^\star(\mathcal{T})), \qquad u^\star(k) = \phi^\star(x^\star(k)) \quad \textit{for all } k \end{aligned}$$

with the second inequality obtained because we have replaced the minimum with a specific policy. We have $J(x^\star(\mathcal{T})) \to 0$ as $\mathcal{T} \to \infty$ by the assumptions on J and $\phi^\star$, and hence $J \le J^\star$.

Putting the two bounds together gives for all x,

$$J^\star(x) \le H(x) \le J(x) \le J^\star(x)$$

□

Proof of Prop. 3.7. The assumptions on J and d_J are imposed so that we may apply Prop. 2.3 for the state space model subject to $u(k) = \phi^J(x(k))$. Part (i) implies that $J^{\phi^J}(x) \le J(x)$ for all x, and (iii) tells us that x^e is globally asymptotically stable under this policy. We can then apply Prop. 3.8 to establish the bound

$$J^{\phi^J}(x) \le \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} c_J(x(k), u(k))$$

As in the proof of Prop. 3.8, the right hand side can only be increased by replacing the minimum with the optimal policy: with $x(0) = x$,

$$J^{\phi^J}(x) \le \sum_{k=0}^{\infty} c_J(x^\star(k), u^\star(k)) \le (1+\varrho) J^\star(x)$$

where the second inequality uses $c_J \le (1+\varrho)c$. ☐

3.5 Bellman Equation is a Linear Program

One approach to control design is to introduce a family of candidate value function approximations $\{J^\theta : \theta \in \mathbb{R}^d\}$, and compute the parameter θ^* that minimizes the Bellman error, such as through minimizing the mean-square criterion (2.49). A significant challenge is that the loss function $\mathcal{E}^\varepsilon(J^\theta)$ is not convex, even for when J^θ depends linearly on θ. This means we cannot appeal to Prop. 4.4 as we search for the global minimum θ^*.

We obtain a convex optimization problem by applying a common trick in optimization: over-parameterize the search space. The first step is to regard $J^\star$ and $Q^\star$ as independent variables, and regard (3.6) as a linear constraint. Following this approach we obtain a linear program that lends itself to RL algorithm design.

Naturally, we obtain a finite-dimensional linear program only if the state space and action space are finite. In this case, as part of the algorithm we choose a *weighting function* μ, and denote for any candidate approximation J,

$$\langle \mu, J \rangle := \sum_{x \in \mathsf{X}} \mu(x) J(x)$$

It is assumed that $\mu(x) > 0$ for each x, and best to make this positive everywhere to be sure the solution to the LP is unique. In most cases this will be a pmf, meaning that in addition we assume $\sum_x \mu(x) = 1$. If the state space is not finite, say $\mathsf{X} = \mathbb{R}^n$, then we must modify the definition:

$$\langle \mu, J \rangle := \sum_{x \in \mathsf{X}} \{\mu(x) J(x) : \mu(x) > 0\}$$

That is, μ is always assumed to have finite support.

Prop. 3.9 states that the Bellman equation can be cast as a linear program. If we have a parameterized family $\{J^\theta, Q^\theta : \theta \in \mathbb{R}^d\}$ that is linear in θ, then we can construct a similar linear program with variable θ. This LP is found in Section 5.5 as eq. (5.53), followed by many other approaches to approximately solve the Bellman equation.

For any function $J \colon \mathsf{X} \to \mathbb{R}$, and any scalar r, let $S_J(r)$ denote the *sublevel set*:

$$S_J(r) = \{x \in \mathsf{X} : J(x) \le r\} \tag{3.33}$$

The function J is called *inf-compact* if the set $S_J(r)$ is either compact, empty, or $S_J(r) = \mathsf{X}$ (the three possibilities depend on the value of r). In most cases we find that $S_J(r) = \mathsf{X}$ is impossible, so that we arrive at the stronger *coercive* condition (2.27):

$$\lim_{\|x\|\to\infty} J(x) = \infty$$

Proposition 3.9. *Suppose that the value function $J^\star$ defined in (3.2) is continuous, inf-compact, and vanishes only at x^e. Then, the pair $(J^\star, Q^\star)$ solve the following convex program in the "variables" (J, Q):*

$$\max_{J,Q} \ \langle \mu, J \rangle \tag{3.34a}$$

$$\text{s.t.} \quad Q(x,u) \le c(x,u) + J(\mathrm{F}(x,u)) \tag{3.34b}$$

$$Q(x,u) \ge J(x)\,, \qquad x \in \mathsf{X}\,,\ u \in \mathsf{U}(x) \tag{3.34c}$$

$$J \text{ is continuous, and } J(x^e) = 0. \tag{3.34d}$$

We can without loss of generality strengthen (3.34b) to equality: $Q(x,u) = c(x,u) + J(\mathrm{F}(x,u))$. Based on this substitution, the variable Q is eliminated:

$$\max_{J} \ \langle \mu, J \rangle \tag{3.35a}$$

$$\text{s.t.} \quad c(x,u) + J(\mathrm{F}(x,u)) \ge J(x)\,, \qquad x \in \mathsf{X}\,,\ u \in \mathsf{U}(x) \tag{3.35b}$$

This more closely resembles what you find in the stochastic control literature (see [5] for a survey). The more complex LP (3.34) is introduced because it is easily adapted to RL applications.

We present next an important corollary that will motivate RL algorithms to come. Denote for any pair (J, Q) the Bellman error:

$$\mathcal{D}(J,Q)_{(x,u)} := -Q(x,u) + c(x,u) + J(\mathrm{F}(x,u))\,, \qquad x \in \mathsf{X}\ \ u \in \mathsf{U} \tag{3.36}$$

Its square is denoted $\mathcal{E}^\varepsilon(J,Q) = \mathcal{D}(J,Q)^2$; a non-negative function on $\mathsf{X} \times \mathsf{U}$ (recall (2.49) for motivation).

Corollary 3.10. *Suppose that the assumptions of Prop. 3.9 hold. Then, for any constants $\kappa^\varepsilon > 0$, $0 \le \varrho^\varepsilon \le 1$, and pmfs μ, ν, the pair $(J^\star, Q^\star)$ solve the following quadratic program:*

$$\max_{J,Q} \ \langle \mu, J \rangle - \kappa^\varepsilon \langle \nu, \mathcal{E}^\varepsilon(J,Q) \rangle \tag{3.37a}$$

$$\text{s.t.} \quad \textit{Constraints } (3.34b)\text{–}(3.34d) \tag{3.37b}$$

$$Q(x,u) \ge (1 - \varrho^\varepsilon) c(x,u) + J(\mathrm{F}(x,u)) \tag{3.37c}$$

The extra constraint (3.37c) is introduced so we arrive at something closer to (3.35). The choice $\varrho^\varepsilon = 0$ is not excluded, but we will need the extra flexibility when we seek approximate solutions.

3.6 Linear Quadratic Regulator

For the linear system model (2.13), with quadratic cost (3.3), it is known that the value function is quadratic, $J^\star(x) = x^\intercal M^\star x$ for each x. The Q-function is also quadratic: combining the definition (3.6) with the system model (2.13a),

$$Q^\star(x,u) = c(x,u) + J^\star(Fx + Gu) \tag{3.38}$$

A more explicit quadratic representation can be found in eq. (3.41).

The optimal policy is obtained by minimizing the Q-function over u, which is easily done via the first-order condition for optimality:

$$0 = \nabla_u Q^\star(x,u) = 2Ru^\star + 2G^\intercal M^\star(Fx + Gu^\star)$$

Under the assumption that $R > 0$ it follows that $R + G^\intercal M^\star G > 0$ (and hence invertible). The minimizer $u^\star = \phi^\star(x)$ defines the optimal policy as linear state feedback:

$$\phi^\star(x) = -K^\star x \qquad \textit{with} \quad K^\star = [R + G^\intercal M^\star G]^{-1} G^\intercal M^\star F \tag{3.39}$$

To obtain $\phi^\star$ we must compute the value function, since the gain $K^\star$ depends on $M^\star \geq 0$. This matrix solves a fixed-point equation known as the *algebraic Riccati equation* (ARE):

$$M^\star = F^\intercal M^\star F - (F^\intercal M^\star G)(R + G^\intercal M^\star G)^{-1}(G^\intercal M^\star F) + S \tag{3.40}$$

This equation will in general have many symmetric solutions, though only one provides a representation of the value function. Prop. 3.9 leads to an alternative LP characterization of $M^\star$ whose solution is unique.

To understand the LP (3.34) in this special case, it is most convenient to express all three functions appearing in (3.34b) in terms of the variable $z^\intercal = (x^\intercal, u^\intercal)$:

$$J^\star(x,u) = z^\intercal M^{J^\star} z \qquad Q^\star(x,u) = z^\intercal M^{Q^\star} z \qquad c(x,u) = z^\intercal M^c z \tag{3.41a}$$

$$M^{J^\star} = \begin{bmatrix} M^\star & 0 \\ 0 & 0 \end{bmatrix} \qquad M^c = \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} \tag{3.41b}$$

$$M^{Q^\star} = M^c + \begin{bmatrix} F^T M^\star F & F^T M^\star G \\ G^T M^\star F & G^T M^\star G \end{bmatrix} \tag{3.41c}$$

Justification of the formula for $M^{Q^\star}$ is contained in the proof of Prop. 3.11 that follows.

Proposition 3.11. *Suppose that $J^\star$ is everywhere finite. Then, the value function and Q-function are each quadratic: $J^\star(x) = x^\intercal M^\star x$ for each x, where $M^\star \geq 0$ is a solution to the algebraic Riccati equation, and the quadratic Q-function is given in (3.41). The matrix $M^\star$ is also the solution to the following convex program:*

$$M^\star \in \arg\max \ \ \operatorname{trace}(M) \tag{3.42a}$$

$$\text{s.t.} \quad \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} F^T M F & F^T M G \\ G^T M F & G^T M G \end{bmatrix} \geq \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \tag{3.42b}$$

where the maximum is over symmetric matrices M, and the inequality constraint (3.42b) is in the sense of symmetric matrices.

Despite its linear programming origins, (3.42) is not a linear program: it is an example of a semi-definite program (SDP) [212]. The proof of the proposition is contained in the Appendix.

Proof of Prop. 3.11. The reader is referred to standard texts for the derivation of the ARE [1, 43]. The following is a worthwhile exercise: postulate that $J^\star$ is a quadratic function of x, and you will find that the Bellman equation implies the ARE.

Now, on to the derivation of (3.42). The variables in the linear program introduced in Prop. 3.9 consist of functions J and Q. For the LQR problem we restrict to quadratic functions:

$$J(x) = x^\intercal M x\,, \qquad Q(x,u) = z^\intercal M^Q z$$

and treat the symmetric matrices (M, M^Q) as variables.

To establish (3.42) we are left to show 1) the objective functions (3.34a) and (3.42a) coincide for some μ, and 2) the functional constraints (3.34b, 3.34c) are equivalent to the matrix inequality (3.42b). The first task is the simplest:

$$\text{trace}\,(M) = \sum_{i=1}^{n} J(e^i) = \langle \mu, J\rangle$$

with $\{e^i\}$ the standard basis elements in $\mathbb{R}^n$, and $\mu(e^i) = 1$ for each i.

The equivalence of (3.42b) and (3.34b, 3.34c) is established next, and through this we also obtain (3.41c). In view of the discussion preceding (3.35), the inequality constraint (3.34b) can be strengthened to equality:

$$Q^\star(x,u) = c(x,u) + J^\star(Fx+Gu)$$

It remains to establish the equivalence of (3.42b) and (3.35).

Applying (3.38), we obtain a mapping from M to M^Q. Denote

$$M^J = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}, \quad \Xi = \begin{bmatrix} F & G \\ F & G \end{bmatrix}$$

giving for all x and $z^\intercal = (x^\intercal, u^\intercal)$,

$$J(x) = x^\intercal M x = z^\intercal M^J z\,, \qquad J(Fx+Gu) = z^\intercal \Xi^\intercal M^J \Xi z$$

This and (3.38) gives, for any z,

$$\begin{aligned} z^\intercal M^Q z = Q(x,u) &= c(x,u) + J(Fx+Gu) \\ &= z^\intercal M^c z + z^\intercal \Xi^\intercal M^J \Xi z \end{aligned}$$

The desired mapping from M to M^Q then follows, under the standing assumption that M^Q is a symmetric matrix:

$$M^Q = M^c + \Xi^\intercal M^J \Xi = \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} F^T M F & F^T M G \\ G^T M F & G^T M G \end{bmatrix}$$

The constraint (3.35) is thus equivalent to

$$z^\intercal M^J z = J(x) \le Q(x,u) = z^\intercal M^Q z\,, \quad \textit{for all } z$$

This is equivalent to the constraint $M^J \le M^Q$, which is (3.42b). ∎

3.7 A Second Glance Ahead

In Section 2.5 it was only possible to talk about RL within the framework of policy selection within a parameterized family. We can broaden this vision now that we know something about optimal control.

Let's turn to a common approach to RL in which we choose a parameterized family of functions $\{Q^\theta : \theta \in \mathbb{R}^d\}$, and seek among them an approximation to the Q-function $Q^\star$ defined in (3.6). For any θ we obtain a policy by mimicking (3.8):

$$\phi^\theta(x) \in \arg\min_u Q^\theta(x,u)\,, \qquad x \in \mathsf{X} \tag{3.43}$$

Consider how we might approximate policy iteration: given an initial policy ϕ^0, generate a sequence of policies $\{\phi^n\}$ and parameter estimates $\{\theta_n\}$ as follows:

(i) Obtain a parameter θ_n to achieve the approximation $Q^{\theta_n} \approx Q_n$, where the latter is the fixed-policy Q-function that satisfies

$$Q_n(x,u) = c(x,u) + Q_n(x^+,u^+)\,, \qquad x^+ = \mathrm{F}(x,u)\,, \quad u^+ = \phi^n(x^+) \tag{3.44}$$

(ii) Define a new policy $\phi^{n+1} = \phi^{\theta_n}$, along with a fresh exploration policy for the next iteration of (i)

The fixed point equation (3.44) is essentially the same as the fixed policy dynamic programming equation (2.24), but defined for the fixed-policy Q-function rather than the value function. Achieving the approximation $Q^{\theta_n} \approx Q_n$ is a topic of Chapter 5, as was noted previously in Section 2.5.2. The two polices in (ii) may be required so we respect the exploration challenge discussed in Section 2.5.

Refinements of this approximation of PIA are favored in the RL literature—to be explored in greater depth in Section 5.3.

An alternative is to reconsider the temporal difference sequence (2.48), which was defined for a fixed-policy value function and associated dynamic programming equation (2.24). Recall the dynamic programming equation for the Q-function introduced in (3.9):

$$Q^\star(x,u) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u))\,, \qquad \underline{Q}^\star(x) = \min_u Q^\star(x,u)$$

Just as in (2.48), for any approximation $\widehat{Q}$ we can observe the Bellman error along the state-input trajectory:

$$\mathcal{D}_{k+1}(\widehat{Q}) := -\widehat{Q}(x(k),u(k)) + c(x(k),u(k)) + \underline{\widehat{Q}}(x(k+1)) \tag{3.45}$$

This is zero for every k if $\widehat{Q} = Q^\star$.

Q-learning is broadly defined as algorithms to choose θ^* so that $|\mathcal{D}_{k+1}(Q^\theta)|$ is in some sense minimized over all θ, based on observations of the system for $k = 0$ to N. The first approach that might come to mind is to mimic the mean-square criterion (2.49):

$$\mathcal{E}^{\varepsilon}(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\mathcal{D}_{k+1}(Q^\theta)\big]^2 \tag{3.46}$$

Unfortunately, minimizing this objective function is often difficult, because of the minimum in the definition of $\underline{Q}^\theta$. Alternative approaches are investigated in Chapter 5.

3.8 Examples

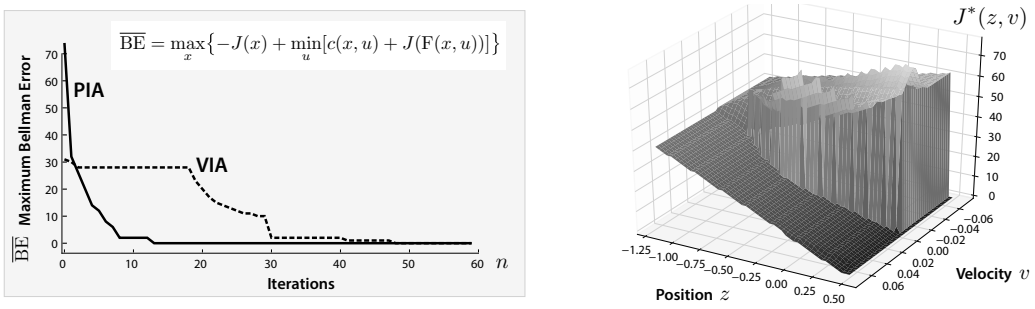


Figure 3.3: Value function for MountainCar, and performance of the two basic dynamic programming algorithms

3.8.1 Mountain Car

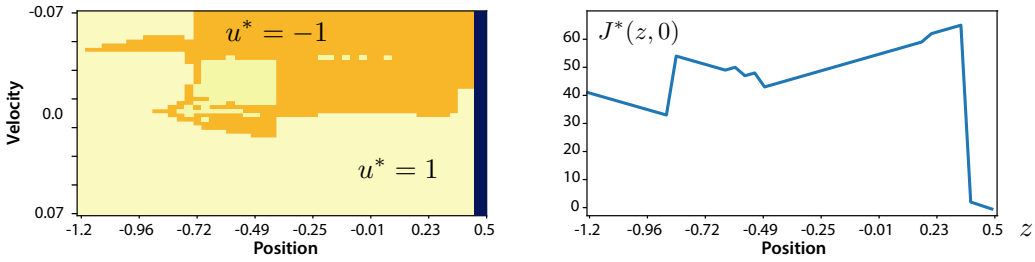


Figure 3.4: Optimal policy for MountainCar, and the total cost $J^\star(x)$ with $x = (z, 0)$.

3.9 Exercises

3.1 Consider the scalar optimal control problem with linear dynamics:

$$x(k+1) = x(k) - u(k) \tag{3.47}$$

with $x(k)$ and $u(k)$ constrained to $\mathsf{X} = \mathbb{R}_+$.

(i) With cost function $c(x,u) = x^2 + ru^2$, show that the LQR solution (**??**) solves the total cost optimization problem (3.2). *Note: you must check that your policy is feasible*, which means that $0 \le u^\star(k) \le x^\star(k)$ for each k.

(ii) Consider the more general cost $c(x,u) = x^p + ru^q$. Assume that $p, q \ge 1$ so that c is a convex function on $\mathbb{R}^2_+$. Linearity of (3.47) then implies that $J^\star$ is also convex, and you might guess that it is approximated by a similar function $J(x) = x^s$, with $s \ge 1$.

WIP Compute the Bellman error (3.30) for arbitrary s, keeping in mind that u is constrained:

$$\mathcal{B}(x) = -J(x) + \min_{0 \le u \le x} [c(x,u) + J(\mathrm{F}(x,u))]\,, \qquad \mathrm{F}(x,u) = x - u$$

3.2 *Box constrained LQR* [217]

3.3 The similarity between (3.26) and (3.10) is explored in this exercise.

To get started, assume that $J^0 \equiv 0$ for the initialization of value iteration, and compute J^1 using (3.10). Verify that $J^1(x) = J^\star(x, \mathcal{T})$ for each x. Repeat: write down the formula for J^2 using (3.10), and verify that $J^2(x) = J^\star(x, \mathcal{T}-1)$ (recall (3.26)). This procedure leads to a proof by induction that $J^{k+1}(x) = J^\star(x, \mathcal{T}-k)$ for each $k \le \mathcal{T}$.

Now a mild generalization: for general J^0, show that the function J^n obtained from the VIA algorithm (3.10) satisfies (3.11) (and observe the similarity with (3.4a)). The proof of (3.11) is an instance of the *principle of optimality* illustrated in Fig. 3.1.

3.4 Consider the double integrator $\ddot{y} = u$.

(a) Obtain a state space model with $x_t = (y(t), \dot{y}(t))^\intercal$

(b) Compute the value function as a function of $x(0) = x$:

$$J^\star(x) = \int_0^\infty y(t)^2 + u_t^2 \, dt,$$

Perform all calculations by hand.

3.5 ?

3.6 ?

3.7 ?

3.8 ?

3.9 ?

3.10 ?

3.11 ?

3.12 ?

3.10 Notes

Work in progress—comments welcome: This chapter is too broad for a full history of contributions!

There are many good books on theory and history of nonlinear optimal control [228, 27]. This chapter has provided only a brief survey of the LQR problem. See the books [125, 1] for much more theory and history, the *December 1971* special issue of the *IEEE Transactions on Automatic Control*, and early work of Kalman [109, 110].

If you have the Control System Toolbox from Matlab, then you have available some brilliant tools for computation:

`lqr`, `lqrd`: Solves the LQR problem using the data A, B, Q, and R.

`are`, `ared`: Solves the algebraic Riccati equation in continuous or discrete time

`conv`, `rlocus`: used to graph the "symmetric root locus" [1, 43] (worth knowing about, but this book is not the best reference!)

Inverse dynamic programming and the control Lyapunov approach to control has a long history in both control theory literature [78, 217, 165]. It lurks behind the curtains in the RL literature: minimization of the Bellman error is closely aligned with the IDP approach to control

The linear programming approach to dynamic programming goes back to Manne in the 60's [138, 65, 5, 34]. There is an on-going research program on LP approaches to optimal control for deterministic systems [214, 100, 101, 130, 111, 37, 88, 89, 38]. A significant program on linear programming approaches to approximate dynamic programming began in [61, 62, 63] and continues today—see Chapter 10 for more history.

Chapter 4

ODE Methods for Algorithm Design

For our purposes, an *algorithm* is a finite sequence of computer-implementable instructions, designed to compute or approximate a policy, its performance, a value function, or related quantities. In algorithm design we will see it is useful to throw away the constraints of computers, and pretend that they can operate with infinite clock-speed. An ordinary differential equation (ODE) will be regarded an example of an algorithm operating on this imaginary computer.

The motivation comes from two sources. First, we want to know if our algorithm will eventually lead to a good approximation. This is easily couched in the theory of stability of ODEs, for which there is a far richer theory than stability of recursions in discrete time. Secondly, once we have constructed an ODE with desirable properties, including stability, we can then get advice from experts to provide the translation from calculus to a practical recursive algorithm. Expertise is required in the theory of numerical methods for ODEs [46], and/or the theory of stochastic approximation (which is a close cousin). My personal favorite for the latter is [36], but there are many other great resources [23, 22, 124].

4.1 Ordinary Differential Equations

Let's start with a question we should probably have posed earlier: *what is an ODE?* The question was taken for granted many times in the preceding pages. Up to now, the state space model in continuous time considered in Section 2.4.3 is the most significant example.

The "state variable" in our applications usually represents the output of an algorithm, rather than anything directly related to a control system. For this reason, throughout this chapter we use $\boldsymbol{\vartheta} = \{\vartheta_t : t \geq 0\}$ to denote the state process for the ODE, and restrict to the Euclidean setting: $\vartheta_t \in \mathbb{R}^d$ for an integer d. The state space model (2.33) in this notation is

$$\tfrac{d}{dt}\boldsymbol{\vartheta} = \mathrm{f}(\boldsymbol{\vartheta}) \tag{4.1}$$

where $\vartheta_0 = \theta_0 \in \mathbb{R}^d$ is given, and $\mathrm{f}\colon \mathbb{R}^d \to \mathbb{R}^d$ is the vector field as in (2.33). Two examples with $n = 1$:

$\mathrm{f}(\theta) = a\theta$ and $\vartheta_t = \theta_0 e^{at}$

$\mathrm{f}(\theta) = \theta^{-2}$ and $\vartheta_t = [\theta_0^{-1} - t]^{-1}$

The ODE (4.1) is *time-homogeneous* since the right hand side does not depend upon t. See Section 3.3 for hints on how to apply state augmentation to create homogeneity for a model in which f does depend on time.

In every application of the ODE approach to algorithm design, the first step is to construct the vector field f so that ϑ_t converges to some desired value $\theta^* \in \mathbb{R}^d$. In particular, if $\vartheta_0 = \theta^*$, then the solution to the ODE should stay put: $\vartheta_t = \theta^*$ for all $t \geq 0$. This requires that $\frac{d}{dt}\vartheta_t = 0$ for all t, which by (4.1) implies that θ^* is an *equilibrium*: $\mathrm{f}(\theta^*) = 0$. Advanced material on stability theory for ODEs is contained in Section 4.8. Some of this can be anticipated from the Lyapunov theory contained in Section 2.4.3.

Understanding theory surrounding existence of solutions of (4.1) is the first step towards understanding ODE principles for algorithm design in this chapter and in Part 2 of the book. Much as how the Bellman equation (3.5) is regarded as a fixed point equation, the ODE (4.1) is a fixed point equation in the variable $\boldsymbol{\vartheta} = \{\vartheta_t : t \geq 0\}$. *Perhaps we can mirror the success of the value iteration algorithm* (3.10) (an instance of successive approximation)? Writing (4.1) as $\vartheta = \vartheta - \frac{d}{dt}\vartheta + \mathrm{f}(\vartheta)$, an analog would be

$$\vartheta^{n+1}(t) = \vartheta^n(t) - \tfrac{d}{dt}\vartheta^n(t) + \mathrm{f}(\vartheta^n(t))\,, \qquad t \geq 0,\ n \geq 0$$

with $\boldsymbol{\vartheta}^0 = \{\vartheta^0(t) : t \geq 0\}$ given as initial condition. This approach is doomed to failure! One source of difficulty is the repeated differentiation in this recursion, which means we have to be very careful with our selection of $\boldsymbol{\vartheta}^0$. Also, this recursion does not respect the requirement $\vartheta_0 = \theta_0$.

The difficulty introduced by differentiation motivates an alternative interpretation of the ODE. The Fundamental Theorem of Calculus provides something sensible:

$$\vartheta_t = \vartheta^0 + \int_0^t \mathrm{f}(\vartheta_\tau)\, d\tau\,, \qquad 0 \leq t \leq T \tag{4.2}$$

The finite time horizon T is chosen for the sake of analysis. Successive approximation is defined as before: take an initial guess $\boldsymbol{\vartheta}^0 = \{\vartheta_t : 0 \leq t \leq T\}$, and define for $n \geq 0$,

$$\vartheta^{n+1}(t) = \vartheta^0 + \int_0^t \mathrm{f}(\vartheta^n_\tau)\, d\tau\,, \qquad 0 \leq t \leq T \tag{4.3}$$

This approach is as successful as value iteration:

Proposition 4.1. *Suppose that the function* f *is globally* Lipschitz continuous*: there is* $\ell > 0$ *such that for each* $x, y \in \mathbb{R}^d$*,*

$$\|\mathrm{f}(x) - \mathrm{f}(y)\| \leq \ell \|x - y\| \tag{4.4}$$

Then for each θ_0 *there exists a solution to* (4.2) *on the infinite time horizon. Moreover successive approximation is uniformly convergent:*

$$\lim_{n\to\infty} \max_{0\leq t\leq T} \|\vartheta^n(t) - \vartheta_t\| = 0$$

□

The proof of Prop. 4.1 can be found in Chapter 9. A key component of the proof is Grönwall's Inequality, which commonly appears in the theory of stochastic approximation, as well as ordinary differential equations. Note that Bellman had early influence here [21], which is why Prop. 4.2 is often called the Bellman-Grönwall Lemma.

Proposition 4.2. (Grönwall Inequality) *Let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{z}$ be real-valued functions defined on an interval $[0, \mathcal{T}]$, with $\mathcal{T} > 0$. Assume that $\boldsymbol{\beta}$ and $\boldsymbol{z}$ are continuous.*

(i) *If $\boldsymbol{\beta}$ is non-negative and if $\boldsymbol{z}$ satisfies the integral inequality*

$$z_t \le \alpha_t + \int_0^t \beta_s z_s \, ds \tag{4.5}$$

Then Grönwall Inequality holds:

$$z_t \le \alpha_t + \int_0^t \alpha_s \beta_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds, \qquad 0 \le t \le \mathcal{T}. \tag{4.6}$$

(ii) *If, in addition, the function $\boldsymbol{\alpha}$ is non-decreasing, then*

$$z_t \le \alpha_t \exp\Big(\int_0^t \beta_s \, ds\Big), \qquad 0 \le t \le \mathcal{T}. \tag{4.7}$$

□

The proof can be found in Section 4.8, or if you have background in linear state space models, you might want to work it out on your own. Hint: first solve the problem with equality:

$$z_t = \gamma_t + \int_0^t \beta_s z_s \, ds \tag{4.8}$$

You can construct a state space model, with state $x_t = z_t - \gamma_t$, and because it is a scalar linear system you obtain an explicit solution. The solution leads to something like (4.6), but with equality.

4.2 A Brief Return to Reality

This entire chapter considers ODE approaches to algorithm design: this means design of the function f appearing in (4.1), or design of the more exotic 'quasi-stochastic' ODEs for which theory and applications are developed in Section 4.9.3 and sections 4.6 and 4.7.

There is the inevitable translation step: any design formulated in continuous time must be translated to create a practical algorithm. If you have taken a first year calculus course, then you probably have predicted the most common appoach: select a sequence of times $\{0 = t_0 < t_1 < \cdots\}$, and replace the derivative in (4.1) by a finite difference: with $\vartheta_0 = \theta_0$ given, define for each $k \ge 0$,

$$\alpha_k^{-1}[\bar{\vartheta}_{t_{k+1}} - \bar{\vartheta}_{t_k}] = \mathrm{f}(\bar{\vartheta}_{t_k})$$

where $\alpha_k = t_{k+1} - t_k > 0$. The recursive nature is evident after rearranging terms:

$$\bar{\vartheta}_{t_{k+1}} = \bar{\vartheta}_{t_k} + \alpha_k \mathrm{f}(\bar{\vartheta}_{t_k}) \tag{4.9}$$

In our final algorithm we simplify notation, writing $\theta_k = \bar{\vartheta}_{t_k}$. This is known as the Euler approximation of an ODE, or simply *Euler's method*.

This approximation is successful under the assumptions of Prop. 4.1: It can be shown that

$$\max_{0 \le t \le T} \|\bar{\vartheta}_t - \vartheta_t\| \le K(\ell, T)\bar{\alpha} \tag{4.10}$$

where $\bar{\alpha} = \max\{\alpha_k : t_{k+1} \le T\}$. The proof is given in Chapter 9, with upper bounds on $K(\ell, T)$ that at first appear frightening (growing exponentially fast in ℓ and T).

Fortunately, asymptotic stability of the ODE often implies stability of (4.10), and in this case we obtain the bound (4.10) with $K(\ell, T)$ independent of $T > 0$.

Example: The Euler approximation for the LTI model in continuous time (2.41), with $\mathrm{f}(x) = Ax$, results in the discrete time model (2.39), with $F = (1 + \alpha A)$ (with sampling interval $\alpha > 0$ fixed).

Consider the scalar case $\frac{d}{dt}\vartheta = a\vartheta$, which admits the solution $\vartheta_t = \theta_0 e^{at}$. The Euler approximation results in a similar solution, as a function of the initial condition:

$$\bar{\vartheta}_{t_{k+1}} = F^k \bar{\vartheta}_0\,, \qquad F = (1 + \alpha a)$$

The approximation $\bar{\vartheta}_{t_k} = \vartheta_{t_k} + O(\alpha)$ follows from the Taylor series approximation for the exponential, $(1 + \alpha a)^k \approx (e^{\alpha a})^k = e^{at_k}$.

If $a < 0$ and $\alpha < |a|^{-1}$, then the approximation holds on the infinite time interval, since both $\bar{\vartheta}_{t_k}$ and ϑ_{t_k} converge to zero geometrically fast, as $k \to \infty$. ☐

Those interested in a high-fidelity approximation of an ODE usually abandon the Euler approximation for more sophisticated techniques, such as the midpoint method or more general Runge-Kutta methods [46, 105]. The update equations are more complex, but this complexity is often offset by the tighter approximation.

4.3 Newton-Raphson Flow

This section concerns an approach to ODE design, in which the goal is to solve the root finding problem:

$$\mathrm{f}(\theta^*) = 0$$

The parameter θ^* may be regarded as an equilibrium condition for the ODE (4.1). The problem we face here is that this ODE may not be stable in any sense. In this section we describe a generic approach to modify the dynamics to ensure stability.

Section 4.4 concerns root finding problems for the special case in which $\mathrm{f} = -\nabla_\theta L$, where $L\colon \mathbb{R}^d \to \mathbb{R}_+$ is a loss function associated with some optimization problem. The root finding problem is then equivalent to the first order condition for optimality, $\nabla_\theta L\,(\theta^*) = 0$. If L has nice properties (such as convexity), then it is not difficult to establish stability of (4.1). In the absence of these nice properties, the techniques in this section may prove useful.

The approach taken here is to modify our objective, treating $\mathrm{f}(\vartheta_t)$ as a "state variable". Our goal is to construct an ODE so that $\lim_{t\to\infty} \mathrm{f}(\vartheta_t) = 0$ for each initial condition. Under mild additional assumptions on f it will follow that $\lim_{t\to\infty} \vartheta_t = \theta^*$, which is our design objective.

If $\mathrm{f}(\vartheta_t)$ is a state variable, this means there is an associated vector field $\mathcal{V}\colon \mathbb{R}^d \to \mathbb{R}^d$, and

$$\tfrac{d}{dt}\mathrm{f}(\vartheta_t) = \mathcal{V}(\mathrm{f}(\vartheta_t)) \tag{4.11}$$

One way to ensure that $\mathrm{f}(\vartheta_t)$ converges to zero is to choose $\mathcal{V}(f) = -f$, giving

$$\tfrac{d}{dt}\mathrm{f}(\vartheta_t) = -\mathrm{f}(\vartheta_t) \tag{4.12}$$

The solution is $\mathrm{f}(\vartheta_t) = e^{-t}\mathrm{f}(\vartheta_0)$, which converges to zero exponentially quickly. *Achieving these dynamics would be an amazing feat!*

Well, it isn't really so difficult: we have the chain rule:

$$\tfrac{d}{dt}\mathrm{f}(\vartheta_t) = A(\vartheta_t)\tfrac{d}{dt}\vartheta_t\,, \qquad \textit{with} \quad A(\theta) = \partial_\theta \mathrm{f}\,(\theta)\,, \quad \theta \in \mathbb{R}^d$$

This means that achieving the dynamics (4.11) is equivalent to

$$\tfrac{d}{dt}\vartheta_t = [A(\mathrm{f}(\vartheta_t))]^{-1}\mathcal{V}(\mathrm{f}(\vartheta_t))$$

Application of this identity, for the special case $\mathcal{V}(f) = -f$, defines the *Newton-Raphson flow*:

$$\tfrac{d}{dt}\vartheta_t = -\big[A(\vartheta_t)\big]^{-1}\mathrm{f}(\vartheta_t) \tag{4.13}$$

The function on the right hand side will be called the *Newton-Raphson vector field*

$$\mathrm{f}^{\mathrm{NRf}}(\theta) = -\big[A(\theta)\big]^{-1}\mathrm{f}(\theta) \tag{4.14}$$

In most applications it is not possible to determine a-priori if the matrix $A(\theta) = \partial_\theta \mathrm{f}\,(\theta)$ is full rank, which motivates a *regularized Newton-Raphson flow*:

$$\tfrac{d}{dt}\vartheta_t = -[\varepsilon I + A(\vartheta_t)^{\intercal}A(\vartheta_t)]^{-1}A(\vartheta_t)^{\intercal}\mathrm{f}(\vartheta_t) \tag{4.15}$$

It is shown in Prop. 4.3 that (4.15) is stable, provided $V = \|\mathrm{f}\|^2$ is a coercive function on $\mathbb{R}^d$. It follows that V serves as a Lyapunov function for (4.15), giving

$$\lim_{t\to\infty} \mathrm{f}(\vartheta_t) = 0 \tag{4.16}$$

Proposition 4.3. *Consider the following conditions for the function* f*:*

(a) f *is globally Lipschitz continuous and continuously differentiable. Hence* $A(\cdot)$ *is a bounded and continuous matrix-valued function.*

(b) $\|\mathrm{f}\|$ *is coercive. That is,* $\{\theta : \|\mathrm{f}(\theta)\| \le n\}$ *is compact for each* n*.*

(c) *The function* f *has a unique rot* θ^**, and* $A^{\intercal}(\theta)\mathrm{f}(\theta) \ne 0$ *for* $\theta \ne \theta^*$*. Moreover, the matrix* $A_* = A(\theta^*)$ *is non-singular.*

The following hold for solutions to the ODE (4.15) *under increasingly stronger assumptions:*

(i) *If (a) holds then for each* t*, and each initial condition*

$$\tfrac{d}{dt}\mathrm{f}(\vartheta_t) = -A(\vartheta_t)[\varepsilon I + A(\vartheta_t)^{\intercal}A(\vartheta_t)]^{-1}A(\vartheta_t)^{\intercal}\mathrm{f}(\vartheta_t) \tag{4.17}$$

(ii) *If in addition (b) holds, then the solutions to the ODE are bounded, and*

$$\lim_{t\to\infty} A(\vartheta_t)^{\intercal}\mathrm{f}(\vartheta_t) = 0 \tag{4.18}$$

(iii) *If (a)–(c) hold, then* (4.15) *is globally asymptotically stable.* ∎

Proof. The result (i) follows from the chain rule and the definitions.

The proof of (ii) is based on the Lyapunov function $V(\vartheta) = \frac{1}{2}\|\mathrm{f}(\vartheta)\|^2$ combined with (a):

$$\tfrac{d}{dt}V(\vartheta_t) = -\mathrm{f}(\vartheta_t)^\intercal A(\vartheta_t)[\varepsilon I + A(\vartheta_t)^\intercal A(\vartheta_t)]^{-1}A(\vartheta_t)^\intercal \mathrm{f}(\vartheta_t)$$

The right hand side is non-positive when $\vartheta_t \neq \theta^*$. Integrating each side gives for any $T > 0$,

$$V(\vartheta_T) = V(\vartheta_0) - \int_0^T \mathrm{f}(\vartheta_t)^\intercal A(\vartheta_t)[\varepsilon I + A(\vartheta_t)^\intercal A(\vartheta_t)]^{-1}A(\vartheta_t)^\intercal \mathrm{f}(\vartheta_t)\, dt \tag{4.19}$$

so that $V(\vartheta_T) \le V(\vartheta_0)$ for all T. Under the coercive assumption, it follows that solutions to (4.15) are bounded. Also, letting $T \to \infty$, we obtain from (4.19) the bound

$$\int_0^\infty \mathrm{f}(\vartheta_t)^\intercal A(\vartheta_t)[\varepsilon I + A(\vartheta_t)^\intercal A(\vartheta_t)]^{-1}A(\vartheta_t)^\intercal \mathrm{f}(\vartheta_t)\, dt \le V(\vartheta_0)$$

This combined with boundedness of ϑ_t implies that $\lim_{t\to\infty} A(\vartheta_t)^\intercal \mathrm{f}(\vartheta_t) = 0$.

We next prove (iii). Global asymptotic stability of (4.15) requires that solutions converge to θ^* from each initial condition, and also that θ^* is stable in the sense of Lyapunov. Assumption (c) combined with (ii) gives the former, that $\lim_{t\to\infty} \vartheta_t = \theta^*$. A convenient sufficient condition for the latter is obtained by considering $A_\infty = \partial_\theta[\mathcal{G}(\theta)\mathrm{f}(\theta)]\,|_{\theta=\theta^*}$. Stability in the sense of Lypaunov holds if this matrix is Hurwitz (all eigenvalues are in the strict left half plane in $\mathbb{C}$) [113, Thm. 4.7]. Apply the definitions, we obtain $A_\infty = -[\varepsilon I + M]^{-1}M$ with $M = A(\theta^*)^\intercal A(\theta^*) > 0$ (recall that $A(\theta^*)$ is assumed to be non-singular). The matrix A_∞ is negative definite, and hence Hurwitz. ◻

4.4 Optimization

Here we turn to a optimization of a loss function $L\colon \mathbb{R}^d \to \mathbb{R}_+$, for which we would like to compute a global minimum

$$\theta^* \in \arg\min L(\theta)$$

The section contains a very brief survey of optimization theory, and ODE techniques to estimate θ^*. In particular, we establish conditions under which the steepest descent ODE is convergent:

$$\tfrac{d}{dt}\vartheta = -\nabla_\theta L(\vartheta) \tag{4.20}$$

The purpose of our analysis is to find minimal conditions under which we know we will be successful. This requires strong assumptions for steepest descent other ODE methods to be considered.

However, please do not feel you have to prove a theorem before you can experiment: the algorithms we obtain are frequently successful in practice, even when our assumptions are violated. For example, the optimization problems arising in training neural networks are not convex, but practitioners commonly apply the gradient descent algorithms described next.

4.4.1 Role of convexity

A function $L\colon \mathbb{R}^d \to \mathbb{R}$ is *convex* if the following bound holds for any $\theta^0, \theta^1 \in \mathbb{R}^d$ and $\alpha \in (0,1)$:

$$L((1-\alpha)\theta^0 + \alpha\theta^1) \le (1-\alpha)L(\theta^0) + \alpha L(\theta^1) \tag{4.21}$$

An equivalent definition has a stronger geometric flavor: for each $\theta^0 \in \mathbb{R}^d$, there is a vector $v^0 \in \mathbb{R}^d$ satisfying

$$L(\theta) \ge L(\theta^0) + \langle v^0, \theta - \theta^0 \rangle\,, \qquad \textit{for all } \theta \in \mathbb{R}^d \tag{4.22}$$

The right hand side is regarded as an affine function of θ, and the bound means that the graph of L is always above the graph of the affine function. The vector v^0 is called a *sub-gradient*. If L is differentiable at θ^0, then it is an ordinary gradient $v^0 = \nabla L\,(\theta^0)$.

Recall that a set $S \subset \mathbb{R}^d$ is convex if it contains all line segments with endpoints in S. That is, if $\theta^0, \theta^1 \in S$, then $(1-\alpha)\theta^0 + \alpha\theta^1 \in S$ for any $\alpha \in (0,1)$. The function L is called *quasi-convex* if the sublevel set $S_L(r)$ is convex (or empty) for any scalar r, where (recalling (2.26)),

$$S_L(r) = \{\theta \in \mathbb{R}^d : L(\theta) \le r\}$$

For example, with $d = 1$, any continuous non-decreasing function is quasi-convex, since in this case $S_L(r) = (-\infty, a(r)]$ for each r, where $a(r) = \max\{\theta : L(\theta) \le r\}$ (continuity isn't required to ensure quasi-convexity, but is required to arrive at this representation for the sublevel set.

A convex function is always quasi-convex. There are also several stronger conditions for the function L:

- ▲ It is *strictly convex* if the inequality (4.21) is strict whenever $\theta^1 \ne \theta^0$.
- ▲ If L is differentiable, then it is called *strongly convex* if for a constant $m > 0$ and all $\theta, \theta^0 \in \mathbb{R}^d$,

$$\{\nabla L(\theta) - \nabla L(\theta^0)\}^{\intercal}(\theta - \theta^0) \ge m\|\theta - \theta^0\|^2 \tag{4.23}$$

Strong convexity is used to establish nice numerical properties of ODEs designed to compute a global optimum. The value of convexity and strict convexity is made clear in the following.

Proposition 4.4. *Suppose that $L\colon \mathbb{R}^d \to \mathbb{R}_+$ is convex. Then, for given $\theta^0 \in \mathbb{R}^d$,*

(i) *If θ^0 is a local minimum, then it is also a global minimum*

(ii) *If L is differentiable at θ^0, with $\nabla L\,(\theta^0) = 0$, then θ^0 is a global minimum.*

(iii) *If either of (i) or (ii) holds, and if L is strictly convex, then θ^0 is the unique global minimum* □

Proposition 4.5. *Suppose that L is strongly convex and continuously differentiable, with unique minimizer θ^*. Then θ^* is globally asymptotically stable for the gradient descent ODE (4.20). If L is strongly convex, then the rate of convergence is exponential:*

$$\tfrac{d}{dt}\|\vartheta_t - \theta^*\| \le e^{-mt}\|\vartheta_0 - \theta^*\|$$

where m appears in (4.23).

Proof. We adopt the Lyapunov function approach, using $V(\theta) = \frac{1}{2}\|\theta - \theta^*\|^2$. From the chain rule,

$$\tfrac{d}{dt}V(\vartheta) = -\nabla_\theta L(\vartheta)^\intercal\{\vartheta - \theta^*\}$$

... □

4.4.2 Constrained optimization

Consider the optimization problem with equality constraints:

$$\begin{aligned} L^\star := \min \quad & L(\theta) \\ \text{s.t.} \quad & g(\theta) = 0 \end{aligned} \tag{4.24}$$

where $g\colon \mathbb{R}^d \to \mathbb{R}^m$ (so there are $m \geq 1$ constraints). The constraints are convex if and only if g is an affine function of θ: for an $m \times d$ matrix D and vector $d \in \mathbb{R}^m$,

$$g(\theta) = D\theta + d$$

One approach to the solution of these problems is through a *Lagrangian relaxation*, defined through a sequence of steps. Introduce the Lagrangian $\mathcal{L}\colon \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$:

$$\mathcal{L}(\theta, \lambda) = L(\theta) + \lambda^\intercal g(\theta)\,, \qquad \theta \in \mathbb{R}^d\,, \quad \lambda \in \mathbb{R}^m$$

The so-called dual function is the minimum of the Lagrangian, with constraints removed:

$$\varphi^*(\lambda) = \min_\theta \mathcal{L}(\theta, \lambda)$$

The value $-\infty$ is possible: consider what happens when L is a linear function of θ.

For any $\lambda \in \mathbb{R}^m$ we obtain a lower bound on $L^\star$ as follows:

$$\varphi^*(\lambda) \leq \min_\theta\{\mathcal{L}(\theta, \lambda) : g(\theta) = 0\} = \min_\theta\{L(\theta) : g(\theta) = 0\} = L^\star$$

where the inequality holds because we have re-introduced the constraints, which means we are minimizing $\mathcal{L}$ over a potentially smaller set. The dual problem is defined to be the maximum of φ^* over all λ:

$$\max_\lambda \min_\theta \mathcal{L}(\theta, \lambda) = \max_\lambda \varphi^*(\lambda) \leq L^\star \tag{4.25}$$

We say there is a *duality gap* if the inequality is strict. The left hand side is called a min-max (or saddle point) problem.

There is a simple ODE to obtain a solution to the saddle point problem (4.25), known as the *primal-dual flow*:

$$\tfrac{d}{dt}\vartheta = -\nabla_\theta \mathcal{L}(\vartheta, \lambda) = -\nabla L(\vartheta) - [\nabla_\theta g(\vartheta)]^\intercal \lambda \tag{4.26a}$$

$$\tfrac{d}{dt}\lambda = \nabla_\lambda \mathcal{L}(\vartheta, \lambda) = g(\vartheta) \tag{4.26b}$$

where for each i,

$$\big\{[\nabla_\theta g(\vartheta)]^\intercal \lambda\big\}_i = \langle \nabla_\theta g_i(\vartheta), \lambda\rangle = \nabla_\theta g_i(\vartheta)^\intercal \lambda$$

Proposition 4.6. *Suppose that L is strictly convex, and g is affine. Then the primal-dual algorithm converges to the unique solution $(\theta^*, \lambda^\star)$ of the dual:*

$$\mathcal{L}(\theta^*, \lambda^\star) = L^\star$$

Proof. The proof is remarkably simple, and similar to Prop. 4.5. Consider the Lyapunov function

$$V(\theta, \lambda) = \tfrac{1}{2}\|\theta - \theta^*\|^2 + \tfrac{1}{2}\|\lambda - \lambda^\star\|^2$$

Applying the chain rule,

$$\begin{aligned}\tfrac{d}{dt}V(\vartheta, \lambda) &= \langle \vartheta_t - \theta^*, \partial_\theta \mathcal{L}(\vartheta_t \lambda_t) \tfrac{d}{dt}\vartheta_t\rangle + \langle \lambda_t - \lambda^\star, \nabla_\lambda \mathcal{L}(\vartheta_t, \lambda_t), \partial_\lambda \mathcal{L}(\vartheta_t \lambda_t) \tfrac{d}{dt}\lambda_t\rangle \\ &\le [\mathcal{L}(\theta^* , \lambda_t) - \mathcal{L}(\theta^* , \lambda^\star)]dt + [\mathcal{L}(\theta^* , \lambda^\star) - \mathcal{L}(\vartheta_t , \lambda^\star)]dt\end{aligned}$$

... □

The case of *inequality constraints* is considered next:

$$\begin{aligned} L^\star := \min \quad & L(\theta) \\ \text{s.t.} \quad & g(\theta) \le \mathsf{0} \end{aligned} \tag{4.27}$$

where again $g\colon \mathbb{R}^d \to \mathbb{R}^m$. If g_i is a convex function for each i (or simply quasi-convex), then the constraint region $S = \{\theta : g(\theta) \le \mathsf{0}\}$ is a convex set.

The Lagrangian and dual function φ^* are defined exactly as before, but we must restrict to $\lambda \in \mathbb{R}^m_+$ to obtain the prior upper bound:

$$\varphi^*(\lambda) \le \min_\theta \{\mathcal{L}(\theta, \lambda) : g(\theta) \le \mathsf{0}\} \le \min_\theta \{L(\theta) : g(\theta) \le \mathsf{0}\} = L^\star$$

where the second inequality is based on the bound $\lambda^\intercal g(\theta) \le 0$ when $g(\theta) \le \mathsf{0}$ and $\lambda \ge \mathsf{0}$. The saddle point problem is defined as before:

$$\max_{\lambda \ge 0} \min_\theta \mathcal{L}(\theta, \lambda) = \max_{\lambda \ge 0} \varphi^*(\lambda) \le L^\star \tag{4.28}$$

Subject to convexity and minor additional assumptions, there is no duality gap (the inequality is actually and equality).

The *primal-dual flow* is almost the same as (4.26). The ODE for the parameter estimate is identical:

$$\tfrac{d}{dt}\vartheta = -\nabla L(\vartheta) - [\nabla_\theta g(\vartheta)]^\intercal \lambda$$

The ODE for the dual variable $\boldsymbol{\lambda}$ must be modified to impose non-negativity. This comes in the form of an m-dimensional *reflection process* $\boldsymbol{\gamma}$. It is easiest to express the new dual dynamics in integral form

$$\lambda_t = \lambda_0 + \int_0^t g(\vartheta_r)\, dr + \gamma_t\,, \tag{4.29}$$

where $\lambda_0 \ge \mathsf{0}$ is the initial condition. The reflection process is defined by three constraints (for each $1 \le i \le m$):

1. $\gamma_0(i) = 0$

2. $\boldsymbol{\gamma}(i)$ is non-decreasing, and the solution to (4.29) is non-negative ($\lambda_t(i)$ is non-negative for each i and t).
3. It is the *minimal* function of time satisfying 1 and 2, which is equivalently expressed

$$\int_0^T \lambda_t(i)\, d\gamma_t(i) = 0\,, \qquad \textit{for all } T > 0 \tag{4.30}$$

The integral (4.30) is defined in the sense of Riemann and Stieltjes, and on combining 1–3 we see that

$$\lambda_t(i) > 0 \Longrightarrow \tfrac{d}{dt}\gamma_t(i) = 0$$

Prop. 4.6 extends easily to this primal-dual flow, and the proof is identical: the property (4.30) makes the reflection process disappear in the Lyapunov function analysis.

The open question is how to translate this into a discrete-time algorithm, since (4.29) is no longer an ODE. A standard primal-dual algorithm is defined by the pair of recursions:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big\{-\nabla L(\theta_n) + \lambda_n^{\intercal}\nabla g(\theta_n)\big\} \tag{4.31a}$$

$$\lambda_{n+1} = \big[\lambda_n - \alpha_{n+1} g(\theta_n)\big]_+ \tag{4.31b}$$

where $\big[\,\cdot\,\big]$ is the component-wise maximum with zero. Hence (4.31b) can be expressed

$$\begin{aligned}\lambda_{n+1} &= \lambda_n - \alpha_{n+1} g(\theta_n) + \delta\gamma_n \\ \delta\gamma_n &= \big[\lambda_n - \alpha_{n+1} g(\theta_n)\big]_+ - [\lambda_n - \alpha_{n+1} g(\theta_n)] \ge 0\end{aligned}$$

with $\delta\gamma_n$ interpreted as an increment of the reflection process. It is approximately minimal, since $\delta\gamma_n \times \lambda_n = 0$ whenever θ_n is feasible (i.e., $g(\theta_n) \le 0$).

4.5 Quasi-Stochastic Approximation

This section is an early introduction to Chapter 9, which concerns ODE approximations and algorithm design in a stochastic setting, based on the theory of *stochastic approximation*.

We are interested in solving a root-finding problem of a special form, which requires an adjustment of notation. We are given a function $f : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$, along with an m-dimensional "random vector" Φ. The function f used in the previous sections is replaced by the average (or expectation):

$$\overline{f}(\theta) := \mathsf{E}[f(\theta, \Phi)]\,, \qquad \theta \in \mathbb{R}^d\,, \tag{4.32}$$

in which Our goal is then to solve $\overline{f}(\theta^*) = 0$ for this strange function $\overline{f}\colon \mathbb{R}^d \to \mathbb{R}^d$. In this section we introduce algorithms to achieve this goal by adapting the ODE approaches in previous sections. The big difference is that we know nothing about f or Φ.

If you don't know what is meant by random, or expectations, you don't have to worry. We avoid any mention of probability in this section by replacing "random variables" by sinusoids, or other "bouncy" functions of time.

For those of you with a background in probability: the stochastic approximation (SA) method of Robbins and Monro [171] amounts to a variation of the Euler scheme (4.9), in which we replace $\overline{f}$ by samples from f:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \Phi_n)\,, \qquad n \ge 0, \tag{4.33}$$

where each Φ_n has the same distribution as Φ, or the distribution of Φ_n converges to that of Φ as $n \to \infty$. It will be seen in Chapter 9 that this recursion does approximate the associated ODE:

$$\frac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t). \tag{4.34}$$

A useful approximation requires assumptions on f, the "noise" Φ_n, and the step-size sequence. The required assumptions, and the mode of analysis, is not very different than what is required to successfully apply the deterministic Euler approximation (4.9).

The upshot of stochastic approximation is that it can be implemented without knowledge of the function f or of the distribution of Φ; rather, it can rely on observations of the sequence $\{f(\theta_n, \Phi_n)\}$. This is one reason why these algorithms are valuable in the context of reinforcement learning.

In much of the SA literature it is assumed that $\boldsymbol{\Phi}$ is a *Markov chain*: a topic considered in depth in Chapter 6. A key observation in the present chapter is that Markov chains need not be stochastic: the deterministic state space model (2.20) (without control) always satisfies the *Markov property* used in Part 2 of this book. For example, for given $\omega > 0$, the sequence $\Phi_n = [\cos(\omega n), \sin(\omega n)]$ is a Markov chain on the unit circle in $\mathbb{R}^2$. This motivates the *quasi-stochastic approximation* (QSA) algorithm:

$$\tfrac{d}{dt}\Theta_t = a_t f(\Theta_t, \xi_t)\,, \tag{4.35}$$

We use the terms *gain* and *stepsize* inter-changeably for the non-negative process $\boldsymbol{a}$.

The *probing signal* $\boldsymbol{\xi}$ is generated from a *deterministic* (possibly oscillatory) signal rather than a stochastic process. Two canonical choices are the m-dimensional mixtures of periodic functions:

$$\xi_t = \sum_{i=1}^{K} v^i [\phi_i + \omega_i t]_{(\text{mod } 1)} \tag{4.36a}$$

$$\xi_t = \sum_{i=1}^{K} v^i \sin(2\pi[\phi_i + \omega_i t]) \tag{4.36b}$$

for fixed vectors $\{v^i\} \subset \mathbb{R}^m$, phases $\{\phi_i\}$, and frequencies $\{\omega_i\}$. Under mild conditions on f we can be assured of the existence of the existence of this limit defining the mean vector field:

$$\overline{f}(\theta) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f(\theta, \xi_t)\, dt, \quad \text{for all } \theta \in \mathbb{R}^d. \tag{4.37}$$

Such signals have well defined steady-state means and covariance matrices. Consider for example (4.36b) in the special case

$$\xi_i(t) = \sqrt{2}\sin(\omega_i t)\,, \qquad 1 \le i \le m \tag{4.38}$$

with $\omega_i \neq \omega_j$ for all $i \neq j$. The steady state mean and covariance are then

$$\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \xi_t\, dt = 0 \tag{4.39a}$$

$$\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \xi_t \xi_t^\intercal\, dt = I \tag{4.39b}$$

where I is the identity matrix.

The following subsections contain examples to illustrate theory of QSA, and also a glimpse at applications.

4.5.1 Quasi Monte-Carlo

Consider the problem of obtaining the integral over the interval $[0,1]$ of a function $y\colon \mathbb{R} \to \mathbb{R}$. In a standard Monte-Carlo approach we would draw independent random variables $\{\Phi_n\}$, with distribution uniform on the interval $[0,1]$, and then average:

$$\theta_n = \frac{1}{n}\sum_{k=0}^{n-1} y(\Phi_k) \tag{4.40}$$

A QSA analog is described as follows: the probing signal is the one-dimensional *sawtooth function*, $\xi_t := t$ (modulo 1) and consider the analogous average

$$\Theta_t = \frac{1}{t}\int_0^t y(\xi_r)\, dr \tag{4.41}$$

Alternatively, we can adapt the QSA model (4.35) to this example, with

$$f(\theta, \xi) := y(\xi) - \theta. \tag{4.42}$$

The averaged function is then given by

$$\overline{f}(\theta) = \lim_{T\to\infty} \frac{1}{T}\int_0^T f(\theta, \xi_t)\, dt = \int_0^1 y(\xi_t)\, dt - \theta$$

so that $\theta^* = \int_0^1 y(\xi_t)\, dt$ is the unique root of $\overline{f}$. Algorithm (4.35) is given by:

$$\tfrac{d}{dt}\Theta_t = a_t[y(\xi_t) - \Theta_t]. \tag{4.43}$$

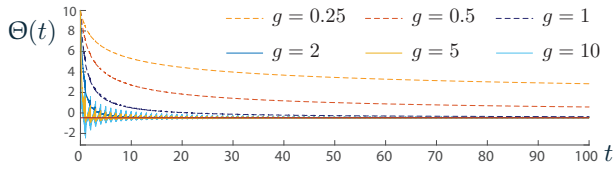


Figure 4.1: Sample paths of Quasi Monte-Carlo estimates.

This Monte-Carlo approach (4.41) can be transformed into something resembling (4.43). Taking derivatives of each side of (4.41), we obtain using the product rule of differentiation, and the fundamental theorem of calculus,

$$\tfrac{d}{dt}\Theta_t = -\frac{1}{t^2}\int_0^t y(\xi_r)\, dr + \frac{1}{t}y(\xi_t) = \frac{1}{t}[y(\xi_t) - \Theta_t]$$

This is precisely (4.43) with $a_t = 1/t$ (not a great choice for an ODE design, since it is not bounded as $t \downarrow 0$).

The numerical results that follow are based on $y(\theta) = e^{4t}\sin(100\theta)$, whose mean is $\theta^* \approx -0.5$. The differential equation (4.43) was approximated using a standard Euler scheme with sampling interval 10^{-3}. Several variations were simulated, differentiated by the gain $a_t = g/(1+t)$. Fig. 4.1 shows typical sample paths of the resulting estimates for a range of gains, and common initialization $\Theta_0 = 10$. In each case, the estimates converge to the true mean $\theta^* \approx -0.5$, but convergence is very slow for $g > 0$ significantly less than one. Recall that the case $g = 1$ is very similar to what was obtained from the Monte-Carlo approach (4.41).

Independent trials were conducted to obtain variance estimates. In each of 10^4 independent runs, the common initial condition was drawn from $N(0, 10)$, and the estimate was collected at time $T = 100$. Fig. 4.2 shows three histograms of estimates for standard Monte-Carlo (4.40), and QSA using gains $g = 1$ and 2. An alert reader must wonder: *why is the variance reduced by 4 orders of magnitude when the gain is increased from* 1 *to* 2*?* The relative success of the high-gain algorithm is explained in Section 4.5.

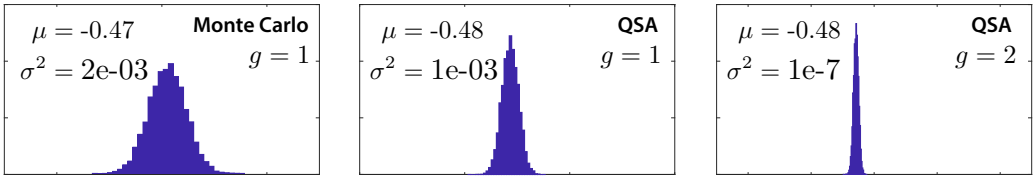


Figure 4.2: Histograms of Monte-Carlo and Quasi Monte-Carlo estimates after 10^4 independent runs. The optimal parameter is $\theta^* \approx -0.4841$.

Buyer beware The remainder of this chapter is based on extensions of quasi Monte-Carlo, which is traditionally framed in discrete time. The sawtooth function used in (4.41) is a common choice in this research area, defined more generally in discrete time as follows:

$$\xi(k) = \xi(0) + 2\pi\omega k \pmod 1 \tag{4.44}$$

Subject to conditions on the parameter ω and function $y\colon \mathbb{R} \to \mathbb{R}$, we have the Law of Large Numbers:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N} y(\xi(k)) = \int_0^1 y(r)\,dr$$

This is known as the Equidistribution Theorem (see [20], and [94, p. 87] for more history). The quasi Monte-Carlo literature contains more sophisticated techniques to define well behaved "probing sequences".

Sinusoids and sawtooth functions are used in this chapter for simplicity (and because of my own ignorance of the substantial literature on pseudo randomness). So, translating a QSA algorithm to a discrete time algorithm requires two steps: 1) an approximation of the ODE, perhaps using the standard Euler scheme, and 2) careful selection of the probing sequence in discrete time. I suspect that the latter choice requires care only for parameter estimation in high dimension.

4.5.2 Constant gain algorithm

In later sections we will consider the linear approximation:

$$f(\theta, \xi) = A(\theta - \theta^*) + B\xi \tag{4.45}$$

This provides insight, and sometimes we can show strong coupling between the linear and nonlinear QSA ODEs. We briefly consider here this linear model in which $a_t = \alpha$ is constant. Then QSA is a time-invariant linear system:

$$\tfrac{d}{dt}\Theta_t = \alpha[A\widetilde{\Theta}_t + B\xi_t]\,, \qquad \widetilde{\Theta}_0 = \tilde{\theta}_0$$

where $\widetilde{\Theta}_t := \Theta_t - \theta^*$ is the error at time t. For this simple model we can solve the ODE when the probing signal is the mixture of sinusoids (4.36b).

A linear system satisfies the *principle of super-position*. To put this to work, consider the probing signal (4.36b), and for each i, consider the ODE

$$\tfrac{d}{dt}\widetilde{\Theta}^i_t = \alpha\big[A\widetilde{\Theta}_t + v^i \sin(2\pi[\phi_i + \omega_i t])\big)\,, \qquad \widetilde{\Theta}^i_0 = 0$$

The principle states that the solution to the ODE is the sum:

$$\widetilde{\Theta}_t = e^{\alpha A t}\tilde{\theta}_0 + B\sum_{i=1}^{K}\widetilde{\Theta}^i_t \tag{4.46}$$

We see that the response to the initial error $\tilde{\theta}_0 = \theta_0 - \theta^*$ decays to zero exponentially quickly. Moreover, to understand the steady-state behavior of the algorithm it suffices to fix a single value of i. For more complex probing signals we can again justify consideration of sinusoids, provided we can justify a Fourier series approximation.

Let's keep things simple, and stick to sinusoids. And it is much easier to work with complex exponentials:

$$\tfrac{d}{dt}\widetilde{\Theta}_t = \alpha[A\widetilde{\Theta}_t + v\exp(j\omega t)]\,, \qquad \widetilde{\Theta}_0 = 0$$

with $\omega \in \mathbb{R}$ and $v \in \mathbb{R}^d$ (dropping the scaling 2π for simplicity, and the phase ϕ is easily returned by a time-shift). We can express the solution as a convolution:

$$\begin{aligned}\widetilde{\Theta}_t &= \alpha\int_0^t \exp\big(\alpha A r\big)v\exp\big(j\omega(t-r)\big)\,dr \\ &= \alpha\Big(\int_0^t \exp\big([\alpha A - j\omega I]r\big)\,dr\Big)v\exp\big(j\omega t\big)\end{aligned}$$

Writing $D = [\alpha A - j\omega I]$, the integral of the matrix exponential is expressed,

$$\int_0^t e^{Dr}\,dr = D^{-1}\big[e^{Dt} - I\big]$$

Using linearity once more, and the fact that the imaginary part of $e^{j\omega t}$ is $\sin(\omega t)$, we arrive at a complete representation for (4.46):

Proposition 4.7. *Consider the linear model with A Hurwitz, and probing signal (4.36b), for which the constant-gain QSA algorithm has the solution (4.46). Then $\widetilde{\Theta}^i_t = \alpha\Gamma^i_t v^i$ for each i and t, with*

$$\Gamma^i_t = \mathrm{Im}\Big(\big[\alpha A - j\omega I\big]^{-1}\big[\exp\big(\alpha A t\big) - \exp\big(2\pi j[\phi_i + \omega_i t]\big)I\big]\Big) \tag{4.47}$$

Prop. 4.7 illustrates a challenge with fixed gain algorithms: if we want small steady-state error, then we require small α (or large ω_i, but this brings other difficulties for computer implementation—*never forget Euler!*). However, if $\alpha > 0$ is very small, then the impact of the initial condition in (4.46) will persist for a long time.

The Ruppert-Polyak averaging technique can be used to improve the steady-state behavior—more on this can be found in Section 4.9.4 for vanishing-gain algorithms. It is easy to illustrate the value for the special case considered here. One form of the technique is to simply average some fraction of the estimates:

$$\Theta_T^{\text{RP}} := \frac{1}{T - T_0} \int_{T_0}^{T} \Theta_t \, dt \tag{4.48}$$

For example, $T_0 = T - T/5$ means that we average the final 20%.

Corollary 4.8. *Suppose that the assumptions of Prop. 4.7 hold, so in particular f is linear. Consider the averaged estimates (4.48) in which $T_0 = T - T/K$ for fixed $K > 1$. Then,*

$$\Theta_T^{RP} = \theta^* + M_T \theta_0 + B \sum_{i=1}^{K} \Theta_T^{RPi}$$

where

$$M_T = \frac{K}{T} \alpha^{-1} A^{-1} \Big[\exp\big(\alpha A T\big) - \exp\big(\alpha A T_0\big) \Big]$$

and $\Theta_T^{RPi} = \alpha \Gamma_t^{RPi} v^i$ for each i and t, with Γ_T^{RPi} equal to the integral of Γ_t^i appearing in (4.47):

$$\begin{aligned} \Gamma_T^{RPi} = \frac{K}{T} \text{Im}\Big([\alpha A]^{-1} [\alpha A - j 2\pi\omega_i I]^{-1} \Big[\exp\big(\alpha A T\big) - \exp\big(\alpha A T_0\big) \Big] \Big) \\ + \frac{K}{T} \text{Im}\Big(\frac{j}{2\pi\omega_i} [\alpha A - j 2\pi\omega_i I]^{-1} \Big[\exp\big(2\pi[\phi_i + \omega_i T] j\big) - \exp\big(2\pi[\phi_i + \omega_i T_0] j\big) \Big] \Big) \end{aligned}$$

Hence, Θ_T^{RP} converges to θ^* at rate $1/T$. ◻

4.5.3 Application to policy iteration

Consider the nonlinear state space model in continuous time,

$$\tfrac{d}{dt} x_t = \mathrm{f}(x_t, u_t) \,, \qquad t \geq 0$$

with $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$. Given a cost function $c \colon \mathbb{R}^{n+m} \to \mathbb{R}$, our goal is to approximate the optimal value function

$$J^\star(x) = \min_{\boldsymbol{u}} \int_0^\infty c(x_t, u_t) \, dt \,, \qquad x = x_0$$

and approximate the optimal policy. For this we first explain how policy iteration extends to the continuous time setting.

For any feedback law $u_t = \phi(x_t)$, denote the associated value function by

$$J^\phi(x) = \int_0^\infty c(x_t, \phi(x_t)) \, dt \,, \qquad x = x_0.$$

It follows from Prop. 2.7 that this solves a dynamic programming equation:

$$0 = c(x, \phi(x)) + \nabla J^{\phi}(x) \cdot \mathrm{f}(x, \phi(x))$$

The policy improvement step in this continuous time setting defines the new policy as the minimizer:

$$\phi^{+}(x) \in \arg\min_{u} \{c(x, u) + \nabla J^{\phi}(x) \cdot \mathrm{f}(x, u)\}$$

Consequently, approximating the term in brackets is key to approximating PIA.

An RL algorithm is constructed through the following steps. First, add J^{ϕ} to each side of the fixed-policy dynamic programming equation:

$$J^{\phi}(x) = J^{\phi}(x) + c(x, \phi(x)) + \nabla J^{\phi}(x) \cdot \mathrm{f}(x, \phi(x))$$

The right-hand side motivates the following definition of the fixed-policy Q-function:

$$Q^{\phi}(x, u) = J^{\phi}(x) + c(x, u) + \mathrm{f}(x, u) \cdot \nabla J^{\phi}(x).$$

The policy update can be equivalently expressed $\phi^{+}(x) = \arg\min_u Q^{\phi}(x, u)$, and this Q-function solves the fixed point equation

$$Q^{\phi}(x, u) = \underline{Q}^{\phi}(x) + c(x, u) + \mathrm{f}(x, u) \cdot \nabla \underline{Q}^{\phi}(x) \tag{4.49}$$

where $\underline{H}^{\phi}(x) = H(x, \phi(x))$ for any function H (note that this is a substitution, rather than the minimization appearing in (3.9)).

Consider now a family of functions Q^{θ} parameterized by θ, and define the Bellman error for a given parameter as

$$\mathcal{E}^{\theta}(x, u) = -Q^{\theta}(x, u) + \underline{Q}^{\theta}(x) + c(x, u) + \mathrm{f}(x, u) \cdot \nabla \underline{Q}^{\theta}(x) \tag{4.50}$$

A model-free representation is obtained, on recognizing that for any state-input pair (x_t, u_t),

$$\mathcal{E}^{\theta}(x_t, u_t) = -Q^{\theta}(x_t, u_t) + \underline{Q}^{\theta}(x_t) + c(x_t, u_t) + \tfrac{d}{dt} \underline{Q}^{\theta}(x_t) \tag{4.51}$$

The error $\mathcal{E}^{\theta}(x_t, u_t)$ can be observed without knowledge of the dynamics f or even the cost function c. The goal is to find θ^* that minimizes the mean square error:

$$\|\mathcal{E}^{\theta}\|^2 := \lim_{T \to \infty} \frac{1}{T} \int_0^T \left[\mathcal{E}^{\theta}(x_t, u_t)\right]^2 dt. \tag{4.52}$$

We choose a feedback law with "exploration", of the form introduced in Section 2.5.3:

$$u_t = \widetilde{\phi}(x_t, \xi_t) \tag{4.53}$$

chosen so that the resulting state trajectories are bounded for each initial condition, and that the joint process $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\xi})$ admits an ergodic steady state.

Whatever means we use to obtain the minimizer, this approximation technique defines an approximate version of PIA: given a policy ϕ and approximation Q^{θ^*}, the policy is updated:

$$\phi^{+}(x) = \arg\min_{u} Q^{\theta^*}(x, u) \tag{4.54}$$

This procedure is repeated to obtain a recursive algorithm.

Least squares solution Consider for fixed T the loss function

$$L_T(\theta) = \frac{1}{T}\int_0^T \left[\mathcal{E}^\theta(x_t, u_t)\right]^2 dt$$

If the function approximation architecture is linear,

$$Q^\theta(x,u) = d(x,u) + \theta^\intercal \psi(x,u)\,, \quad \theta \in \mathbb{R}^d. \tag{4.55}$$

in which $d\colon \mathsf{X}\times\mathsf{U}\to\mathbb{R}$. Then L_T is a quadratic function of θ:

$$L_T(\theta) = \theta^\intercal M\theta - 2b^\intercal\theta + L_T(0) = (\theta-\theta^*)^\intercal M(\theta-\theta^*) + L_T(0)$$

We leave it to the reader to find expressions for M, b, and $L_T(0)$.

In this special case we do not need gradient descent techniques: the matrices M and b can be represented as Monte-Carlo, as surveyed in Section 4.5.1, and then $\theta^* = M^{-1}b$.

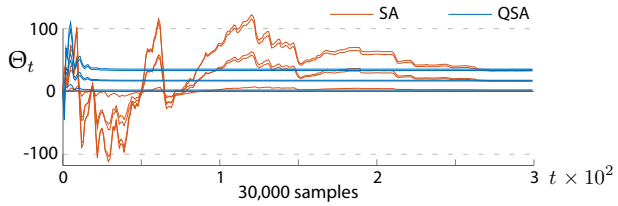


Figure 4.3: Comparison of QSA and Stochastic Approximation (SA) for policy evaluation.

Gradient descent The first-order condition for optimality is expressed as a root-finding problem: $\nabla_\Theta \|\mathcal{E}^\theta\|^2 = 0$, and the standard gradient descent algorithm in ODE form is

$$\tfrac{d}{dt}\vartheta_t = -\tfrac{1}{2}a\nabla_\theta\|\mathcal{E}^{\vartheta_t}\|^2 = -a\mathcal{E}^{\vartheta_t}\nabla_\theta\mathcal{E}^{\vartheta_t}$$

with $a>0$. This is an ODE of the form (4.34), whose QSA counterpart (4.35) is the QSA steepest descent algorithm,

$$\begin{aligned}\tfrac{d}{dt}\Theta_t &= -a_t\mathcal{E}^{\Theta_t}(x_t,u_t)\zeta_t^{\Theta_t}\\ \zeta_t^\theta &:= \nabla_\theta\mathcal{E}^\theta(x_t,u_t)\end{aligned} \tag{4.56}$$

Where, based on (4.51) we can typically swapping derivative with respect to time and derivative with respect to θ to obtain

$$\nabla_\theta\mathcal{E}^\theta(x_t,u_t) = -\nabla_\theta Q^\theta(x_t,u_t) + \left\{\nabla_\theta Q^\theta(x_t,\phi(x_t)) + \tfrac{d}{dt}\nabla_\theta Q^\theta(x_t,\phi(x_t))\right\}$$

The QSA gradient descent algorithm (4.56) is best motivated by a nonlinear function approximation, but it is instructive to see how the ODE simplifies for the the linearly parameterized family (4.55). We have in this case

$$\zeta_t = -\psi(x_t,u_t) + \psi(x_t,\phi(x_t)) + \tfrac{d}{dt}\psi(x_t,\phi(x_t))$$

and $\mathcal{E}^\theta(x_t,u_t) = b_t + \zeta_t^\intercal\theta$ using

$$b_t = c(x_t,u_t) - d(x_t,u_t) + d(x_t,\phi(x_t)) + \tfrac{d}{dt}d(x_t,\phi(x_t))$$

so that (4.56) becomes

$$\tfrac{d}{dt}\Theta_t = -a_t \left[\zeta_t \zeta_t^\intercal \, \Theta_t + b_t \zeta_t\right] \tag{4.57}$$

The convergence of (4.57) may be very slow if the matrix

$$G := \lim_{t\to\infty} \frac{1}{t} \int_0^t \zeta_\tau \zeta_\tau^\intercal \, d\tau \tag{4.58}$$

has eigenvalues close to zero. This can be resolved through the introduction of a larger gain $\boldsymbol{a}$, or a matrix gain. One approach is to estimate G from data and invert:

$$\widehat{G}_t = \frac{1}{t} \int_0^t \zeta_\tau \zeta_\tau^\intercal \, d\tau, \qquad 0 \le t \le T \tag{4.59a}$$

$$\tfrac{d}{dt}\Theta_t = -a_t \widehat{G}_T^{-1} \left[\zeta_t \zeta_t^\intercal \, \Theta_t + b_t \zeta_t\right], \qquad t \ge T \tag{4.59b}$$

This might be motivated by the ODE approximation

$$\tfrac{d}{dt}\vartheta = -a\{\vartheta - \theta^*\}$$

Justification for this approximation is contained in Section 4.9.

Numerical example Consider the LQR problem in which $\mathrm{f}(x,u) = Ax + Bu$, and $c(x,u) = x^\intercal M x + u^\intercal R u$, with $M \ge 0$ and $R > 0$. Given the known structure of the problem, we know that the function Q^ϕ associated with any linear policy $\phi(x) = Kx$, takes the form

$$Q^\phi = \begin{bmatrix} x \\ u \end{bmatrix}^\intercal \left(\begin{bmatrix} M & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A^\intercal P + PA + P & PB \\ B^\intercal P & 0 \end{bmatrix} \right) \begin{bmatrix} x \\ u \end{bmatrix}$$

where P solves the Lyapunov equation

$$A^\intercal P + PA + K^\intercal R K + Q = 0$$

This motivates a quadratic basis, which for the special case $n = 2$ and $m = 1$ becomes

$$\{\psi_1, \ldots, \psi_6\} = \{x_1^2, x_2^2, x_1 x_2, x_1 u, x_2 u, u^2\}$$

and there is no harm in setting $d(x,u) \equiv 0$.

In order to implement the algorithm (4.59b) we begin with selecting an input of the form

$$u_t = K_0 x_t + \xi_t \tag{4.60}$$

where K_0 is a stabilizing controller and $\xi_t = \sum_{j=1}^q v^j \sin(\omega_j t + \phi_j)$. Note that K_0 need not be the same K whose value function we are trying to evaluate.

The numerical results that follow are based on a double integrator with friction:

$$\ddot{y} = -0.1\dot{y} + u$$

which can be expressed in state space form using $x = (y, \dot{y})^\intercal$:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.61}$$

We took a relatively large cost on the input:

$$M = I \qquad R = 10$$

and gain $a_t = 1/(1+t)$.

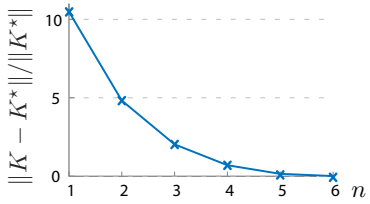


Figure 4.4: Iterations of PIA

Fig. 4.3 shows the evolution of the QSA algorithm for the evaluation of the policy $K = [-1, 0]$ using the stabilizing controller $K_0 = [-1, -2]$ and ξ in (4.60) as the sum of 24 sinusoids with random phase shifts and whose frequency was sampled uniformly between 0 and 50 rad/s. The QSA algorithm is compared with the related SA algorithm in which ξ is "white noise" instead of a deterministic signal[6]

Fig. 4.4 shows the weighted error for the feedback gains obtained using the approximate policy improvement algorithm (4.54) and the optimal controller $K^\star$ (which can be easily computed for an LQR problem). Each policy evaluation performed by the model-free algorithm (4.59). The PIA algorithm indeed converges to the optimal control gain $K^\star$.

4.5.4 A brief tour of QSA theory

While QSA theory is far simpler than stability of its stochastic ancestor, the technicalities are best left to the end of the chapter – see Section 4.9 for details. Contained here is an overview, and some guidelines for algorithm design.

QSA-ODE solidarity The *apparent noise* plays a crucial role in the analysis:

$$\widetilde{\Xi}_t = f(\Theta_t, \xi_t) - \overline{f}(\Theta_t) \tag{4.62}$$

so that

$$\tfrac{d}{dt}\Theta_t = a_t[\overline{f}(\Theta_t) + \widetilde{\Xi}_t] \tag{4.63}$$

While this is similar to the ODE (4.34), an apparent discrepancy is that the gain $\boldsymbol{a}$ is absent. There are two ways to proceed. The most obvious is to introduce a gain in (4.34): for any $t_0 > 0$, consider the solution to the ODE

$$\tfrac{d}{dt}\bar{\Theta}_t = a_t\overline{f}(\bar{\Theta}_t)\,, \qquad t \ge t_0\,,\ \bar{\Theta}_{t_0} = \Theta_{t_0} \tag{4.64}$$

This is simply a time-scaling of (4.34), using

$$\tau = g_t := \int_0^t a_r\, dr, \qquad t \ge 0. \tag{4.65}$$

For example, if $a_r = (1 + r)^{-1}$, then

$$\tau = \log(1 + t) \quad \text{and} \quad \xi(g^{-1}(\tau)) = \xi(e^\tau - 1). \tag{4.66}$$

Lemma 4.9. *Let $\{\vartheta_\tau : \tau \ge \tau_0\}$ denote the solution to (4.34) with $\vartheta_{\tau_0} = \Theta_{t_0}$, and $\tau_0 = g_{t_0}$. The solution to (4.64) is then given by*

$$\bar{\Theta}_t = \vartheta_\tau\,, \quad t \ge t_0\,, \qquad \textit{with} \quad \tau = g_t = \int_0^t a_r\, dr$$

□

[6]For implementation, both (4.59) and the linear system (4.61) were approximated using Euler's method, with time-step of 0.01s.

The second and more standard approach is to perform a time-scaling for the QSA ODE:

$$\widehat{\Theta}_\tau := \Theta(g^{-1}(\tau)) = \Theta_t\Big|_{t=g^{-1}(\tau)} \tag{4.67}$$

The chain rule of differentiation gives

$$\frac{d}{d\tau}\Theta(g^{-1}(\tau)) = f(\Theta(g^{-1}(\tau)), \xi(g^{-1}(\tau)).$$

That is, the time-scaled process solves the ODE,

$$\frac{d}{d\tau}\widehat{\Theta}_\tau = f(\widehat{\Theta}_\tau, \xi(g^{-1}(\tau)). \tag{4.68}$$

The two processes $\boldsymbol{\Theta}$ and $\widehat{\boldsymbol{\Theta}}$ differ only in time scale, and hence, proving convergence of one to θ^* proves that of the other.

These transformations are a starting point in Section 4.9.2, where you can find a proof of convergence of QSA. The required assumptions are mild:

(QSA1) The process $\boldsymbol{a}$ is non-negative, monotonically decreasing to zero, and

$$\int_0^\infty a_r\, dr = \infty$$

(QSA2) Lipschitz continuity for both $\overline{f}$ and f, and solidarity of these vector fields: there exists a constant $b_0 < \infty$, such that for all $\theta \in \mathbb{R}^d$, $T > 0$,

$$\left\| \frac{1}{T}\int_0^T f(\theta, \xi_t)\, dt - \overline{f}(\theta) \right\| \le \frac{b_0}{T}(1 + \|\theta\|)$$

(QSA3) The ODE (4.34) has a globally asymptotically stable equilibrium θ^*.

It is recommended to choose the gain in this standard family:

$$a_t = g/(1+t)^\rho \tag{4.69}$$

This satisfies (QSA1) for any $g > 0$, and $0 < \rho \le 1$.

Convergence rates and coupling We say that the rate of convergence is $1/t^{\varrho_0}$ if

$$\limsup_{t\to\infty} t^\varrho \|\widetilde{\Theta}_t\| = \begin{cases} \infty & \varrho > \varrho_0 \\ 0 & \varrho < \varrho_0 \end{cases} \tag{4.70}$$

where $\widetilde{\Theta}_t := \Theta_t - \theta^*$ is the estimation error. By careful design we can achieve $\varrho_0 = 1$, which is optimal in most cases (such as the Monte-Carlo example).

The following partial integrals play a central role when we turn to rates of convergence in Section 4.9.3: for $\theta \in \mathbb{R}^d$ and $T \ge 0$,

$$\Xi_T^{\mathrm{I}}(\theta) = \int_0^T [f(\theta, \xi_t) - \overline{f}(\theta)]\, dt \tag{4.71}$$

This is assumed bounded in T, which is justified under mild assumptions on f and $\boldsymbol{\xi}$ (see Section 4.9.1 for further discussion). Subject to further assumptions, we obtain a coupling result:

$$\lim_{t\to\infty} \|a_t^{-1}\widetilde{\Theta}_t - \Xi_t^{\mathrm{I}}(\theta^*)\| = 0 \tag{4.72}$$

which of course implies precise bounds on the rate of convergence of Θ_t to θ^*.

In addition to global asymptotic stability, to obtain the coupling bound it is assumed that the linearization matrix A^* is Hurwitz: this is defined as the $d \times d$ matrix with entries

$$A^*_{i,j} = \frac{\partial}{\partial\theta_j}\overline{f}_i\,(\theta^*)$$

Gain selection For the standard gain (4.69) there is a big difference between $\rho < 1$ and $\rho = 1$. This might be predicted from Lemma 4.9: if $\rho = 1$, the temporal transformation results in $\tau = g\log(1+t)$, which grows very slowly with t. For $\rho < 1$ we have

$$\tau = g\frac{1}{1-\rho}(1+t)^{1-\rho}$$

which grows faster than any polynomial function of t. This difference is reflected in the theory: to obtain (4.72) using $a_t = g/(1+t)^\rho$ is not difficult using $0 < \rho < 1$, and mainly requires that the linearization matrix A^* is Hurwitz. For $\rho = 1$, the gain g must be chosen sufficiently large so that $I + gA$ is Hurwitz.

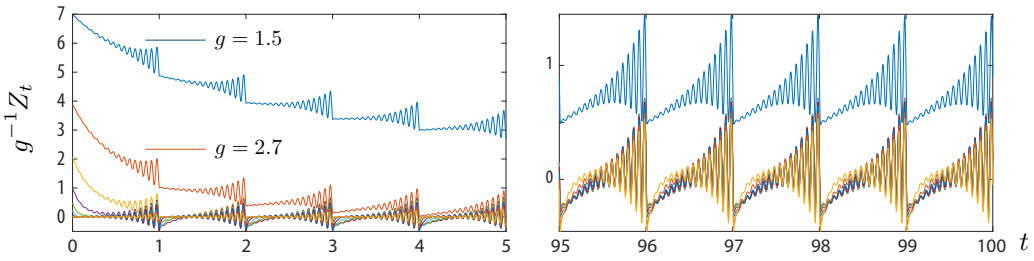


Figure 4.5: Evolution of $Z_t = (1+t)\widetilde{\Theta}_t$ using Quasi Monte-Carlo estimates for a range of gains.

Coupling for a linear approximation Consider the very special linear model (4.45), for which (4.71) is independent of θ:

$$\Xi_T^{\mathrm{I}}(\theta) = \Xi_T^{\mathrm{I}} = B\int_0^t \xi_r\,dr$$

The coupling result (4.72) is illustrated here using the simple Monte-Carlo example, whose plots are shown in Fig. 4.1. The representation (4.43) is easily modified to take the form (4.45). First, denote by $\boldsymbol{\xi}^0$ a periodic function of time whose sample paths define the uniform distribution on $[0,1]$: for any continuous function c,

$$\lim_{T\to\infty}\frac{1}{T}\int_0^T c(\xi_t^0)\,dt = \int_0^1 c(x)\,dx.$$

We previously used the sawtooth function, $\xi_t^0 = t \pmod 1$. Introduce a gain $g > 0$, and consider

$$\tfrac{d}{dt}\Theta_t = \frac{g}{1+t}[y(\xi_t^0) - \Theta_t] \tag{4.73}$$

This is of the form (4.45) with $A = -1$, $B = 1$, and $\xi_t = [y(\xi_t^0) - \theta^*]$. Prop. 4.22 below establishes the coupling result (4.72) only for $g > 1$. Figs. 4.1 and 4.2 illustrate the qualitative conclusion of Prop. 4.22. Coupling is illustrated in Fig. 4.5.

Denote $Z_t = a_t^{-1}\widetilde{\Theta}_t$. The scaled errors $g^{-1}\boldsymbol{Z}$ are compared since $\boldsymbol{\xi}$ grows linearly with g: we expect $g^{-1}Z_t \approx \int_0^t (y(\xi^0(r)) - \theta^*)$ for large t. The initial condition was set to $\Theta_0 = 10$ in each experiment.

The figure compares results using ten gains, approximately equally spaced on a logarithmic scale. The smallest gain is $g = 1.5$, and all other gains satisfy $g \geq 2$. Prop. 4.22 asserts that $|Z_t - \Xi_t^I| = O\big([1+t]^{-\delta_S}\big)$, where $\delta_S < 0.5$ for $g = 1.5$, and $\delta_S = 1$ for $g \geq 2$. The scaled errors $\{g^{-1}Z_t : 95 \leq t \leq 100\}$ are nearly indistinguishable when $g \geq 2$.

In practice it is often convenient to simply use a fixed constant $a_t \equiv \alpha > 0$. We can expect some bias in the estimates, but we will see in examples that the bias is often negligible. Low bias is suggested by Prop. 4.7.

4.5.5 Zap QSA

The convergence theory surveyed in Section 4.9 requires that the ODE (4.34) have a globally asymptotically stable equilibrium θ^*. A tight bound on the rate of convergence requires that the linearization matrix A^* is Hurwitz.

What if A^* is not Hurwitz? Or worse, what if the crucial stability assumption fails? Consider the two time-scale algorithm,

$$\begin{aligned}
\tfrac{d}{dt}\Theta_t &= \frac{g}{1+t}[-\widehat{A}_t]^{-1} f(\Theta_t, \xi_t)\,, \\
\tfrac{d}{dt}\widehat{A}_t &= \frac{1}{(1+t)^\rho}[\partial_\theta f(\Theta_t, \xi_t) - \widehat{A}_t]
\end{aligned} \tag{4.74}$$

This is called Zap-QSA, designed to mimic the Newton-Raphson flow.

The second ODE is introduced so that $\widehat{A}_t \approx A(\Theta_t)$ (following a transient). This requires $0 < \rho < 1$, meaning we use high gain for the matrix estimate. Provided we can ensure a bounded inverse, we arrive at something more closely resembling the Newton-Raphson flow:

$$\tfrac{d}{dt}\Theta_t = \frac{g}{1+t}[-A(\Theta_t)]^{-1}\big[\overline{f}(\Theta_t) + \Xi_t\big]$$

where $\Xi_t = f(\Theta_t, \xi_t) - \overline{f}(\Theta_t) + \varepsilon_t$—the error ε_t comes from the approximation $\widehat{A}_t \approx A(\Theta_t)$. If $A(\theta^*)$ is Hurwtiz, then it is invertible. Since $\overline{f}(\theta^*) = 0$, we have by the product rule

$$\partial_\theta\Big\{[-A(\theta)]^{-1}\overline{f}(\theta)\Big\}\Big|_{\theta=\theta^*} = -I$$

The most important motivation for the matrix gain in (4.74) is stability, by appealing to theory for the Newton-Raphson flow. In addition, theory in Section 4.9 suggests that the convergence rate for Zap-QSA is $1/t$, provided $g > 1$.

4.6 Gradient-Free Optimization

How can we find the minimum of a function $L(\theta)$ without computing its gradient? There are gradient-free variants of stochastic approximation available that provide an answer. Rather than wait until Chapter 9, in this section we take a look at *extremum seeking control*: a family of continuous time algorithms for gradient-free optimization [133, 6]. The preceding QSA theory inspires many possibilities.

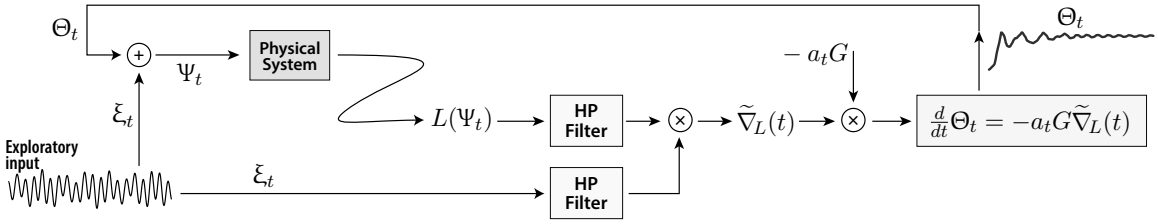


Figure 4.6: Extremum seeking control for gradient free optimization

Quasi-Gradient Descent Consider the unconstrained minimization problem

$$\min_{\theta\in\mathbb{R}^d} L(\theta). \tag{4.75}$$

It is assumed that $L\colon \mathbb{R}^d \to \mathbb{R}$ has a unique minimizer, denoted as θ^*. The goal here is to estimate θ^* based on observations of $L(\Psi_t)$, where the signal $\boldsymbol{\Psi}$ is chosen by design.

The first step is to relax our goal: find a solution to $\overline{f}(\theta^*) = 0$, where

$$\overline{f}(\theta) := \nabla L(\theta)\,, \qquad \theta \in \mathbb{R}^d\,. \tag{4.76}$$

This is equivalent to our original objective if L is convex. The two general algorithms described below are each based on the following architecture: construct an ODE of the form

$$\tfrac{d}{dt}\Theta_t = -a_t\widetilde{\nabla}_L(t) \tag{4.77}$$

where $\boldsymbol{a}$ is a non-negative gain, and $\widetilde{\nabla}_L(t)$ is designed to approximate (4.76) in an average sense:

$$\int_{T_0}^{T_1} a_t\widetilde{\nabla}_L(t)\,dt \approx \int_{T_0}^{T_1} a_t\nabla L(\Theta_t)\,dt\,, \qquad \text{for } T_1 \gg T_0 \geq 0.$$

This is achieved through algorithm design, and a particular choice for the signal $\boldsymbol{\Psi}$: the sum of two terms $\Psi_t = \Theta_t + \varepsilon\xi_t$, $t \geq 0$, where $\varepsilon > 0$, and $\boldsymbol{\xi}$ is a probing (or "exploration") signal.

Any algorithm of the form (4.77) is called *quasi Stochastic Gradient Descent* (qSGD), as it is a cousin of SGD algorithms that are also designed to approximate gradient descent.

Fig. 4.6 shows a block-diagram architecture of the second algorithm derived below: qSGD #2. The block diagram is more general than described here, since in the second algorithm the HP (high pass) filters are chosen to be pure differentiation:

$$\widetilde{\nabla}_L(t) = \tfrac{d}{dt}\xi_t \times \tfrac{d}{dt}L(\Psi_t)$$

The kind of output you can expect is illustrated in Fig. 4.7. The gradient of the function L shown has two roots: $\theta^* = 1$ is the global minimum, and $\theta^s = -1$ is a saddle point (while the derivative is zero, it is neither a local minimum or local maximum). Standard gradient descent would converge to θ^s for any initial condition $\Theta_0 \le \theta^s$. The plot on the right hand side in Fig. 4.7 shows the evolution of the estimates Θ_t using qSGD #2, initialized at $\Theta_0 = -5 < \theta^s$. The solution is not slowed down by the shallow slope of L near θ^s; this is a benefit of the probing signal.

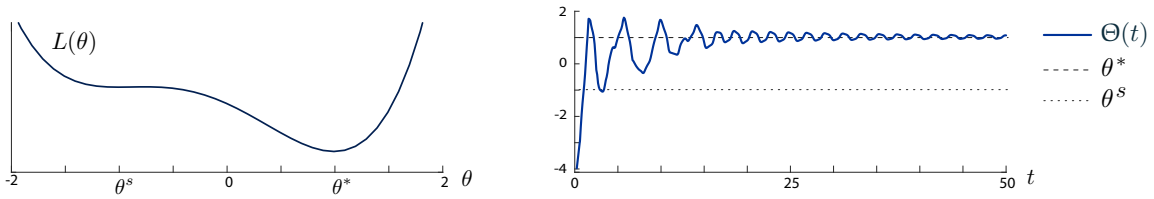


Figure 4.7: Extremum seeking control: avoiding a saddle point

For a given $\Theta \in \mathbb{R}^d$, consider the second-order Taylor expansion of the objective function:

$$L(\Theta + \varepsilon\xi_t) = L(\Theta) + \varepsilon\xi_t^\intercal \nabla L(\Theta) + \tfrac{1}{2}\varepsilon^2 \xi_t^\intercal \nabla^2 L(\Theta)\xi_t + o(\varepsilon^2).$$

Define $f_\varepsilon(\Theta, \xi) := -\xi L(\Theta + \varepsilon\xi)$. Under (4.39a) and (4.39b), It is easy to verify that

$$\overline{f}_\varepsilon(\Theta) := \lim_{T\to\infty} \frac{1}{T}\int_{t=0}^{T} f(\Theta, \xi_t)\, dt = -\varepsilon\nabla L(\Theta) + \mathrm{Err}(\varepsilon) \tag{4.78}$$

where $\|\mathrm{Err}(\varepsilon)\| = O(\varepsilon^2)$. Thus, based on the QSA ODE (4.35), the following algorithm seeks (approximate) zeros of ∇L:

Quasi Stochastic Gradient Descent #1a

For a given $d \times d$ matrix G, and initial condition Θ_0,

$$\tfrac{d}{dt}\Theta_t = -a_t \frac{1}{\varepsilon} G\xi_t L(\Psi_t) \tag{4.79a}$$

$$\Psi_t = \Theta_t + \varepsilon\xi_t \tag{4.79b}$$

The choice $G = I$ approximates the steepest descent algorithm.

Let's consider how analysis of qSGD #1a can be cast in the framework of Section 4.5. The algorithm takes the form (4.35):

$$\tfrac{d}{dt}\Theta_t = a_t f(\Theta_t, \xi_t)\,, \qquad \textit{with} \quad f(\Theta_t, \xi_t) = -\frac{1}{\varepsilon}\xi_t L(\Theta_t + \varepsilon\xi_t)$$

The representation (4.78) suggests that the algorithm will approximate gradient descent, provided the right hand side of (4.78) is Lipschitz continuous (which will hold on mild conditions on L). The problem we may face is with the function f. In many problems of interest we know that ∇L is globally Lipschitz continuous, but L is *not*. Consider for example a quadratic function. In this case f is not Lipschitz continuous, and we cannot apply QSA theory. A slightly more complex algorithm resolves this issue:

Quasi Stochastic Gradient Descent #1

For a given $d \times d$ matrix G, and initial condition Θ_0,

$$\tfrac{d}{dt}\Theta_t = -a_t \frac{1}{2\varepsilon} G\xi_t \Big\{ L(\Theta_t + \varepsilon\xi_t) - L(\Theta_t - \varepsilon\xi_t) \Big\} \tag{4.80}$$

Denoting by $a_t f(\Theta_t, \xi_t)$ the right hand side of (4.80), we can show that f is Lipschitz in its first variable whenever this is true for ∇L.

The vector field $\overline{f}_\varepsilon$ for (4.80) admits the same approximation (4.78). In fact, under mild assumptions, the two vector fields coincide. We write $\boldsymbol{\xi} \overset{\text{dist}}{=} -\boldsymbol{\xi}$ if for any continuous function g,

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T \{g(\xi_t) - g(-\xi_t)\} = 0$$

This holds for example if the components of $\boldsymbol{\xi}$ are sinusoids. Under this assumption, the limit on the left hand side of (4.78) is the same for either algorithm. However, the following consistency result can only be established for qSGD #1:

Proposition 4.10. *Suppose that the following hold for function and algorithm parameters in qSGD #1:*

(i) *Assumption (QSA1) of Section 4.9 holds:* $\boldsymbol{a}$ *is non-negative and monotonically decreasing, and*

$$\lim_{t\to\infty} a_t = 0, \qquad \int_0^\infty a_r\, dr = \infty.$$

(ii) L *has a unique minimizer* $\theta^* \in \mathbb{R}^d$.

(iii) ∇L *is globally Lipschitz continuous, and* L *is strongly convex:* (4.23) *holds for some* $m > 0$.

Then there exists $\bar{\varepsilon} > 0$ *such that for each* $0 < \varepsilon < \bar{\varepsilon}$,

(a) There is a unique root θ^*_ε *of* $\overline{f}_\varepsilon$, *satisfying* $\|\theta^*_\varepsilon - \theta^*\| \le O(\varepsilon)$.

(b) Convergence holds from each initial condition:

$$\lim_{t\to\infty} \Theta_t = \theta^*_\varepsilon$$

□

This is not to say that qSGD #1a is worthless. We simply cannot establish convergence from each initial condition. Many of the illustrations in this section are based on the first algorithm.

There are many other approaches to qSGD. For the next algorithm, it is assumed that $\{\xi_t\}$ is differentiable with respect to time. We view $\xi^\bullet(t) := (\xi_t, \frac{d}{dt}\xi_t) \in \mathbb{R}^{2m}$ as the probing signal; the reason for this definition will become apparent shortly.

The following limit is assumed to exist, and the limit is assumed to be invertible:

$$\Sigma_{\text{II}} := \lim_{T\to\infty} \frac{1}{T} \int_0^\intercal \tfrac{d}{dt}\xi_t \big[\tfrac{d}{dt}\xi_t\big]^\intercal\, dt;$$

this is indeed true for the signal chosen in (4.38).

For a given $\Theta \in \mathbb{R}^d$, we have

$$\tfrac{d}{dt} L(\Theta + \varepsilon \xi_t) = \varepsilon \tfrac{d}{dt} \xi_t^{\intercal} \nabla L\left(\Theta + \varepsilon \xi_t\right),$$

implying that

$$\begin{aligned} \tfrac{d}{dt} \xi_t \tfrac{d}{dt} L(\Theta + \varepsilon \xi_t) &= \varepsilon \tfrac{d}{dt} \xi_t \tfrac{d}{dt} \xi_t^{\intercal} \nabla L\left(\Theta + \varepsilon \xi_t\right) \\ &= \varepsilon \tfrac{d}{dt} \xi_t \tfrac{d}{dt} \xi_t^{\intercal} \left[\nabla L\left(\Theta\right) + \varepsilon \nabla^2 L\left(\Theta\right) \xi_t\right] + O(\varepsilon^2) \\ &= \varepsilon \tfrac{d}{dt} \xi_t \tfrac{d}{dt} \xi_t^{\intercal} \nabla L\left(\Theta\right) + O(\varepsilon^2). \end{aligned} \tag{4.81}$$

Therefore,

$$\lim_{T \to \infty} \frac{1}{T} \int_0^T \tfrac{d}{dt} \xi_t \tfrac{d}{dt} L(\Theta + \varepsilon \xi_t) = \varepsilon \Sigma_{\text{II}} \nabla L\left(\Theta\right) + O(\varepsilon^2). \tag{4.82}$$

This motivates

QSA Gradient Descent #2

For a given $d \times d$ matrix G, and initial condition Θ_0,

$$\tfrac{d}{dt} \Theta_t = -a_t \frac{1}{\varepsilon} G \xi_t' \tfrac{d}{dt} L(\Psi_t) \tag{4.83a}$$

$$\Psi_t = \Theta_t + \varepsilon \xi_t \tag{4.83b}$$

where $\xi_t' = \tfrac{d}{dt} \xi_t$.

This is the first algorithm that clearly fits into the block diagram Fig. 4.6: the high pass filter refers to differentiation.

In view of (4.81) and (4.82), the algorithm (4.83) is approximately equivalent to a QSA algorithm of the form (4.35), with

$$\begin{aligned} f(\Theta, \xi^\bullet) &:= -G \xi_2^\bullet \xi_2^{\bullet\intercal} \nabla L\left(\Theta + \varepsilon \xi_1^\bullet\right), \quad \xi^\bullet = (\xi_1^\bullet, \xi_2^\bullet) \in \mathbb{R}^{2m} \\ \overline{f}(\Theta) &\approx -G \Sigma_{\text{II}} \nabla L\left(\Theta\right). \end{aligned}$$

Any of these algorithms can be implemented based on observations of $\{L(\Psi_t)\}$, without knowledge of the gradient. Unfortunately, there is a complete theory only for qSGD #1, and we find in simple examples that troubles can emerge for the other two techniques:

A simple example Consider $L(\theta) = \theta^2$, whose minimum is evidently $\theta^* = 0$. We find that all three algorithms will converge to the optimum, but implementing methods #1a or #2 on a computer presents numerical challenges.

We begin with qSGD #1a, for which

$$\begin{aligned} \xi_t L(\Psi_t) &= \xi_t \left[\Theta_t^2 + 2\varepsilon \xi_t \Theta_t + \varepsilon \xi_t^2\right] \\ &= \xi_t \Theta_t^2 + \varepsilon \xi_t^2 \nabla L\left(\Theta_t\right) + \varepsilon^2 \xi_t^3 \end{aligned} \tag{4.84}$$

where the second equality follows because $\nabla L(\theta) = 2\theta$. If $\xi_t = \sqrt{2}\sin(\omega t)$ for some $\omega > 0$, then the final term is zero mean, and the first term has mean approximately zero if Θ_t is nearly constant: this is true, provided we are using a small (or vanishing) gain $\boldsymbol{a}$. Take $G = 1$ in (4.79a), giving

$$\frac{d}{dt}\Theta_t = -a_t \frac{1}{\varepsilon}\xi_t L(\Psi_t) = -a_t\Big\{\xi_t^2 \nabla L(\Theta_t) + \xi_t \varepsilon^{-1}\Theta_t^2 + \varepsilon\xi_t^3\Big\}$$

The right hand side is not Lipschitz in Θ_t, which will make it difficult to approximate using a simple Euler method.

Algorithm qSGD #1 has a similar representation, with the quadratic term removed:

$$\frac{d}{dt}\Theta_t = -a_t\Big\{\xi_t^2 \nabla L(\Theta_t) + \varepsilon\xi_t^3\Big\}$$

As predicted by Prop. 4.10, the right hand side is Lipschitz continuous in Θ_t.

The evolution of qSGD #2 is more complex:

$$\begin{aligned}\frac{d}{dt}\Theta_t &= -a_t \frac{d}{dt}\xi_t \frac{d}{dt} L(\Psi_t) \\ &= -a_t \xi'(t)[\nabla L(\Theta_t) + 2\varepsilon\xi_t][\frac{d}{dt}\Theta_t + \varepsilon\xi'(t)]\end{aligned}$$

with $\xi'(t) = \frac{d}{dt}\xi_t$. The product of Θ_t and its derivative on the right hand side suggests trouble, but theory is currently lacking.

Fig. 4.8 shows a comparison of the three algorithms for this example using $\xi_t = \sqrt{2}\sin(t)$, and exploration gain $\varepsilon = 0.05$. The first three qSGD plots were obtained based on the constant step-size $\alpha = 0.075$, which was approximately the maximal gain for which method #2 was stable. Also shown is the evolution of the ideal gradient flow $\frac{d}{dt}\vartheta_t = -\alpha\nabla L(\vartheta_t)$ (this is the vector field (4.78) with error removed, and scaled by α/ε).

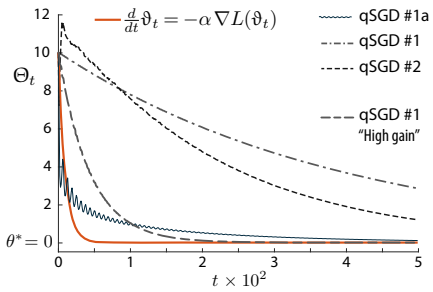


Figure 4.8: qSGD Comparison.

Each ODE was approximated using the standard Euler method, with sampling interval equal to one second. For the initial condition $\theta_0 = 10$ we see that qSGD #1a is the winner, in the sense that the parameter estimates are very close to $\theta^* = 0$ at the end of the run (1,000 samples). However, the situation changes with initial condition increased to $\theta_0 = 100$: the plot for qSGD #1 is similar, but both qSGD #1a and qSGD #2 diverge.

Convergence will hold for all algorithms by reducing the step-size, or refining the Euler approximation, but this comes with extra computational cost.

The fourth QSA plot labeled "high gain" shows the evolution of estimates using qSGD #1, with gain doubled to $\alpha = 0.15$. With this minor change, qSGD #1 emerges as the winner! This is a curse of algorithm design: we must experiment with "meta-parameters" to get quick convergence. ☐

4.7 Quasi Policy Gradient Algorithms

It is not difficult to apply these techniques to the "tamer" examples considered in Chapter 2. Consider the Mountain Car example introduced in Section 2.7.1, and the simple policy (2.55).

This cannot be optimal since it is clear the state will remain near $z^{\min}$ far longer than necessary from certain initial conditions. A more sensible policy will avoid this "western frontier". Here is one suggestion, based on a threshold θ in the interval $[z^{\min}, z^{\text{goal}}]$.

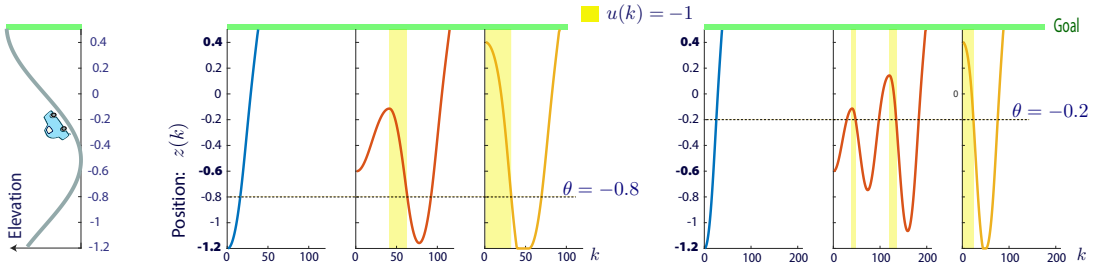


Figure 4.9: Trajectories for the Mountain Car for two policies, and three initial conditions.

With $z(k) = x_1(k)$ and $v(k) = x_2(k)$ denoting position and velocity at time-step k, define $u(k) = \phi_\theta(x(k))$ as follows:

$$u(k) = \begin{cases} 1 & \textit{if } \ x(k) + v(k) \le \theta \\ \text{sign}(v(k)) & \textit{else} \end{cases} \tag{4.85}$$

If the model is perfect, then $x(k+1) = x(k) + v(k)$; if this were a real-life application, then measurements would be noisy, so this is simply a prediction. The policy "panics" and accelerates the car towards the goal whenever the estimate of $x(k+1)$ is at or below the threshold θ.

The range of acceptable θ can be estimated by examining the graph of potential energy shown in Fig. 2.11, in the case of static input $u(k) \equiv 1$. Its minimum is at $z^\circ \approx -0.48$. Denoting $v(1) = \mathrm{F}_2(x, u)$, we have by definition of z°,

$$0 = \tfrac{d}{dz}\mathcal{U}\,(z^\circ) = g\sin(\theta(z^\circ)) - k/m = v(0) - v(1)\,, \qquad u = 1\,,\ x = (z^\circ, 0)^\intercal$$

That is, $v(1) = v(0) = 0$, which implies that the Mountain Car has stalled: $x(1) = x(0)$. We therefore cannot allow a policy for which $\phi(x^\circ) = 1$, which means that the policy ϕ_θ is not acceptable if $\theta > z^\circ$. We don't observe infinite total cost in experiments that follow because we artificially bound the value function, as explained below.

Fig. 4.9 shows trajectories of position as a function of time from three initial conditions, and with two instances of this policy: $\theta = -0.8$, and $\theta = -0.2$. The former is a much better choice from initial condition $z(0) = -0.6$: we see that the time to reach the goal is nearly twice as long when using $\theta = -0.2$ as compared to $\theta = -0.8$.

Let's see how to adapt qSGD #1a to find the optimal value of θ. A discrete-time counterpart of the recursion (4.79) is

$$\Theta_{n+1} = \Theta_n + \alpha_{n+1} G \xi_{n+1} L(\Psi_{n+1}) \tag{4.86a}$$
$$\Psi_{n+1} = \Theta_n + \varepsilon \xi_{n+1} \tag{4.86b}$$

The question is, how do we define L?

The total cost in this example coincides with the time to reach the goal. For fixed initial condition $x_0 \in \mathbb{R}^d$, we might estimate the minimum of the corresponding total cost $J_\theta(x_0)$ over θ. A natural approach is episodic: at stage n of the algorithm, we initialize the car at state x_0,

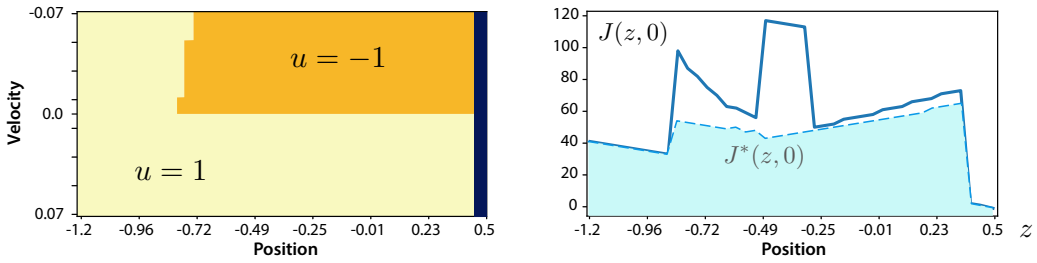


Figure 4.10: Policy ϕ^θ with $\theta = -0.8$ for MountainCar, and the total cost $J(x)$ with $x = (z, 0)$.

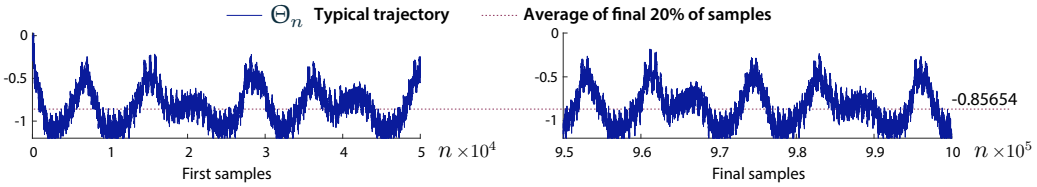


Figure 4.11: qPG #1a for the Mountain Car using the gradient-free optimization algorithm (4.86) using a large constant step-size.

and run the policy ϕ_θ using $\theta = \Psi_{n+1} = \Theta_n + \varepsilon\xi_{n+1}$. On reaching the goal state we have a measurement of $L(\Psi_{n+1}) = J_\theta(x_0)$ for this value of θ.

What if the policy is worthless? There may be values of θ for which $L(\Psi) = \infty$. It is best to modify the objective function: $L(\Psi_{n+1}) = \min\{J^{\max}, J_\theta(x_0)\}$ for a constant $J^{\max}$ of our choosing. The value $J^{\max} = 5,000$ was used in these experiments.

It may be more valuable to introduce randomization in the initial condition. In this case we introduce a second probing signal $\{\xi^x_n\}$. We proceed as above, but define $L(\Psi_{n+1}) = J_\theta(x_{n+1})$ with θ as before, but with

$$x_{n+1} = [\![x_0 + \varepsilon^x \xi^x_{n+1}]\!] \tag{4.87}$$

where the brackets again indicate projection onto the state space X. The goal then is to minimize the *average cost*: with $N \gg 1$ a large integer,

$$\mathsf{E}[J_\theta(X)] = \frac{1}{N}\sum_{n=1}^{N} \min\{J^{\max}, J_\theta(\xi^x_n)\} \tag{4.88}$$

A run using (4.86, 4.87) is shown in Fig. 4.11, with constant step-size $\alpha_n = 0.1$, $\varepsilon = 0.05$, and $\xi_n = \sin(n)$. The large step-size was chosen simply to illustrate the exotic nonlinear dynamics that emerge from this algorithm. It would seem that the algorithm has failed, since the estimates oscillate between -1.2 and -0.3 in steady-state, while the actual optimizer is $\theta^* \approx -0.8$. The dashed line shows the average of $\{\Theta_n\}$ over the final 20% of estimates. This average is very nearly optimal, since the objective function is nearly flat for θ near the optimizer.

Details on the choice of parameters in (4.87), and an approach to compute the optimal parameter θ^* will be explained shortly.

Fig. 4.12 shows results from an experiment with decaying step-size, and a minor change in the probing signal:

$$\alpha_n = 1/n^{0.75}\,, \quad \xi_n = \sin(2\pi\phi + n) \tag{4.89}$$

The exploration gain was also taken to be vanishing, using $\varepsilon_n = \alpha_n$ in (4.86b). The phase variable ϕ will be selected randomly in $[0, 1]$ when we perform repeated experiments.

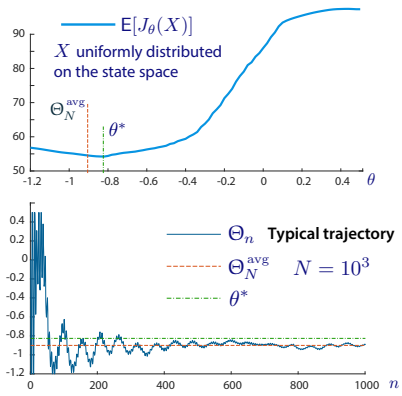


Figure 4.12: qPG #1a for Mountain Car: objective function, and typical behavior of estimates.

The signal $\{\xi_n^x = (\xi_n^z, \xi_n^v)^\intercal\}$ was chosen to cover the state space uniformly: introduce two signals that are quasi-uniform and independent on $[0, 1]$:

$$\mathcal{W}_n^v = \text{frac}(nr_v)\,, \qquad \mathcal{W}_n^z = \text{frac}(nr_z)\,,$$

where "frac" denotes the fractional part of a real number, r_v, r_z are irrational, and their ratio is also irrational. Then define

$$\begin{aligned} \xi_n^v &= \overline{v}(2\mathcal{W}_n^v - 1) \\ \xi_n^z &= z^{\min} + [z^{\text{goal}} - z^{\min}]\mathcal{W}_n^z \end{aligned} \tag{4.90}$$

The values $r_v = \pi$ and $r_z = e$ were chosen in these experiments. Also, $x_0 = 0$ and $\varepsilon^x = 0$ in (4.87), giving $x_{n+1} = \xi_{n+1}^x$.

The upper plot in Fig. 4.12 shows the average cost (4.88), with $N = 10^4$ for a range of θ. The value of θ^* was obtained by computing the minimum of this function. This approach to estimate the optimal threshold is simpler and more reliable than QSA techniques! Brute-force methods make sense if the dimension of θ is one or two; in complex situations we need a more clever search strategy.

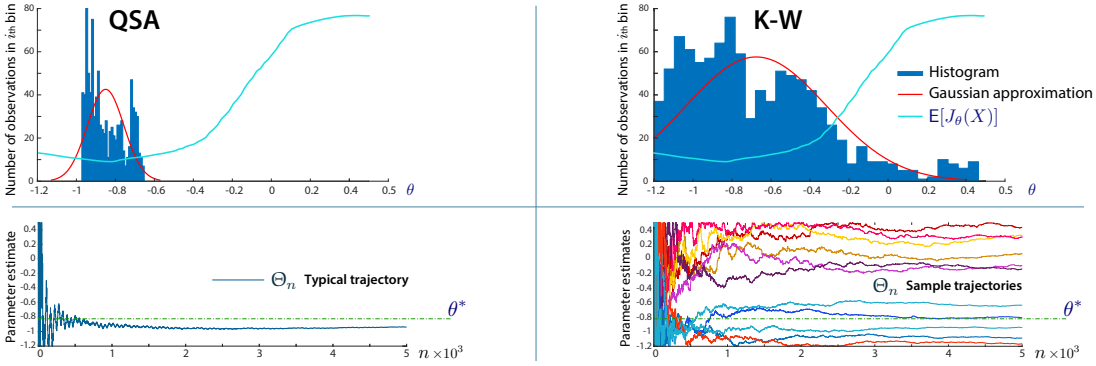


Figure 4.13: Error analysis for two PG algorithms for Mountain Car, using QSA and traditional randomized exploration.

Fig. 4.13 shows results from 10^3 independent runs, each with horizon length $\mathcal{T} = 10^4$. In each case, the parameter estimates evolve according to (4.86), to obtain estimates $\{\Theta_n^i : 1 \le n \le \mathcal{T}\,,\ 1 \le i \le 10^3\}$. The two columns are distinguished by the probing signals ξ_n and ξ_n^x. For QSA the probing signal $\boldsymbol{\xi}$ was a sinusoid, with phase ϕ selected independently in the interval $[0, 1)$, in each of the 10^3 runs (the phase appears in (4.89)). The probing sequence $\boldsymbol{\xi}^x$ was fixed as (4.90).

The results displayed in the second column used an independent sequence for the probing signals, each uniform on their respective ranges (in particular, the distribution of ξ_n was chosen uniform on the interval $[-1, 1]$ for each n). The label "K-W" refers to the algorithms of Kiefer and Wolfowitz that are also based on i.i.d. exploration (see the *Notes* section at the end of this chapter for history).

qSGD #2 is also easily adapted to this application:

$$\Theta_{n+1} = \Theta_n + \alpha_{n+1} G \xi'_{n+1} \, L(\Psi_{n+1})' \qquad (4.91a)$$

$$\Psi_{n+1} = \Theta_n + \varepsilon \xi_{n+1} \qquad (4.91b)$$

where the primes denote approximations of the derivatives appearing in (4.83a):

$$\xi'_{n+1} = \frac{1}{\delta}\big(\xi_{n+1} - \xi_n\big)\,, \quad L(\Psi_{n+1})' = \frac{1}{\delta}\big(\Psi_{n+1} - \Psi_n\big)$$

with $\delta > 0$ is the sampling interval.

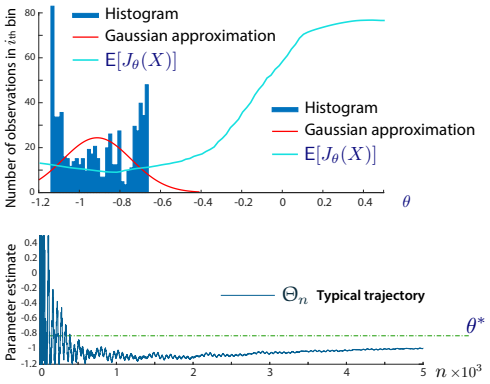


Figure 4.14: qPG for Mountain Car using eq. (4.91)

A histogram and sample path of parameter estimates are shown in Fig. 4.14, based on algorithm (4.91) with $\delta^2 = 0.5$, and all of the same choices for parameters, except that the step-size in (4.89) was reduced to avoid large initial transients: $\alpha_n = \min(1/n^{0.75}, 0.05)$. This results in $\alpha_n = 1/n^{0.75}$ for $n \ge 55$.

Based on the histogram, the performance appears slightly worse than observed for method # 1a in Fig. 4.13, but these outcomes are a product of particular choices for algorithm parameters.

What about high dimensions? The qSGD algorithms are easy to code, and quickly converge to an approximately optimal parameter for the example considered in this section. In high dimensions we can't expect to blindly apply any of these algorithms. For example, consider the choice of probing signal (4.38), where i ranges from 1 to $d = 1000$. If the frequencies $\{\omega_i\}$ are chosen in a narrow range, then the limit (4.39b) will converge very slowly. The rate will be faster if the frequencies are widely separated, but we then need a much higher resolution Euler approximation to implement an algorithm.

This challenge is well understood in the optimization literature. One approach to create a reliable algorithm is to employ *block coordinate descent* to effectively reduce the dimension of the optimization problem. This requires two ingredients:

(i) A sequence of timepoints $T_0 = 0 < T_1 < T_2 < \cdots$

(ii) A sequence of "parameter blocks" $B_k \subset \{1, \dots, d\}$ for each $k \ge 0$, where the number of elements d_B in B_i is far smaller than d.

The qSGD ODE (4.77) is modified so that $\Theta_t(i)$ is held constant on the interval $[T_k, T_{k+1})$ for $i \notin B_k$, and

$$\tfrac{d}{dt}\Theta_t(i) = -a_t[\widetilde{\nabla}_L(t)]_i \qquad i \in B_k\,, \quad t \in [T_k, T_{k+1})$$

The alternating direction method of multipliers (ADMM) employs a similar scheme.

4.8 Stability of ODEs*

This "asterisk" on this section title indicates that it contains advanced material. The stability theory here is needed if you want to fully understand why the ODE methods surveyed in this chapter are "well behaved", and the concepts will be useful in the latter chapters of the book.

We begin with the proof of a central result and a simple corollary.

4.8.1 Grönwall's Inequality

Proof of Grönwall's Inequality, Prop. 4.2. Consider first the simpler equality (4.8):

$$z_t = \gamma_t + \int_0^t \beta_s z_s \, ds$$

Observe that $\boldsymbol{\gamma}$ is continuous, under the assumption that $\boldsymbol{z}$ and $\boldsymbol{\beta}$ are continuous. We can "solve" this equation through the construction of a state space model with state $x_t = z_t - \gamma_t$, and output z_t. From the integral equation

$$x_t = \int_0^t \beta_s \big[x_s + \gamma_s\big] \, ds$$

we obtain the time varying linear state space model

$$\begin{aligned} \tfrac{d}{dt} x_t &= \beta_t x_t + \beta_t \gamma_t \,, \qquad x_0 = 0 \\ z_t &= x_t + \gamma_t \end{aligned}$$

where the initial condition $x_0 = z_0 - \gamma_0 = 0$ follows from the initial specification for $\boldsymbol{z}$. This scalar linear state space model, with "input" $u_t = \beta_t \gamma_t$, has an explicit solution, even in the time varying case:

$$x_t = \int_0^t u_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds$$

We next turn to the inequality (4.5), which we can write as

$$z_t = \alpha_t + \int_0^t \beta_s z_s \, ds - \delta_t$$

where $\delta_t \ge 0$ for each t. Define $\gamma_t = \alpha_t - \delta_t$ and obtain, with $u_t = \beta_t \gamma_t$ as before,

$$x_t = \int_0^t u_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds \le \int_0^t \beta_s \alpha_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds$$

Using $z_t = x_t + \gamma_t \le x_t + \alpha_t$ then gives (i).

(ii) If the function $\boldsymbol{\alpha}$ is non-decreasing, then from part (i) and the assumption that $\boldsymbol{\beta}$ is non-negative,

$$z_t \le \alpha_t + \alpha_t \int_0^t \beta_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds, \qquad 0 \le t \le \mathcal{T}.$$

This bound implies (ii) on substituting

$$\int_0^t \beta_s \exp\Big(\int_s^t \beta_r \, dr\Big) \, ds = \exp\Big(\int_0^t \beta_r \, dr\Big) - 1$$

□

Grönwall's Inequality implies a crude bound that is needed in approximations:

Proposition 4.11. *Consider the ODE (4.1), subject to the Lipschitz condition (4.4). Then,*

(i) *There is a constant B_f depending only on* f *such that*

$$\|\vartheta_t\| \le \big(B_f + \|\vartheta_0\|\big)e^{\ell t} - B_f\,, \qquad t \ge 0$$

(ii) *If there is an equilibrium θ^*, then for each initial condition,*

$$\|\vartheta_t - \theta^*\| \le \|\vartheta_0 - \theta^*\|e^{\ell t}\,, \qquad t \ge 0$$

Proof. We present a complete proof of (ii), and just an outline of the proof of (i), since it is similar.

If there is an equilibrium θ^*, this means that $\mathrm{f}(\theta^*) = 0$. The proof of (i) then begins with (4.2), in the form

$$\vartheta_t - \theta^* = \vartheta_0 - \theta^* + \int_0^t \mathrm{f}(\vartheta_\tau)\,d\tau\,, \qquad 0 \le t \le T$$

Under the equilibrium condition and the Lipschitz assumption,

$$\|\mathrm{f}(\vartheta_\tau)\| = \|\mathrm{f}(\vartheta_\tau) - \mathrm{f}(\theta^*)\| \le \ell\|\vartheta_\tau - \theta^*\|$$

Writing $z_t = \|\vartheta_t - \theta^*\|$, this bound combined with (4.2) gives

$$z_t \le z_0 + \ell\int_0^t z_\tau\,d\tau\,, \qquad 0 \le t \le T$$

Grönwall Inequality then gives (i): apply Prop. 4.2 (ii) using $\beta_t \equiv \ell$ and $\alpha_t \equiv z_0$.

To establish (i), take any $\theta^\bullet \in \mathbb{R}^d$ and use the Lipschitz condition to obtain

$$\begin{aligned}\|\mathrm{f}(\theta)\| &\le \|\mathrm{f}(\theta) - \mathrm{f}(\theta^\bullet)\| + \|\mathrm{f}(\theta^\bullet)\| \\ &\le \ell\|\theta) - \theta^\bullet\| + \|\mathrm{f}(\theta^\bullet)\| \\ &\le \ell\|\theta)\| + \|\theta^\bullet\| + \|\mathrm{f}(\theta^\bullet)\|\end{aligned}$$

With $\theta^\bullet$ fixed, define $B_f = [\|\theta^\bullet\| + \|\mathrm{f}(\theta^\bullet)\|]/\ell$, so that

$$\|\mathrm{f}(\theta)\| \le \ell[\|\theta\| + B_f]\,, \qquad \theta \in \mathbb{R}^d$$

Applying (4.2) then gives

$$\begin{aligned}\vartheta_t + B_f &= \vartheta_0 + B_f + \int_0^t \mathrm{f}(\vartheta_\tau)\,d\tau \\ &\le \vartheta_0 + B_f + \int_0^t [\|\vartheta_\tau\| + B_f]\,d\tau\end{aligned}$$

Grönwall's Inequality is then used to establish (i), using $z_t = \vartheta_t + B_f$ for each t, and $\alpha_t = z_0$. ☐

4.8.2 Lyapunov methods

The survey contained in Section 2.4.3 tells us much of what we need to know about Lyapunov functions. Given the goals of algorithm design, our interest is global asymptotic stability, so that the drift condition of interest is (2.35) with x^e replaced by θ^*:

$$\langle \nabla V(\theta), \mathrm{f}(\theta)\rangle < 0, \qquad \theta \ne \theta^*$$

from which we obtain convergence of ϑ by applying Prop. 2.5.

Applications to optimization in Section 4.4.1 makes heavy use of Lyapunov function techniques. We can often take $V = L$, the loss function we wish to minimize.

Often the first step is establishing consistency of an algorithm is to first show that the estimates do not "blow up": the estimates are uniformly bounded in time. The continuous-time version of this concept is defined here: the ODE is called *ultimately bounded* if there is a bounded set $S \subset \mathbb{R}^d$ such that for each initial condition ϑ^0, there is a time $T(\vartheta^0)$ such that $\vartheta_t \in S$ for $t \geq T(\vartheta^0)$.

There is naturally a Lyapunov condition to check:

$$\langle \nabla V(\theta), \mathrm{f}(\theta) \rangle \leq -1, \qquad \theta \in S^c \tag{4.92}$$

Proposition 4.12. *Assume also that there is a continuously differentiable function $V \colon \mathbb{R}^d \to \mathbb{R}_+$ satisfying (4.92) for some set $S \subset \mathbb{R}^d$. Then, $T_S(\theta) \leq V(\theta)$ for $\theta \in \mathbb{R}^d$, where*

$$T_S(\theta) = \min\{t \geq 0 : \vartheta_t \in S\}, \qquad \vartheta_0 = \theta \in \mathbb{R}^d$$

If in addition S is compact, and V is inf-compact, then the ODE (4.1) is ultimately bounded.

Proof. The bound on the first entrance time T_S is part of Exercise 2.8! It follows easily from the sample path interpretation of (4.92):

$$\tfrac{d}{dt} V(\vartheta_t) \leq -1\,, \qquad 0 \leq t \leq T_S(\theta), \qquad \vartheta_0 = \theta \in \mathbb{R}^d \tag{4.93}$$

Integrate each side from from time $t = 0$ to $t = T_N = \min(N, T_S(\theta))$ (the minimum with N is required since we don't yet know if $T_S(\theta) < \infty$). Next, apply the fundamental theorem of calculus:

$$-V(\vartheta_0) \leq V(\vartheta_{T_N}) - V(\vartheta_0) \leq -T_N$$

giving $\min(N, T_S(\theta)) \leq V(\vartheta_0)$, and the desired bound on choosing $N > V(\vartheta_0)$.

The crucial part of the proposition requires that we modify the set S. Since it is compact, and V is inf-compact, there exists $N < \infty$ such that $S \subset S_V(N) = \{\theta : V(\theta) \leq N\}$, with $S_V(N)$ also compact. Hence,

$$\langle \nabla V(\theta), \mathrm{f}(\theta) \rangle \leq -1, \qquad \theta \in \mathbb{R}^d\,,\ V(\theta) \geq N$$

In fact, we should write $V(\theta) > N$, since this corresponds to $\theta \in S_V(N)^c$, but remember the left hand side is continuous. Because $V(\vartheta_t)$ is decreasing whenever $\vartheta_t \in S_V(N)^c$, it follows that the set $S_V(N)$ is *absorbing*, which means that $\vartheta_t \in S_V(N)$ for all $t \geq T_S(\theta)$. ∎

4.8.3 The ODE at ∞

We consider here an entirely different way to verify that the ODE (4.1) is ultimately bounded.

The idea is pretty simple: to see if the ODE is ultimately bounded, we only need to consider values of ϑ that are "very big". Rather than bring out a telescope to examine these big states, we scale the state, and examine the resulting dynamics. To make this explicit requires that we make dependency on the initial condition explicit, writing $\vartheta(t; \theta_0)$ the solution to (4.1) with initial condition $\vartheta_0 = \theta_0$.

Let $r \geq 1$ by a scaling parameter (assumed large), consider the solution of the ODE with $\vartheta_0 = r\theta_0$, and scale the solution to obtain

$$\vartheta_t^r := r^{-1}\vartheta(t; r\theta_0)$$

We have $\vartheta_0^r = \theta_0$ for any $r \geq 1$, and we obtain from (4.1),

$$\tfrac{d}{dt}\vartheta_t^r = r^{-1}\tfrac{d}{dt}\vartheta(t; r\theta_0) = r^{-1}\mathrm{f}\big(\vartheta(t; r\theta_0)\big)$$

On denoting $\mathrm{f}_r(\theta) = r^{-1}\mathrm{f}(r\theta)$ for $\theta \in \mathbb{R}^d$, this becomes

$$\tfrac{d}{dt}\vartheta_t^r = \mathrm{f}_r(\vartheta_t^r) \tag{4.94}$$

Suppose that a limiting vector field exists:

$$\mathrm{f}_\infty(\theta) := \lim_{r\to\infty} \mathrm{f}_r(\theta) = \lim_{r\to\infty} r^{-1}\mathrm{f}(r\theta)\,, \qquad \theta \in \mathbb{R}^d\,, \tag{4.95}$$

and define the ODE at ∞ as the limiting case of (4.94):

$$\tfrac{d}{dt}\vartheta_t^\infty = \mathrm{f}_\infty(\vartheta_t^\infty)\,, \qquad \theta \in \mathbb{R}^d \tag{4.96}$$

Observe that by definition we have $\mathrm{f}_\infty(0) = 0$, so the origin is an equilibrium of (4.96).

Proposition 4.13. *Suppose that* f *is globally Lipschiz continuous, with Lipschitz constant* ℓ*. Suppose that the limit* (4.95) *exists for all* θ *to define a continuous function* $\mathrm{f}_\infty\colon \mathbb{R}^d \to \mathbb{R}^d$*. Then, if the origin is asymptotically stable for* (4.96)*, it follows that the ODE* (2.33) *is ultimately bounded.*

To prove the proposition we first need to better understand the special properties of the solution to (4.96):

Lemma 4.14. *Suppose that the assumptions of Prop. 4.13 hold, so in particular the origin is asymptotically stable for* (4.96)*. Then the following hold:*

(i) *For each* $\theta \in \mathbb{R}^d$ *and* $s \geq 0$*,*

$$\mathrm{f}_\infty(s\theta) = s\mathrm{f}_\infty(\theta)$$

(ii) *If* $\{\vartheta_t^\infty : t \geq 0\}$ *is any solution to the ODE* (4.96)*, and* $s > 0$*, then* $\{y_t = s\vartheta_t^\infty : t \geq 0\}$ *is also a solution, starting from* $y_0 = s\vartheta_0^\infty \in \mathbb{R}^d$*.*

(iii) *The origin is* globally *asymptotically stable for* (4.96)*, and convergence to the origin is exponentially fast: for some* $R < \infty$ *and* $\rho > 0$*,*

$$\|\vartheta_t^\infty\| \leq Re^{-\rho t}\|\vartheta_0^\infty\|\,, \qquad \vartheta_0^\infty \in \mathbb{R}^d$$

Proof. Consider first the scaling result in part (i): from the definition (4.95), with $s > 0$,

$$\mathrm{f}_\infty(s\theta) = s\lim_{r\to\infty}(sr)^{-1}\mathrm{f}(sr\theta)s\mathrm{f}_\infty(\theta)$$

The case $s = 0$ is trivial, since it is clear that $\mathrm{f}_\infty(0) = 0$. This establishes (i).

Next, write

$$\vartheta_t^\infty = \vartheta_0^\infty + \int_0^t \mathrm{f}_\infty(\vartheta_\tau^\infty)\, d\tau$$

Multiplying both sides by s and applying (i) gives (ii).

Under the assumption that the origin is asymptotically stable, there exists $\varepsilon > 0$ such that $\lim_{t\to\infty} \vartheta_t = 0$, whenever $\|\vartheta_0\| \le \varepsilon$. Moreover, the convergence is uniform: there exists $T_0 > 0$ such that

$$\|\vartheta_{T_0}\| \le \tfrac{1}{2}\varepsilon \qquad \textit{whenever } \|\vartheta_0\| \le \varepsilon$$

Next, apply scaling: for any initial condition ϑ_0, consider $y_t = s\vartheta_t^\infty$ using $s = \varepsilon/\|\vartheta_0\|$, chosen so that $\|y_0\| = \varepsilon$. Then $\|y_{T_0}\| \le \frac{1}{2}\varepsilon = \frac{1}{2}\|y_0\|$, implying

$$\|\vartheta_{T_0}\| \le \tfrac{1}{2}\|\vartheta_0\| \qquad \vartheta_0 \in \mathbb{R}^d$$

This easily implies (iii) by iteration, as follows: for any t we can write $t = nT_0 + t_0$, with $0 \le t_0 < T_0$, so that

$$\|\vartheta_t\| \le \tfrac{1}{2}\|\vartheta_{(n-1)T_0+t_0}\| \le 2^{-n}\|\vartheta_{t_0}\|$$

Prop. 4.11 gives $\|\vartheta_{t_0}\| \le e^\ell \|\vartheta_0\|$, so that

$$\|\vartheta_t\| \le 2e^\ell\, 2^{-(n+1)}\|\vartheta_0\|$$

where the right hand side has been arranged to make use of the bound $t \le (n+1)T_0$, giving $2^{-(n+1)} \le \exp(-\log(2)t/T_0)$. We arrive at the bound in (iii) with $R = 2e^\ell$ and $\rho = \log(2)/T_0$.

□

Proof of Prop. 4.13. Denote

$$\mathcal{E}(\theta) = \|\mathrm{f}(\theta) - \mathrm{f}_\infty(\theta)\|$$

so that by Lemma 4.14, with $s = \|\theta\|$,

$$s^{-1}\mathcal{E}(\theta) = \|\mathrm{f}_s(\theta/s) - \mathrm{f}_\infty(\theta/s)\|$$

Because the functions $\{\mathrm{f}_s : s \ge 1\}$ are uniformly Lipschitz continous, the right hand side converges to zero uniformly in $\theta \ne 0$. Consequently,

$$\mathcal{E}(\theta) = o(\|\theta\|)$$

Let's think about what this means: for any $\varepsilon > 0$, there exists $N(\varepsilon) < \infty$, such that $\mathcal{E}(\theta) \le \varepsilon\|\theta\|$ whenever $\|\theta\| \ge N(\varepsilon)$. From this we get the simpler looking bound:

$$\mathcal{E}(\theta) \le B_\varepsilon + \varepsilon\|\theta\|\,, \qquad \textit{where } \; B_\varepsilon = \max\{\mathcal{E}(\theta) : \|\theta\| \le N(\varepsilon)\} \tag{4.97}$$

For any initial condition ϑ_0 we compare the two solutions:

$$\vartheta_t = \vartheta_0 + \int_0^t \mathrm{f}(\vartheta_\tau)\, d\tau$$

$$\vartheta_t^\infty = \vartheta_0 + \int_0^t \mathrm{f}_\infty(\vartheta_\tau^\infty)\, d\tau$$

Write $z_t = \|\vartheta_t - \vartheta_t^\infty\|$ and use the preceding definition to obtain,

$$\begin{aligned} z_t &\le \int_0^t \|\mathrm{f}_\infty(\vartheta_\tau) - \mathrm{f}_\infty(\vartheta_\tau^\infty)\|\, d\tau + \int_0^t \mathcal{E}(\vartheta_\tau)\, d\tau \\ &\le \ell \int_0^t \|\vartheta_\tau - \vartheta_\tau^\infty\|\, d\tau + \int_0^t \mathcal{E}(\vartheta_\tau)\, d\tau \end{aligned}$$

Grönwall's Inequality in its second form (4.7) holds, with $\beta_t \equiv \ell$, and α_t the second integral, giving

$$z_t \le e^{\ell t} \int_0^t \mathcal{E}(\vartheta_\tau)\, d\tau \le e^{\ell t} \int_0^t \big\{B_\varepsilon + \varepsilon\|\vartheta_\tau\|\big\}\, d\tau$$

where the second inequality uses (4.97), with $\varepsilon > 0$ to be chosen. Prop. 4.11 gives $\|\vartheta_\tau\| \le \big\{B_f + \|\vartheta_0\|\big\}e^{\ell\tau}$, so that

$$z_t = \|\vartheta_t - \vartheta_t^\infty\| \le te^{\ell t}B_\varepsilon + \varepsilon e^{\ell t}\big\{B_f + \|\vartheta_0\|\big\}\big\{\ell^{-1}e^{\ell t}\big\}$$

And applying the triangle inequality once more,

$$\|\vartheta_t\| \le \|\vartheta_t^\infty\| + \varepsilon\ell^{-1}e^{2\ell t}\|\vartheta_0\| + B(\varepsilon, t)$$

where the value $B(\varepsilon, t)$ can be obtained by rearranging terms. Finally now we can bring in Lemma 4.14, which implies the existence of T_0 such that $\|\vartheta_t^\infty\| \le \frac{1}{2}\|\vartheta_0^\infty\| = \frac{1}{2}\|\vartheta_0\|$ when $t \ge T_0$. Hence,

$$\|\vartheta_{T_0}\| \le \Big(\tfrac{1}{2} + \varepsilon\ell^{-1}e^{2\ell T_0}\Big)\|\vartheta_0\| + B(\varepsilon, T_0)$$

Choose $\varepsilon > 0$ so small that the term in parentheses is less than $1/4$:

$$\|\vartheta_{T_0}\| \le \rho\|\vartheta_0\| + B(\varepsilon, T_0)\,, \qquad \rho = 3/4$$

Arguing as in the proof of Lemma 4.14, we can iterate to obtain for each integer n, and $t \le T_0$,

$$\|\vartheta_{nT_0+t}\| \le \rho^n\|\vartheta_t\| + \frac{1}{1-\rho}B(\varepsilon, T_0) \le \rho^n\big\{B_f + \|\vartheta_0\|\big\}e^{\ell T_0} + \frac{1}{1-\rho}B(\varepsilon, T_0)$$

This establishes ultimate boundedness, and we can choose

$$S = \Big\{\theta : \|\theta\| \le \frac{1}{1-\rho}B(\varepsilon, T_0) + 1\Big\}$$

□

4.9 Convergence theory for QSA*

We consider in this section the general nonlinear ODE (4.35), subject to the following assumptions. They are required to establish convergence of QSA, and the first steps towards obtaining bounds on the convergence rate:

(QSA1) The process $\boldsymbol{a}$ is non-negative and monotonically decreasing, and

$$\lim_{t\to\infty} a_t = 0, \qquad \int_0^\infty a_r\, dr = \infty. \tag{4.98}$$

(QSA2) The functions $\overline{f}$ and f are Lipschitz conditions: for a constant $\ell_f < \infty$,

$$\begin{aligned}\|\overline{f}(\theta') - \overline{f}(\theta)\| &\le \ell_f \|\theta' - \theta\|,\\ \|f(\theta', \xi) - f(\theta, \xi)\| &\le \ell_f \|\theta' - \theta\|, \quad \theta', \theta \in \mathbb{R}^d\,,\ \xi \in \Omega\end{aligned}$$

There exists a constant $b_0 < \infty$, such that for all $\theta \in \mathbb{R}^d$, $T > 0$,

$$\left\| \frac{1}{T}\int_0^T f(\theta, \xi_t)\, dt - \overline{f}(\theta) \right\| \le \frac{b_0}{T}(1 + \|\theta\|) \tag{4.99}$$

(QSA3) The ODE (4.34) has a globally asymptotically stable equilibrium θ^*.

Justification of (4.99) is provided next for a special class of probing signals.

4.9.1 Deterministic Markovian model

It is most convenient to assume that $\boldsymbol{\xi}$ is defined through state space dynamics,

$$\tfrac{d}{dt}\xi = \mathrm{H}(\xi) \tag{4.100}$$

where $\mathrm{H}\colon \Omega \to \Omega$ is continuous, with Ω a compact subset of Euclidean space. An extension of the mixture of sinusoidal model (4.36b) is obtained on taking the state space for the probing process equal to the K-dimensional torus: $\Omega = \{x \in \mathbb{C}^K : |x_i| = 1\,,\ \ 1 \le i \le K\}$, and $\boldsymbol{\xi}$ defined to allow modeling of excitation as a mixture of sinusoids:

$$\xi_t = [\exp(j\omega_1 t), \dots, \exp(j\omega_1 t)]^\intercal \tag{4.101}$$

with distinct frequencies, ordered for convenience: $0 < \omega_1 < \omega_2 < \cdots < \omega_K$. The dynamical system (4.100) is linear in this special case. Although deterministic, the ODE (4.100) defines a Markov process on Ω. It is ergodic, in a sense made precise in Lemma 4.15.

Lemma 4.15. *Suppose that $g\colon \mathbb{C}^K \to \mathbb{R}$ admits the Taylor series representation,*

$$g(x) = \sum_{n_1,\dots,n_K} a_{n_1,\dots,n_K} x_1^{n_1} \cdots x_K^{n_K}\,, \qquad x \in \Omega\,, \tag{4.102}$$

where the sum is over all K-length sequences in $\mathbb{Z}_+^K$, and the coefficients $\{a_{n_1,\dots,n_K}\} \subset \mathbb{C}^K$ are absolutely summable:

$$\sum_{n_1,\dots,n_K} |a_{n_1,\dots,n_K}| < \infty \tag{4.103}$$

Then,

(i) *The ergodic limit holds:*

$$a_0 = \lim_{T\to\infty} \frac{1}{T}\int_0^T g(\xi_t)\, dt = \int_0^1 \cdots \int_0^1 g\big(e^{2\pi j t_1}, \dots, e^{2\pi j t_K}\big)\, dt_1 \cdots dt_K$$

where ξ_t is defined in (4.101), and a_0 is the coefficient when $n_i = 0$ for each i.

(ii) *There exists a function* $\hat{g}\colon \mathbb{C}^K \to \mathbb{R}$ *of the form* (4.102)*:*

$$\hat{g}(x) = \sum_{n_1,\dots,n_K} \hat{a}_{n_1,\dots,n_K} x_1^{n_1} \cdots x_K^{n_K} \tag{4.104}$$

in which $|\hat{a}_{n_1,\dots,n_K}| \le |a_{n_1,\dots,n_K}|/\omega_1$ *for each coefficient, and for each* t_0, t_1*,*

$$\hat{g}(\xi_{t_0}) = \int_{t_0}^{t_1} [g(\xi_t) - a_0]\, dt + \hat{g}(\xi_{t_1}) \tag{4.105}$$

Proof. Complex exponentials and the Fourier representation are used to obtain the simple formula:

$$g(\xi_t) = \sum_{n_1,\dots,n_K} a_{n_1,\dots,n_K} \exp\bigl(\{n_1\omega_1 + \cdots + n_K\omega_K\}jt\bigr)$$

The absolute-summability assumption (4.103) justifies Fubini's Theorem:

$$\int_{t_0}^{t_1} [g(\xi_t) - a_0]\, dt = \sum_{n_1,\dots,n_K} a_{n_1,\dots,n_K} \int_{t_0}^{t_1} \exp\bigl(\{n_1\omega_1 + \cdots + n_K\omega_K\}jt\bigr)\, dt = \hat{g}(\xi_{t_0}) - \hat{g}(\xi_{t_1})$$

where $\hat{g}$ is given by (4.104) with $\hat{a}_0 = 0$ (that is, $n_k = 0$ for each k), and for all other coefficients

$$\hat{a}_{n_1,\dots,n_K} = a_{n_1,\dots,n_K}\{n_1\omega_1 + \cdots + n_K\omega_K\}^{-1} j$$

□

ODE QSA solidarity The importance of (4.105) is that it implies a rate of convergence in the ergodic limit:

$$\frac{1}{T}\int_0^T g(\xi_t)\, dt = a_0 + \frac{1}{T}\Bigl(\hat{g}(\xi_T) - \hat{g}(\xi_0)\Bigr)$$

Since $\hat{g}$ is bounded, this means the convergence rate is $1/T$. It will be assumed that a similar approximation holds for the QSA vector field, for every $\theta \in \mathbb{R}^d$:

$$\hat{f}(\theta, \xi_{t_0}) = \int_0^T [f(\theta, \xi_t) - \overline{f}(\theta)]\, dt = +\frac{1}{T}\Bigl(\hat{f}(\theta, \xi_T) - \hat{f}(\theta, \xi_0)\Bigr) \tag{4.106}$$

This leads to the solidarity we seek between the QSA ODE (4.35) and the standard ODE (4.34) with the mean vector field $\overline{f}$.

4.9.2 Stability

The first step in establishing convergence of QSA is to show that the solutions are bounded in time. Two approaches can be borrowed from previous sections: Lyapunov function techniques, or the ODE at ∞ introduced in Section 4.8.3. We deal exclusively with $\widehat{\boldsymbol{\Theta}}$, which solves the gain-free ODE (4.68). it is on the 'right' time scale for comparison with $\boldsymbol{\vartheta}$, the solution of (4.34).

When applying the techniques of [39, 36] we require the vector field at ∞:

$$\overline{f}_\infty(\theta) := \lim_{r\to\infty} r^{-1}\overline{f}(r\theta)\,, \qquad \theta \in \mathbb{R}^d \tag{4.107}$$

Theorem 4.16. *Suppose that Assumptions (QSA1)–(QSA3) hold, along with the conditions of Prop. 4.13 in the context of this section:*

(i) *The limit (4.107) exists for all θ to define a continuous function $\overline{f}_\infty\colon \mathbb{R}^d \to \mathbb{R}^d$*

(ii) *The origin is globally asymptotically stable for the ODE at ∞:*

$$\tfrac{d}{dt}\vartheta_t^\infty = \overline{f}_\infty(\vartheta_t^\infty)\,, \qquad \theta \in \mathbb{R}^d$$

Then the solution to (4.35) converges to θ^* for each initial condition.

When applying Lyapunov function techniques we impose the following:

(QSV1) There exists a continuous function $V : \mathbb{R}^d \to \mathbb{R}_+$ and a constant $c_0 > 0$ such that, for any initial condition ϑ_0 of (4.34), and any $0 \le T \le 1$, the following bounds hold whenever $\|\vartheta_s\| > c_0$,

$$V(\vartheta_{s+T}) - V(\vartheta_s) \le -T\|\vartheta_s\|.$$

The Lyapunov function is Lipschitz continuous: there exists a constant $\ell_V < \infty$ such that $\|V(\theta') - V(\theta)\| \le \ell_V\|\theta' - \theta\|$ for all θ, θ'.

Assumption (QSV1) ensures that there is a Lyapunov function V with a strictly negative drift whenever $\boldsymbol{\vartheta}$ escapes a ball of radius c_0. This is used to establish boundedness of $\boldsymbol{\Theta}$.

Verifying (QSV1) for a linear system. Consider the ODE (4.34) in which $\overline{f}(x) = Ax$ with A a Hurwitz $d \times d$ matrix. There is a quadratic function $V_2(x) = x^\intercal P x$ satisfying the Lyapunov equation $PA + A^\intercal P = -I$, with $P > 0$. Consequently, solutions to (4.34) satisfy

$$\tfrac{d}{dt}V_2(\vartheta_t) = -\|\vartheta_t\|^2$$

Choose $V = k\sqrt{V_2}$ with $k > 0$, so that by the chain rule

$$\tfrac{d}{dt}V(\vartheta_t) = -\frac{k}{2}\frac{1}{\sqrt{V_2(\vartheta_t)}}\|\vartheta_t\|^2$$

For $k > 0$ sufficiently large we obtain

$$\tfrac{d}{dt}V(\vartheta_t) \le -\|\vartheta_t\|$$

and hence this V is a Lipschitz solution to (QSV1), for any $c_0 > 0$. ☐

Theorem 4.17. *Under Assumptions (QSA1)–(QSA3) and (QSV1), the solution to (4.35) converges to θ^* for each initial condition.*

Define $\vartheta^\tau(w)$, $w \ge \tau$, to be the unique solution to (4.34) 'starting' at $\widehat{\Theta}_\tau$:

$$\tfrac{d}{dw}\vartheta^\tau(w) = \overline{f}(\vartheta^\tau(w)),\ w \ge \tau,\ \vartheta_\tau^\tau = \widehat{\Theta}_\tau. \tag{4.108}$$

The following result is required to prove Thm. 4.17.

Lemma 4.18. *Under the assumptions of Thm. 4.17, for any $T > 0$,*

$$\lim_{\tau\to\infty} \sup_{v\in[0,T]} \Big\| \int_\tau^{\tau+v} \big[f(\widehat{\Theta}_w, \xi(g^{-1}(w)) - \overline{f}(\widehat{\Theta}_w) \big]\, dw \Big\| = 0$$

$$\lim_{\tau\to\infty} \sup_{v\in[0,T]} \|\widehat{\Theta}_{\tau+v} - \vartheta^\tau(\tau+v)\| = 0.$$

The proof of Lemma 4.18 can be found in [24], and is similar to results in the SA literature (e.g., Lemma 1 in Chapter 2 of [35]).

Proof of Thm. 4.17. The first step in the proof is to establish ultimate boundedness of $\widehat{\Theta}_\tau$: there exists $b < \infty$ such that for each $\theta \in \mathbb{R}^d$, there is a T_θ such that

$$\|\widehat{\Theta}_\tau\| \le b \quad \text{for all } \tau \ge T_\theta\,,\ \widehat{\Theta}_0 = \theta$$

The (lengthy) proof is contained in [24].

Thus, for $\tau \ge T_\theta$, $\|\vartheta^\tau_\tau\| = \|\widehat{\Theta}_\tau\| \le b$. By the definition of global asymptotic convergence, for every $\varepsilon > 0$, there exists a $\mathcal{T}_\varepsilon > 0$, independent of the value ϑ^τ_τ, such that $\|\vartheta^\tau(\tau+v) - \theta^*\| < \varepsilon$ for all $v \ge \mathcal{T}_\varepsilon$. Lemma 4.18 gives,

$$\limsup_{\tau\to\infty} \|\widehat{\Theta}_{\tau+\mathcal{T}_\varepsilon} - \theta^*\| \le \limsup_{\tau\to\infty} \|\widehat{\Theta}_{\tau+\mathcal{T}_\varepsilon} - \vartheta^\tau_{\tau+\mathcal{T}_\varepsilon}\| + \limsup_{\tau\to\infty} \|\vartheta^\tau_{\tau+\mathcal{T}_\varepsilon} - \theta^*\| \le \varepsilon.$$

Since ε is arbitrary, we have the desired limit. ∎

4.9.3 Gain Selection

This section is devoted to choice of gain (equivalently, stepsize) for the QSA algorithm (4.35). It is assumed $\boldsymbol{\Theta}$ converges to a unique value θ^* from each initial condition $\Theta_0 \in \mathbb{R}^d$. Sufficient conditions for this are given in Thm. 4.16 or Thm. 4.17. Our interest is in optimizing the rate of convergence (4.35) (recall the definition (4.70)).

In some cases we can establish boundedness of the scaled error $Z_t = (1+t)\widetilde{\Theta}_t$. The convergence rate is $1/t$ in this case, since

$$\widetilde{\Theta}_t = Z_t/(1+t)$$

Sometimes we can identify a "covariance" for the scaled error:

$$\overline{\Sigma}_\theta := \lim_{T\to\infty} \frac{1}{T} \int_0^T Z_t Z_t^\intercal\, dt. \tag{4.109}$$

Which implies in particular that

$$\lim_{t\to\infty} t^2 \|\Theta_t - \theta^*\|^2 = \operatorname{trace}(\overline{\Sigma}_\theta) \tag{4.110}$$

This is far faster than what is obtained using standard stochastic approximation: we must introduce an expectation, and settle for a much slower rate:

$$\lim_{t\to\infty} t\mathsf{E}[\|\Theta_t - \theta^*\|^2] = \operatorname{trace}(\Sigma_\theta) \tag{4.111}$$

That is, the rate is $1/\sqrt{t}$ rather than $1/t$. This theory is fully developed in Section 9.3, along with a formula for the covariance matrix Σ_θ.

Justification for approximating QSA with a linearization is provided next.
Remainder is WIP, based on a journal article in preparation
To obtain bounds on the rate of convergence for QSA we need to strengthen (4.99):

(QSA4) The probing signal is the solution to (4.100), with Ω a compact subset of Euclidean space. It has a unique invariant measure π on Ω, and satisfies the following ergodic theorems for the functions of interest, for each initial condition $\xi_0 \in \Omega$:

(i) For each θ there exists a function $\hat{f}(\theta, \cdot)$ satisfying

$$\hat{f}(\theta, \xi_{t_0}) = \int_{t_0}^{t_1} [f(\theta, \xi_t) - \overline{f}(\theta)]\, dt + \hat{f}(\theta, \xi_{t_1}), \qquad 0 \le t_0 \le t_1 \tag{4.112}$$

with

$$\overline{f}(\theta) = \int_\Omega f(\theta, x)\, \pi(dx) \qquad \textit{and} \qquad 0 = \int_\Omega \hat{f}(\theta, x)\, \pi(dx)$$

(ii) The function $\hat{f}$, and derivatives $\partial_\theta \overline{f}$ and $\partial_\theta f$ are Lipschitz continuous in θ. In particular, $\hat{f}$ admits a derivative $\widehat{A}$ satisfying

$$\widehat{A}(\theta, \xi_{t_0}) = \int_{t_0}^{t_1} [A(\theta, \xi_t) - A(\theta)]\, dt + \widehat{A}(\theta, \xi_{t_1}), \qquad 0 \le t_0 \le t_1$$

where $A(\theta, \xi) = \partial_\theta f(\theta, \xi)$ and $A(\theta) = \partial_\theta \overline{f}\,(\theta)$ was defined in (**??**). Lipschitz continuity is assumed uniform with respect to the exploration process: increasing the constant $\ell_f < \infty$ in (QSA2) if necessary,

$$\begin{aligned}
\|\hat{f}(\theta', \xi) - \hat{f}(\theta, \xi)\| &\le \ell_f \|\theta' - \theta\| \\
\|A(\theta', \xi) - A(\theta, \xi)\| &\le \ell_f \|\theta' - \theta\| \\
\|\widehat{A}(\theta', \xi) - \widehat{A}(\theta, \xi)\| &\le \ell_f \|\theta' - \theta\|, \qquad \theta', \theta \in \mathbb{R}^d, \ \xi \in \Omega
\end{aligned}$$

(iii) Denote $\Upsilon_t = [\widehat{A}(\theta^*, \xi_0) - \widehat{A}(\theta^*, \xi_t)] f(\theta^*, \xi_t)$. The following limit exists:

$$\overline{\Upsilon} := \lim_{T \to \infty} \frac{1}{T} \int_0^T \Upsilon_t\, dt = -\int_\Omega \widehat{A}(\theta^*, x) f(\theta^*, x)\, \pi(dx)$$

and the partial integrals are bounded:

$$\sup_{T \ge 0} \Big| \int_0^T [\Upsilon_t - \overline{\Upsilon}]\, dt \Big| < \infty$$

□

Assumption (QSA4) (iii) is imposed because the vector $\overline{\Upsilon}$ arises in an approximation for the scaled error Z_t: the first appearance of Υ_t is in Prop. 4.19. The assumption is not much stronger than the others. In particular, the partial integrals will be bounded if there is a bounded solution to Poisson's equation:

$$\widehat{\Upsilon}_{t_0} = \int_{t_0}^{t_1} [\Upsilon_t - \overline{\Upsilon}]\, dt + \widehat{\Upsilon}_{t_1}$$

Recall the definition of the scaled error Z_t introduced in (**??**), and denote for $t \ge 0$,

$$Y_t := Z_t - \Xi_t^{\mathrm{I}}(\Theta_t) \tag{4.113}$$

Proposition 4.19. *Suppose that $r_t \le ba_t$ for a constant b, and all $t \ge 0$. Then, the vector-valued process $\boldsymbol{Y}$ satisfies the differential equation,*

$$\tfrac{d}{dt} Y_t = a_t \left[A^* Y_t + \Delta_t^Y - \Upsilon_t + A^* \Xi_t^{\mathrm{I}} \right] + r_t [Y_t + \Xi_t^{\mathrm{I}}] \tag{4.114}$$

where $\Xi_t^{\mathrm{I}} = \Xi_t^{\mathrm{I}}(\theta^*)$, and $\|\Delta_t^Y\| = o(1 + \|Y_t\|)$ as $t \to \infty$. That is, for scalars $\{\varepsilon_t^Y\}$,

$$\|\Delta_t\| \le \varepsilon_t^Y \{1 + \|Y_t\|\}, \qquad t \ge t_0$$

with $\varepsilon_t^Y \to 0$ as $t \to \infty$.

Proposition 4.20. *Suppose that (QSA1)–QSA3) hold, and that solutions to (4.35) converge to θ^* for each initial condition. Suppose in addition that $\overline{f}$ is differentiable, and its derivative $A = \partial_\theta \overline{f}$ is also Lipschitz continuous. Then, the scaled error admits the representation*

$$\tfrac{d}{dt} Z_t = \left[r_t I + a_t A(\bar{\Theta}_t) \right] Z_t + a_t \Delta_t + \widetilde{\Xi}_t, \qquad Z_{t_0} = 0 \tag{4.115}$$

where $r_t = -\frac{d}{dt} \log(a_t)$, $\widetilde{\Xi}_t = f(\Theta_t, \xi_t) - \overline{f}(\Theta_t)$, and $\|\Delta_t\| = o(\|Z_t\|)$ as $t \to \infty$.

In particular, setting $A^* = A(\theta^*)$,

(i) *With $a_t = g/(1+t)$,*

$$\tfrac{d}{dt} Z_t = a_t \left[g^{-1} I + A^* \right] Z_t + a_t \Delta_t + \widetilde{\Xi}_t \tag{4.116}$$

(ii) *For any $\rho \in (0,1)$, using the gain $a_t = g/(1+t)^\rho$ gives*

$$\tfrac{d}{dt} Z_t = a_t A^* Z_t + a_t \Delta_t + \widetilde{\Xi}_t \tag{4.117}$$

where the definition of Δ_t is different in each appearance, but in each case satisfies $\|\Delta_t\| = o(\|Z_t\|)$.

The challenge in applying Prop. 4.20 is that the "noise" process $\widetilde{\Xi}_t$ appearing in (4.115) is non-vanishing, and is not scaled by a vanishing term. We show here that this term can be removed through a change of variables.

Proposition 4.21. *Suppose that (QSA1)–(QSA4) hold, and that A^* is Hurwitz. Suppose the gain is $a_t = 1/(1+t)^\rho$, with $\rho < 1$. Then the following hold:*

$$\begin{aligned} Z_t &= \bar{Y} + \Xi_t^{\mathrm{I}} + o(1) \\ \Theta_t &= \theta^* + a_t [\bar{Y} + \Xi_t^{\mathrm{I}}] + o(a_t) \end{aligned} \tag{4.118}$$

Proposition 4.22. *Suppose that (QSA1)–(QSA4) hold, and that $I + A^*$ is Hurwitz. Suppose the gain is $a_t = 1/(1+t)^\rho$, with $\rho = 1$. Then the following hold:*

$$\begin{aligned} Z_t &= \bar{Y} + \Xi_t^{\mathrm{I}} + o(1) \\ \Theta_t &= \theta^* + a_t [\bar{Y} + \Xi_t^{\mathrm{I}}] + o(1/t) \end{aligned} \tag{4.119}$$

4.9.4 Ruppert-Polyak averaging

4.10 Exercises

4.1 Compute the Newton-Raphson vector field $\mathrm{f}^{\mathrm{NRf}}$ defined in (4.14) for the three scalar examples: $\mathrm{f}(x) =$
(a) $-\nabla J\,(x)$ with $J(x) = x^2(1 + (x + 10)^2)$
(b) $-\nabla J\,(x)$ with $J(x) = \log(e^x + e^{-x})$
(c) $\sin(x)$

In each case,

- Plot $\mathrm{f}^{\mathrm{NRf}}(\theta)$ as a function of θ
- Obtain the roots of f and $\mathrm{f}^{\mathrm{NRf}}$
- *Identify the regions of attraction*: we say that θ is in the *region of attraction* an equilibrium θ° for the Newton-Raphson flow if

$$\lim_{t\to\infty} \Theta_t = \theta^\circ$$

where Θ_t is the solution to (4.13) at time t, with initial condition $\Theta_0 = \theta$.
Describe the region of attraction for each root of $\mathrm{f}^{\mathrm{NRf}}$.

4.2 Let's revisit part (a) of Exercise 4.1 to see some of the difficulties minimizing the function $J(x) = x^2(1 + (x + 10)^2)$ using gradient descent. One problem is that it is not convex, and also has multiple local minimum. Another is that its gradient has cubic growth, which introduces potential numerical problems.

(a) Code an Euler approximation of gradient descent $\frac{d}{dt}\Theta = -\nabla J(\Theta)$. Perform multiple runs, with varying initial condition (it will eventually fail when you choose an initial condition too large).

(b) Introduce a weighting function $w\colon \mathbb{R} \to [1, \infty)$, and consider the normalized algorithm:

$$\tfrac{d}{dt}\Theta = -\frac{1}{w(\Theta)}\nabla J(\Theta)$$

Choose a continuous weighting function so that the right hand side is globally Lipschitz continuous, and test the Euler approximation with a range of initial conditions.

We will revisit this example once more in Exercise 9.2.

4.3 Pendulum swing up

4.4 Rowing game

4.5 Oscillator game

4.11 Notes

Work in progress—comments welcome

ODE methods for algorithm design This is today a significant trend in both RL and ML.

Essays to write on the amazing results on acceleration and Runge-Kutta methods, and more ... [193, 224, 181]

Along side ODE approaches to algorithm design are SDE (stochastic differential equation) techniques [**?**].

NR flow The ODE (4.13) was introduced in the economics literature, which led to the comprehensive analysis by Smale [186] for smooth $\overline{f}$. The term *Newton-Raphson flow* for (4.13) was introduced in the deterministic control literature [183, 218]. The Zap SA algorithm was introduced at the same time, and based on the same ODE [71].

QSA Much of Sections 4.5 and 4.6 and section 4.9.3 is adapted from [24, 25], which was inspired by the prior results in [142, 182]; [52] contains applications to gradient-free optimization with constraints. The first appearance of QSA methods appears to have originated in the domain of quasi-Monte Carlo methods applied to finance; see [128, 129].

Ruppert-Polyak averaging was introduced independently by their namesakes [174, 167]. However, this work has nothing to do with QSA, but concerns optimizing the covariance Σ_θ appearing in (4.111) for stochastic approximation—see the Notes section of Chapter 9 for more background. The application of averaging techniques for rate optimization in QSA appears to be new.

The function $\hat{g}$ in Lemma 4.15 (ii) is precisely the solution to Poisson's equation, with forcing function $\tilde{g} = g - a_0$, that appears in theory of simulation of Markov processes, average-cost optimal control, and stochastic approximation [91, 7, 145, 23].

The phrase *ODE method* is frequently tributed to Ljung [134], though most authors use this to mean a method of analysis, rather than a technique for algorithm design.

The "ODE@∞" (4.95) was introduced in [39] for stability verification in stochastic approximation: Prop. 4.13 is a very special case of the Borkar-Meyn Theorem [39, 36], which has been refined considerably in recent years [169, 170]. The use of abstract ODE models to verify stability of stochastic recursions also appears in queueing networks [58, 59, 147] and MCMC [85]. We will revisit this approach to stability verification in Chapter 9.

SGD and Extremum seeking control Gradient-free optimization has been studied in two, seemingly disconnected lines of work. The first line of work, typically known as "bandit optimization" (see e.g., [84, 10, 44]) leverages a *stochastic* estimate of the gradient, based on a single or multiple evaluations of the objective function. The ideas originate in the paper of Kiefer and Wolfowitz [114] (see see [36, 188] for more history as well as refined algorithms). These techniques have been analyzed extensively using tools similar to the classical SA approach, with similar conclusion on the high variance of the estimates [50].

The second line of work, typically termed "extremum-seeking control" (ESC) [133, 6], leverages a *deterministic* estimate of the gradient. The gradient-free optimization techniques surveyed in Section 4.6 are stylized versions of the ESC approach. The gain $\boldsymbol{a}$ is typically assumed constant in this literature, and there is a large literature on how to improve the algorithm, such as through the introduction of a linear filter on the measurements $\{L(\Psi_t)\}$. Stability of ESC was analyzed in e.g., [123, 216]; see [6] for a comprehensive overview of the methods.

Policy gradient techniques are traditionally posed in a stochastic setting, in which $\boldsymbol{\xi}$ is i.i.d. (independent and identically distributed). The most popular approach is the Actor-Critic method, in which a value function approximation algorithm such as TD-learning acts as a subroutine. There is an enormous literature, and it is best to refer to [26, 194] for history, as well as the recent work [137].

Regret analysis of stochastic and nonstochastic multi-armed bandit problems is the subject of [45].

Chapter 5

Value Function Approximations

We now have all the preliminaries necessary to describe reinforcement learning algorithms designed for value function approximation.

The approximation techniques in this book are built around a parameterized family: for approximating the Q-function $Q^\star$ (defined in (3.6) for the total cost criterion), the family is denoted $\{Q^\theta : \theta \in \mathbb{R}^d\}$, where $d \geq 1$ is the dimension of the function class. Standard examples are discussed inSection 5.1. In this chapter, the focus is on algorithms based on optimization: an algorithm designed to compute the *optimal* parameter θ^* will be based on some loss function $\mathcal{E}(\theta)$, with $\theta^* = \arg\min_\theta \mathcal{E}(\theta)$. The algorithm may be recursive, in which case it generates a sequence of parameter estimates $\{\theta_n\}$, designed so that $\theta_n \to \theta^*$ as $n \to \infty$.

Reinforcement learning algorithms are typically designed to be *model free*, in which the inputs to the algorithm consist of three terms: $\{u(k)\}$ the input sequence to the control system, the sequence of observed costs $c(x(k), u(k))$, and observed *features* that depend on the class of algorithms. For the linear parameterization $Q^\theta(x,u) = \sum_i \theta_i \psi_i(x,u)$, the sequence of features is the d-dimensional sequence $\{\psi(x(k), u(k))\}$. Fig. 5.1 is included to emphasize that these are the only inputs to the algorithm. We don't require a model, and the state sequence $\{x(k)\}$ may not be fully observed.

For any approximation Q^θ we define a policy inspired by the Bellman equation, and in particular eq. (3.8):

$$\phi^\theta(x) = \arg\min_u Q^\theta(x,u)\,, \qquad x \in \mathsf{X} \tag{5.1}$$

That is,

$$u(k) = \arg\min_u Q^\theta(x(k),u)$$

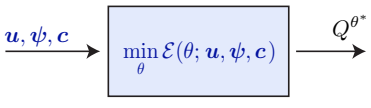


Figure 5.1: Online Q-learning: inputs are observed features, and costs or rewards

It is assumed that $Q^\theta(x(k),u)$ depends on $x(k)$ only through the realized cost and features $\boldsymbol{\psi}$ illustrated in (5.1). Hence we may regard $\phi^\theta(x(k))$ as a function of $(c(x(k)), \psi(x(k),u(k)))$. This is an important distinction if the space on which $\boldsymbol{x}$ evolves is much larger than the dimension of ψ.

Reinforcement learning can be regarded as collection of system identification for control. In standard control textbooks there is a two-step process: 1. identify a model, such as the ARMA model (2.4), and 2. design a control solution based on this model (perhaps

through state feedback). Described in this chapter is a one step process in which we estimate a value function or Q-function, and from this we immediately obtain a policy.

What's a Good Approximation? If you have read Chapter 3 on optimal control, you surely want to learn how to approximation the Q-function $Q^\star$. However, you are more eager to obtain an estimate of the optimal policy:

$$\phi^\star(x) = \arg\min_u Q^\star(x,u)\,, \qquad x \in \mathsf{X}$$

A few things to keep in mind as we evaluate an algorithm:

(i) *Approximation fidelity.* We do not need a highly accurate approximation of the Q-function if our goal is to obtain a policy that is approximately optimal. Rather, the goal is that the performance of the policy ϕ^{θ^*} is approximately optimal. Ideally then, $\mathcal{E}(\theta)$ would be some measure of policy performance. The mean-square Bellman error (5.5) is a common surrogate, since estimating performance may be computationally costly.

(ii) *Policy evaluation.* It may not be very expensive to compute or approximate $\mathcal{E}(\theta_n)$ based on a model, in which case we can keep a tally of performance for selected iterations $\{\mathcal{E}(\theta_{n_k}) : k \ge 1\}$. We then select those policies among $\{\phi^{\theta_{n_k}} : k \ge 1\}$ with the highest performance. Most likely we will do further testing, following the guidelines in Section 2.2, and suggestions found in Section 5.1.5.

5.1 Function Approximation Architectures

This section might be viewed as the briefest crash course on machine learning. See [32] for a more leisurely introduction to the function approximation concepts covered here.

The goal is to approximation a function $H\colon \mathsf{Z} \to \mathbb{R}$, where interpretation of H and the definition of the set of points Z depends on context. This is regarded as a *learning* problem when the estimate is based on data gathered in an experiment. For example, we might select H to be the Q-function defined in (3.6), and $\mathsf{Z} = \mathsf{X} \times \mathsf{U}$. The data will be obtained from experiments on the control system: the input applied to the system, along with functions of the resulting state process. The techniques described here for function approximation require a few ingredients:

(i) A *function class* $\mathcal{H}$. Three examples are described below: a d-dimensional linear function class, d-dimensional non-linear function class defined by a neural network, and one infinite-dimensional class: the *reproducing kernel Hilbert space* (RKHS).

(ii) For each $h \in \mathcal{H}$ we associate a non-negative "loss", defined by a *loss function* $\mathcal{E}(h)$. It is designed so that $\mathcal{E}(h)$ is small when $h = H$; our approximation is lousy if $\mathcal{E}(h)$ is very large. We impose just one requirement on this loss function: assumed given are samples $\{z_i : 1 \le i \le N\} \subset \mathsf{Z}$, and $\mathcal{E}$ depends only on h evaluated at the samples. Consequently, rather than thinking of the domain of $\mathcal{E}$ as the abstract collection $\mathcal{H}$, it is a mapping $\mathcal{E}\colon \mathbb{R}^N \to \mathbb{R}$, with

$$\mathcal{E}(h) = \mathcal{E}(h(z_1), \ldots, h(z_N)) \tag{5.2}$$

(iii) An algorithm to obtain the minimizer of $\mathcal{E}(h)$ over $h \in \mathcal{H}$. This book is filled with techniques for constructing algorithms, and techniques to obtain insight on their rate of convergence.

We begin with two examples of loss functions, and then three examples of the function class $\mathcal{H}$.

5.1.1 Function approximation based on training data

Curve fitting Suppose that we have noisy observations of the function:

$$y_i = H(z_i) + d_i$$

where the disturbance $\{d_i\}$ is not too large, and has nice statistical properties (for example, its average is close to zero). The sequence $\{(z_i, y_i) : 1 \le i \le N\}$ is called *training data*. The quadratic loss function is defined by

$$\mathcal{E}(h) = \frac{1}{N}\sum_{i=1}^{N}[y_i - h(z_i)]^2\,, \qquad h \in \mathcal{H} \tag{5.3}$$

If $\mathcal{E}(h^*) = 0$ then the function exactly matches the observations: $h^*(z_i) = y_i$ for each i. This is good news in the disturbance-free setting ($d_i = 0$ for each i), so that $h^*(z_i) = H(z_i)$ for each i.

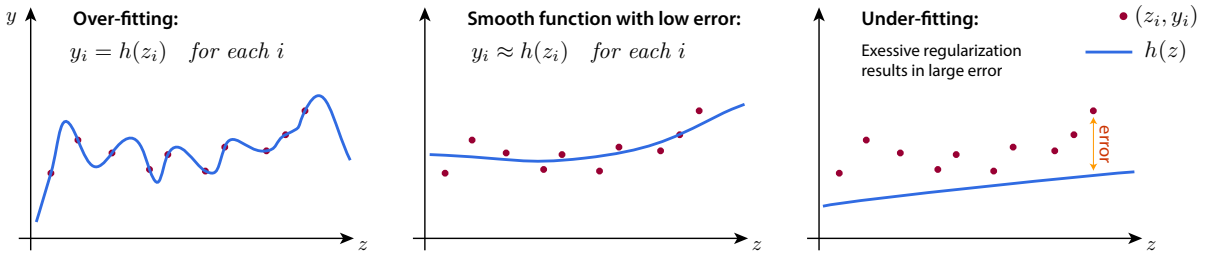


Figure 5.2: Three attempts to approximate the data $\{z^i, y^i\}$ with a smooth function.

Fig. 5.2 shows function approximation outcomes from three different algorithms: each algorithm constructed the function h based on the training samples $\{(z_i, y_i)\}$. The first plot illustrates typical results when we put too much trust in the data: we achieved $\mathcal{E}(h) = 0$, which should be good news. However, it is unlikely that the true function exhibits so many peaks and valleys – this behavior is most likely the product of a bad algorithm. The wiggly behavior is called over-fitting. A good algorithm produces the smooth approximation shown in the middle. This is achieved using a regularizer. With too much regularization, you obtain a poor approximation, as shown on the right.

The preference for the middle plot in Fig. 5.2 is based on a "*smoothness prior*" for the underlying data: a substitute for the probabilistic priors used in Bayesian statistics.

Mean-square Bellman error Our goal is to estimate the optimal Q-function $Q^\star(x, u)$ defined in (3.6): we take $H = Q^\star$ and $\mathsf{Z} = \mathsf{X} \times \mathsf{U}$. Our *second glance ahead* in Section 3.7 provided a roadmap, inspired by the Bellman error equation (3.9) for the Q-function. For any function $Q\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$, and any input-state sequence $(\boldsymbol{u}, \boldsymbol{x})$, the temporal difference is defined in (3.45), and recalled here:

$$\mathcal{D}_{k+1}(Q) := -Q(x(k), u(k)) + c(x(k), u(k)) + \underline{Q}(x(k+1)) \tag{5.4}$$

Given a time horizon $K \geq 1$, and the input-state sequence $\{u(k), x(k) : 0 \leq k \leq K\}$, we must take $N = K+1$ and observations $z_i = (x(i-1), u(i-1))$ to match the notation (5.2), and from this define the loss function

$$\mathcal{E}(h) = \frac{1}{N}\sum_{i=1}^{N} \big[D_i(h(z_i), h(z_{i+1}))\big]^2 \tag{5.5a}$$

$$D_i(h(z_i), h(z_{i+1})) = -h(x(i-1), u(i-1)) + c(x(i-1), u(i-1)) + \underline{h}(x(i)) \tag{5.5b}$$

with $\underline{h}(x) = \min_u h(x,u)$ for any function h. The complex looking term (5.5a) is the temporal difference, $D_i(h(z_i), h(z_{i+1})) = \mathcal{D}_i(h)$, as defined in (5.4).

Empirical distributions In the RL literature you will find the term *experience replay buffer* in reference to training data, and from this the *empirical distribution* (or *empirical pmf*) generated from this data:

$$\Gamma^N(x,u,x^+) = \frac{1}{N}\sum_{k=0}^{N-1} \mathbf{1}\{x(k) = x\,,\ u(k) = u\,,\ x(k+1) = x^+\}\,, \qquad x, x^+ \in \mathsf{X} \tag{5.6}$$

This is a pmf on $\in \mathsf{X} \times \mathsf{U} \times \mathsf{X}$ for any sequence of values $\{x(k), u(k)\}$ and any $N \geq 1$. Simple accounting leads to the following alternate expression for (5.5):

$$\mathcal{E}(h) = \sum_{x,u,x^+} \Gamma^N(x,u,x^+)\{-h(x,u) + c(x,u) + \underline{h}(x^+)\}^2 \tag{5.7}$$

where the sum is over all $(x,,ux^+) \in \mathsf{X} \times \mathsf{U} \times \mathsf{X}$ for which $\Gamma^N(x,x^+) > 0$. This interpretation of $\mathcal{E}(h)$ as an *empirical mean* is useful for both intuition and theory to come (such as the LP approach to MDPs that is surveyed in Section 7.2.3).

A bit more terminology: $\mathcal{E}$ is also known as the *empirical risk*, and its minimization over a function class $\mathcal{H}$ is known as *empirical risk minimization*.

5.1.2 Linear function approximation

This refers to a family of functions, linearly parameterized by $\theta \in \mathbb{R}^d$:

$$H^\theta(x) = \sum_{i=1}^{d} \theta_i \psi_i(x)\,, \qquad x \in \mathsf{Z} \tag{5.8}$$

where $\{\psi_i\}$ are the basis functions. It is convenient to stack these together to form a function $\psi\colon \mathsf{Z} \to \mathbb{R}^d$, and then write $H^\theta = \theta^\intercal \psi$. For any smooth loss function, the first-order condition for optimality is $0 = \nabla_\theta \mathcal{E}(H^\theta)$. For the mean-square Bellman error (5.5a) this becomes

$$\begin{aligned} 0 &= \sum_{i=1}^{N} D_i(H^\theta(z_i), H^\theta(z_{i+1}))\zeta_i^\theta \\ &\text{where} \quad \zeta_i^\theta = \nabla_\theta D_i(H^\theta(z_i), H^\theta(z_{i+1})) \end{aligned} \tag{5.9}$$

The choice of basis can be informed by some understanding of the control problem. For example, if $\mathsf{Z} = \mathbb{R}^2$ and it is known that H is convex, then it may be sufficient to choose H^θ quadratic, with $d = 7$:

$$\psi_1(x) = x_1\,,\ \psi_2(x) = x_2\,,\ \psi_3(x) = x_3\,,$$
$$\psi_4(x) = x_1^2\,,\ \psi_5(x) = x_1x_2\,,\ \psi_6(x) = x_2^2\,,$$

and $\psi_7(x) = 1$ for all $x \in \mathbb{R}^2$.

Galerkin relaxation The term *Galerkin relaxation* appears throughout the book as a means to approximate equality constraints such as (5.9), and sometimes also inequality constraints. As an example of this technique, consider again the loss function (5.5) associated with the mean-square Bellman error. An alternative approximation of the Bellman equation is obtained by constructing a d_ζ-dimensional sequence $\{\zeta_k\}$, and search for a function h that satisfies the constraint:

$$0 = \frac{1}{N}\sum_{i=1}^{N} D_i(h(z_i), h(z_{i+1}))\zeta_i\,, \qquad 1 \le i \le d_\zeta \tag{5.10}$$

This is a Galerkin relaxation, and certainly a relaxation of our ultimate if unrealistic goal: to find a function h for which the temporal difference $D_i(h(z_i), h(z_{i+1}))$ is zero for each i. In the context of RL, the vectors $\{\zeta_k\}$ appear as *eligibility vectors* in standard algorithms (see Section 5.4).

For a finite-dimensional function class we take $d_\zeta = d$, so that (5.10) represents d constraints, which is consistent with the d unknowns, $\{\theta_i^* : 1 \le i \le d\}$.

Equation (5.10) appears similar to (5.9). However, $\zeta_i = \zeta_i^\theta$ is not a valid choice, since the Galerkin relaxation does not allow ζ_i to depend on the θ. In practice we might design $\{\zeta_k\}$ so that $\zeta_i \approx \zeta_i^\theta$ for θ in a region of interest.

We are not always so fortunate to have intuition regarding the shape of H, which is why there has been so much attention focused on "black box" function approximation architectures.

5.1.3 Neural networks

Neural networks can be used to define a parameterized family of approximations $\{H^\theta\}$ that are highly nonlinear in θ. The purpose of this very brief introduction is to explain how a neural network can be used for function approximation, and especially for applications to value function approximation.

Fig. 5.3 shows an example of a feed-forward neural network with a single input layer, a single output layer, and three *hidden layers* (the optional *bias* terms are not included). For our purposes, this figure represents a function approximation $H\colon \mathbb{R}^3 \to \mathbb{R}$, so that the input layer is $x = (x_1, x_2, x_3)^\intercal$.

This is called a feed-forward network because calculation of y as a function of x is performed sequentially, moving from left to right. For given weight vectors $\{w_k^j\}$ (whose dimensions will be clear from the definitions), the calculations proceed as follows:

The first step is to calculate values $h^1 \in \mathbb{R}^4$ in hidden layer one, via

$$h_k^1 = \sigma(\langle w_k^1, x\rangle)\,, \qquad 1 \le k \le 4$$

where the notation $\langle w_k^1, x\rangle$ represents the usual dot product of two vectors, and $\sigma\colon \mathbb{R} \to \mathbb{R}_+$ is known as the *activation function*. Two standard choices:

$$\textit{Sigmoid:}\quad \sigma(r) = 1/(1+e^{-r}) \qquad\qquad \textit{ReLu:}\quad \sigma(r) = \max(0, r)$$

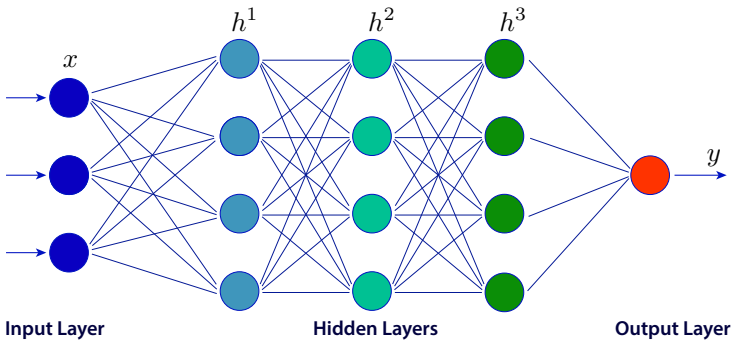


Figure 5.3: Neural network with three hidden layers.

Calculation of $h^2, h^3 \in \mathbb{R}^4$ is similar:

$$h_k^2 = \sigma(\langle w_k^2, h^1 \rangle)\,, \quad h_k^3 = \sigma(\langle w_k^3, h^2 \rangle)\,, \qquad 1 \le k \le 4$$

The output is then defined by $y = \langle w_k^4, h^3 \rangle$, which is a linear function of the third hidden layer, but a complex nonlinear function of the input x. The weights are identified with the parameter θ: we may write $y = H^\theta(x)$, with

$$\{\theta_i : 1 \le i \le d\} = \{w_k^j\}\,, \qquad d = 3 \times 4 + 4 \times 4 + 4 \times 4 + 4 = 48$$

5.1.4 Kernels

Let's start at the conclusion: when applying kernel methods, our approximation of H takes the form

$$H^\theta(x) = \sum_{i=1}^{N} \theta_i k(x, z_i)\,, \qquad x \in \mathsf{Z}\,, \tag{5.11}$$

where k is the kernel function that we choose from a large library.

You might argue that this is simply the linear function approximation approach described earlier, with $d = n$ and $\psi_i(x) = k(x, z_i)$ for each i and x. Your argument is absolutely correct! To appreciate the kernel method, you need to see how we arrive at this particular form for H^θ.

We return to the "beginning", which is the choice of kernel.

Choice of kernel, and requirements Three standard examples are

$$\begin{aligned}
&\text{Gaussian:} && k(x,y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}} \\
&\text{Laplacian:} && k(x,y) = e^{\frac{-\|x-y\|}{\sigma}} \\
&\text{Polynomial:} && k(x,y) = (r\langle x, y\rangle + 1)^m\,, \qquad x, y \in \mathsf{Z}
\end{aligned}$$

where $\sigma > 0$, $r > 0$ and $m \ge 1$ are design parameters.

Recall that in some control applications we may know that H is convex and non-negative. In this case, the polynomial kernel is attractive because H^θ in (5.11) is convex if m is even, and $\{\theta_i\}$ are non-negative.

Each of these three examples has the symmetry property, $k(x,y) = k(y,x)$. This is one of the several required properties of a kernel. A crucial requirement is that it is *positive definite*: for every $n \geq 1$, every collection $\{z_i : 1 \leq i \leq n\} \subset \mathsf{Z}$, and every $\alpha \in \mathbb{R}^n$,

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j k(z_i, x_j) \geq 0 \tag{5.12}$$

with equality if and only if $\alpha = 0$.

Function class for approximation Once we have selected a kernel, we arrive at a function class $\mathcal{H}$: an infinite-dimensional analog of the set of functions $\{H^\theta : \theta \in \mathbb{R}^d\}$ defined in (5.8). The collection $\mathcal{H}$ contains every function of the form,

$$h_\alpha(x) = \sum_{i=1}^{n} \alpha_i k(x, z_i)\,, \qquad x \in \mathsf{Z}$$

for any integer n, scalars $\{\alpha_i\}$, and $\{z_i\} \subset \mathsf{Z}$. The primitive functions h_α are also *dense* in $\mathcal{H}$. That is, if $h \in \mathcal{H}$, then for each $\varepsilon > 0$, there is an integer n, scalars $\{\alpha_i\}$, and $\{z_i\} \subset \mathsf{Z}$ (all depending on ε), with $\|h - h_\alpha\|_{\mathcal{H}} \leq \varepsilon$. The choice of norm $\|\cdot\|_{\mathcal{H}}$ is a critical part of the theory.

For the primitive functions h_α and h_β (with m and $\{z_i\}$ arbitrary), an inner product is introduced that defines the norm:

$$\langle h_\alpha, h_\beta \rangle_{\mathcal{H}} = \sum_{i,j=1}^{n} \alpha_i \beta_j k(z_i, x_j) \tag{5.13a}$$

$$\|h_\alpha\|_{\mathcal{H}} = \sqrt{\langle h_\alpha, h_\alpha \rangle_{\mathcal{H}}} \tag{5.13b}$$

The positivity assumption (5.12) ensures that $\langle h_\alpha, h_\alpha \rangle_{\mathcal{H}}$ is non-negative. The definition of the inner product and norm can be extended to the larger collection of functions $\mathcal{H}$, and endowed with this inner product it is known as a *reproducing kernel Hilbert space* (*RKHS*).

For our purposes, details regarding $\mathcal{H}$ are not required because the *Representer Theorem* tells us we can restrict to the primitive functions in the function approximation problems of interest to us. To present this theorem requires one more ingredient.

Regularized loss function In addition to the loss function $\mathcal{E}$ we require a *regularizer*, defined here as an increasing function $G\colon \mathbb{R}_+ \to \mathbb{R}_+$. A typical choice is $G(r) = \lambda r^2$ or $G(r) = \lambda r$, with $\lambda > 0$. The regularizer is introduced to manage the over-fitting problem illustrated in Fig. 5.2.

Our interest is solving the optimization problem

$$h^* = \arg\min\{\mathcal{E}(h) + G(\|h\|_{\mathcal{H}}) : h \in \mathcal{H}\} \tag{5.14}$$

Theorem 5.1. (Representer Theorem) *Consider a positive-definite real-valued kernel $k : \mathsf{Z} \times \mathsf{Z} \to \mathbb{R}$. Suppose that $\{z_i : 1 \leq i \leq N\}$ are given, along with a loss function of the form (5.2) and regularizer G. Then, any minimizer of the optimization problem (5.14) can be expressed*

$$h^*(\cdot) = \sum_{i=1}^{N} \alpha_i^* k(\cdot, z_i), \tag{5.15}$$

for some $\alpha^* \in \mathbb{R}^N$. ∎

We now return to the two examples:

Curve fitting Consider the quadratic loss (5.3). If the regularizer is also quadratic, $G(r) = \lambda r^2$, then the Representer Theorem provides an explicit solution to (5.14). We are left to obtain the optimal parameter:

$$\alpha^* = \arg\min_{\alpha} \Big\{ \sum_{i=1}^{N} [y_i - h_\alpha(z_i)]^2 + \lambda \|h_\alpha\|_{\mathcal{H}}^2 \Big\}$$

Let K denote the $n \times n$ matrix with entries $K_{i,j} = k(z_i, x_j)$. We then have $\|h_\alpha\|_{\mathcal{H}}^2 = \alpha^\intercal K \alpha$, and $h_\alpha(z_i) = \sum_j \alpha_j k(z_i, x_j) = [K\alpha]_i$. To compute α^* we set the partial derivatives of the loss equal to zero:

$$0 = \frac{\partial}{\partial \alpha_j} \Big\{ \sum_{i=1}^{N} [y_i - [K\alpha]_i]^2 + \lambda \alpha^\intercal K \alpha \Big\} = 2 \sum_{i=1}^{N} [y_i - [K\alpha]_i] K_{i,j} + 2\lambda [K\alpha]_j$$

With $y, \alpha^* \in \mathbb{R}^N$ column vectors, this gives[7]

$$\alpha^* = (K^\intercal K + \lambda K)^{-1} K^\intercal y \tag{5.16}$$

Mean-square Bellman error We no longer have an explicit solution to (5.14), even with G quadratic, but we know that h^* is of the form (5.15) for some vector α^*. A more successful approach may proceed using a convex loss function $\mathcal{E} \colon \mathbb{R}^N \to \mathbb{R}_+$, constructed by applying the representations in Section 3.5.

5.1.5 Are We Done Yet?

This is mainly a guide to how to receive a passing grade on your experimental oriented homework.

Let's think about how to answer the question within the context of minimizing the mean-square Bellman error using linear function approximation, which results in the root finding problem (5.9): $\overline{f}_N(\theta_N^*) = 0$, with

$$\overline{f}_N(\theta) = \sum_{i=1}^{N} D_i(H^\theta(z_i), H^\theta(z_{i+1}))\zeta_i^\theta$$

While it may take you a long time to compute θ_N^*, you are far from done.

Some experiments you can perform to obtain more confidence that you have a useful solution:

Is your parameterization redundant? Consider $Q^\theta = \theta^\intercal \psi$, an approximate Q-function, and estimate the covariance Σ_ψ using samples:

$$\widehat{\Sigma}_\psi = \frac{1}{N} \sum_{i=1}^{N} \psi(z_i)\psi(z_i)^\intercal$$

[7] We have assumed that $K = K^\intercal$: the transpose is included anyway, since this is a standard approximation of the inverse of a matrix K

Look at the eigenvalues of this positive semidefinite matrix—if there is a null space, then you are in trouble. That is, if $\widehat{\Sigma}_\psi v = 0$ for some non-zero vector v, then obviously $v^\intercal \widehat{\Sigma}_\psi v = 0$, meaning that

$$0 = v^\intercal \widehat{\Sigma}_\psi v = \frac{1}{N} \sum_{i=1}^{N} \big(v^\intercal \psi(z_i)\big)^2$$

It follows that Q^θ, with $\theta = v$, is identically zero on the samples observed. And it means your basis is highly redundant, in the sense that one ψ_k is a linear combination of the others: if $v_k \neq 0$, then for each i,

$$\psi_k(z_i) = -\frac{1}{v_k} \sum_{j \neq k} v_j \psi_j(z_i)$$

There are two potential explanations: either your basis is truly linearly dependent in an algebraic sense:

$$v^\intercal \psi(z) = 0\,, \qquad z \in \mathsf{Z}$$

A second potential explanation is insufficient exploration, so that your samples $\boldsymbol{z}$ evolve in a low-dimensional subset of Z.

Is your parameter predictive using fresh data? Obtain $M \gg 1$ more batches of data $\{\boldsymbol{z}^m : 1 \leq m \leq M\}$, and compute $\overline{f}_N^m(\theta^*_N)$ for each m, with

$$\overline{f}_N^m(\theta) = \sum_{i=1}^{N} D_i(H^\theta(z_i^m), H^\theta(z_{i+1}^m))\zeta_i^\theta\,, \qquad 1 \leq m \leq M$$

If there is large variability in the M values, then you need to increase N.

Is the output of your algorithm predictive of what really matters? This will take some work, but it is truly essential. With $M \gg 1$ batches of data, estimate the performance you obtain with the output of your algorithm. This means that for each m you must

(i) Obtain an estimate θ^{*m} using your algorithm.

(ii) Obtain $\phi_m(x) \in \arg\min_u Q^{\theta^{*m}}(x, u)$

(iii) Run more experiments to estimate the performance. For the total cost problems considered here, choose a pmf μ with finite support. For each initial condition x^i satisfying $\mu(x^i) > 0$ run a simulation to estimate $J(x^i)$ under policy ϕ_m, and then obtain $L_m = \sum_i \mu(x^i)\widehat{J}_m(x^i)$. Look at the sample mean and variance of $\{L_m : 1 \leq m \leq M\}$. High variance means you need a longer run.

Or, you might decide to look more closely at those policies for which L_m is smallest—maybe you got lucky! To know for sure, you need a deeper investigation of performance, using data independent of what was used for training.

If we are talking about *real life*, rather than a homework problem, then you need advice from experts! For example, if your θ^* is supposed to define an optimal policy for an autonomous car, then you need experts in sociology as well as highway engineering to conduct realistic experiments.

5.2 Exploration and ODE Approximations

The success of the RL algorithms surveyed in this chapter depends in part on the choice of input $\boldsymbol{u}$ used for training. The purpose of this section is to make this precise, and present our main assumption on the input designed for generating data to train the algorithm (that is, *exploration*, as first surveyed in Section 2.5.3). Throughout this chapter it is assumed that the input used for training is state-feedback with perturbation, of the form

$$u(k) = \widetilde{\phi}(x(k), \xi(k)) \tag{5.17}$$

where $\boldsymbol{\xi}$ is a bounded sequence evolving on $\mathbb{R}^p$ for some $p \geq 1$. It plays the same role as the probing signal introduced for gradient-free optimization in Section 4.6, with applications to policy gradient algorithms in Section 4.7.

It is sometimes convenient to assume that the exploration itself evolves as an autonomous state space model

$$\xi(k+1) = \mathrm{H}(\xi(k)) \tag{5.18}$$

in which $\mathrm{H}\colon \mathbb{R}^p \to \mathbb{R}^p$ is continuous. Subject to the policy (5.17), it follows that the triple $\Phi(k) = (x(k), u(k), \xi(k))^\intercal$ has a similar recursive form, evolving on the larger state space $\mathsf{Z} = \mathsf{X} \times \mathsf{U} \times \mathbb{R}^p$. In some cases, such as in TD(λ) learning, it is necessary to add additional components to $\Phi(k)$, and extend the state space Z. This is the reason for the abstract description of $\boldsymbol{\Phi}$ in Assumption (A$\boldsymbol{\xi}$) below.

For any continuous function $g\colon \mathsf{Z} \to \mathbb{R}$ and $N \geq 1$, denote

$$\overline{g}_N = \frac{1}{N} \sum_{k=1}^{N} g(\Phi(k))$$

For any $\ell > 0$ denote

$$\mathcal{G}_\ell = \{g : \|g(z') - g(z)\| \leq \ell \|z - z'\|, \text{ for all } \ z, z' \in \mathsf{Z}\}$$

The "quasi-randomized" policy structure defined by eqs. (5.17) and (5.18) is imposed simplify analysis of algorithms, along with assumptions on a slight generalization of the larger process $\boldsymbol{\Phi}$:

(A$\boldsymbol{\xi}$) The state and action spaces X and U are each closed subsets of Euclidean space; F defined in (3.1), $\widetilde{\phi}$ defined in (5.17), and H in (5.18) are each continuous on their domains. There is a larger state process $\boldsymbol{\Phi}$ with the following properties:

(i) $\boldsymbol{\Phi}$ evolves on a closed subset of Euclidean space, denoted Z, and $(x(k), u(k), \xi(k)) = w(\Phi(k))$ for each k, where $w\colon \mathsf{Z} \to \mathbb{R}$ is continuous.

(ii) $\boldsymbol{\Phi}$: is ergodic in the following sense: There is a probability measure ϖ such that for any continuous function $g\colon \mathsf{Z} \to \mathbb{R}$, the following ergodic average exists for each initial condition $\Phi(0)$

$$\mathsf{E}_\varpi[g(\Phi)] := \lim_{N\to\infty} \overline{g}_N \tag{5.19}$$

(iii) The limit in (5.19) is uniform on $\mathcal{G}_\ell$, for each $\ell < \infty$:

$$\lim_{N\to\infty} \sup_{g\in\mathcal{G}_\ell} |\overline{g}_N - \mathsf{E}_\varpi[g(\Phi)]| = 0$$

Please remember that these assumptions are *not* essential for successful implementation of algorithms. They are introduced to simplify analysis.

ODE approximations Just as in the previous chapter, ergodicity allows for approximation of algorithms by simpler ODE approximations. In particular, consider a recursion of the form

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n)\,, \qquad n \ge 0 \tag{5.20}$$

in which $\{f_n\}$ is a sequence of functions that admits an ergodic limit:

$$\overline{f}(\theta) := \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} f_k(\theta)\,, \qquad \theta \in \mathbb{R}^d$$

The associated ODE is defined using this vector field:

$$\tfrac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t) \tag{5.21}$$

An ODE approximation is defined by mimicking the usual Euler construction: the time-scale for the ODE is defined by the non-decreasing time points $t_0 = 0$ and $t_n = \sum_0^n \alpha_k$ for $n \ge 1$. Define a continuous time process $\boldsymbol{\Theta}$ by $\Theta_{t_n} = \theta_n$ for each n, and extend to all t through piecewise linear interpolation. The next step is to fix a time horizon for analysis of length $\mathcal{T} > 0$, where the choice of $\mathcal{T}$ is determined based on properties of the ODE. Denote $\mathcal{T}_0 = 0$, and

$$\mathcal{T}_{n+1} = \min\{t_n : t_n - \mathcal{T}_n \ge \mathcal{T}\}\,, \qquad n \ge 0 \tag{5.22}$$

Let $\{\vartheta_t^n : t \ge \mathcal{T}_n\}$ denote the solution to the ODE (5.21) with initial condition $\vartheta^n_{\mathcal{T}_n} = \theta_{k(n)}$, with index defined so that $t_{k(n)} = \mathcal{T}_n$. We then say that the algorithm (5.20) admits an *ODE approximation* if for each initial θ_0,

$$\lim_{n\to\infty} \sup_{\mathcal{T}_n \le t \le \mathcal{T}_{n+1}} \|\Theta_t - \vartheta_t^n\| = 0 \tag{5.23}$$

5.3 TD-learning and SARSA

TD learning is defined as methods to approximate a value function for a fixed policy ϕ. This may be just one step in the PIA introduced in Section 3.2.2, which requires estimation of J^n to be used in the policy improvement step (3.15).

In our *second glance ahead* discussion, contained in Section 3.7, we learned that it is much better to estimate the fixed-policy Q-function. An example of application to approximate policy iteration is contained in Section 4.5.3. For the policy ϕ, we may let J^ϕ denote the associated value function, and then the fixed-policy Q-function is defined by

$$Q^\phi(x,u) = c(x,u) + J^\phi(\mathrm{F}(x,\phi(x)))$$

In this notation, he fixed point equation (3.44) becomes

$$Q^\phi(x,u) = c(x,u) + Q^\phi(x^+,u^+)\,, \qquad x^+ = \mathrm{F}(x,u)\,, \quad u^+ = \phi(x^+) \tag{5.24}$$

For any approximation Q, we can observe the error in this fixed point equation as another temporal difference: for any input-state sequence $(\boldsymbol{u}, \boldsymbol{x})$, denote

$$\mathcal{D}_{k+1}(Q) := -Q(x(k),u(k)) + c(x(k),u(k)) + \underline{Q}_\phi(x(k+1)) \tag{5.25}$$

with the fresh definition,

$$\underline{Q}_\phi(x) = Q(x, \phi(x)), \qquad x \in \mathsf{X} \tag{5.26}$$

The temporal difference (5.25) is zero for all k if we substitute Q^ϕ for Q.

Algorithms to approximate Q^ϕ based on the temporal difference sequence (5.25) are called SARSA. These algorithms are only a minor variation on the TD-learning algorithms designed to estimate J^ϕ, so we opt for the simpler terminology "TD-learning" throughout the book.

There are two distinct flavors of TD-learning: *on policy* and *off policy*. The on-policy versions choose $u(k) = \phi(x(k))$ in the definition (5.25). Theory for on-policy algorithms is elegant (with origins mainly in the Markovian setting, rather than the deterministic control problems considered in this part of the book). The difficulties with on-policy algorithms should be clear following the discussion in Section 2.5.3 regarding exploration: if ϕ is a good policy, in the sense that $x(k) \to x^e$, $u(k) = \phi(x(k)) \to u^e$ as $k \to \infty$, then for any function Q

$$\begin{aligned} \lim_{k\to\infty} \mathcal{D}_{k+1}(Q) &= \lim_{k\to\infty} \big\{ -Q(x(k), u(k)) + c(x(k), u(k)) + Q(x(k+1), \phi(x(k+1)) \big\} \\ &= c(x^e, u^e) = 0, \qquad \textit{if } u(k) = \phi(x(k)) \textit{ for each } k \end{aligned} \tag{5.27}$$

Recall that the convention $c(x^e, u^e) = 0$ is required so that J^ϕ is finite valued for some policy.

In this part of the book we focus mainly on off-policy algorithms. More depth on the elegant theory for on-policy algorithms is surveyed in Part 3.

5.3.1 TD-Learning and linear regression

Consider a parameterization introduced previously in (4.55):

$$Q^\theta(x, u) = d(x, u) + \theta^\intercal \psi(x, u), \quad \theta \in \mathbb{R}^d. \tag{5.28}$$

in which $d\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is regarded as an estimate of the cost function (or $d = c$ if the cost function is given). In strict mathematical terms, this is an *affine* function class, but most would continue to use the term *linear function class*.

Given the assumption that $Q(x^e, u^e) = c(x^e, u^e) = 0$, it is important to construct the function class with this in mind:

$$d(x^e, u^e) = 0, \quad \textit{and} \quad \psi_i(x^e, u^e) = 0 \quad 1 \le i \le d \tag{5.29}$$

From the definition (5.25) we obtain

$$\mathcal{D}_{k+1}(Q^\theta) = -Q^\theta(x(k), u(k)) + c(x(k), u(k)) + \underline{Q}^\theta_\phi(x(k+1))$$

This can be expressed in a form that will inspire any young statistician. On denoting

$$\gamma(k+1) = c(x(k), u(k)) - d(x(k), u(k)) + d(x(k+1), \phi(x(k+1))) \tag{5.30a}$$

$$\Upsilon(k+1) = \psi(x(k), u(k)) - \psi(x(k+1), \phi(x(k+1))) \tag{5.30b}$$

we obtain the representation

$$\gamma(k+1) = \Upsilon(k+1)^\intercal \theta + \mathcal{D}_{k+1}(Q^\theta) \tag{5.30c}$$

This is the form of a standard regression problem:

$$\gamma(k) = \Upsilon(k)^\intercal \theta^* + \varepsilon_k$$

where $\{\varepsilon_k = \mathcal{D}_k(Q^{\theta^*}) : k \ge 0\}$ is regarded as "noise", and θ^* is typically defined as the minimum variance parameter: $\theta^* = \arg\min_\theta L(\theta)$, with

$$L(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} [\gamma(k) - \Upsilon(k)^\intercal \theta]^2 \tag{5.31}$$

Convergence of this limit require conditions on the input, and further conditions are required so that this loss function is meaningful. In particular, for the on-policy approach in which (5.27) holds, $L(\theta) = 0$ for every θ!

This is where exploration comes in. Consider a "randomized" feedback law

$$u(k) = \widetilde{\phi}(x(k), \xi(k)) \tag{5.32}$$

in which $\xi(k)$ is a sampled exploratory signal of the form used in QSA. For example, it may be a vector-valued sequence, with components of the form (4.44) with varying frequencies:

$$\xi_i(k) = \xi_i(0) + 2\pi\omega_i k \pmod 1$$

It is not unreasonable to expect convergence of the limit (5.31) in this case. For the limit L to define a meaningful loss function requires several considerations: the frequencies must be selected with care, and the dimension of $\boldsymbol{\xi}$ should be at least as large as d (the dimension of θ). Justification for these restrictions is possible in the case of LQR—see Exercise 5.2.

Least Squares Temporal Difference Learning

For a given $d \times d$ matrix $W > 0$, integer N, and observed samples $\{u(k), x(k) : 0 \le k \le N\}$, the minimizer is obtained:

$$\theta_N^* = \arg\min_\theta L_N(\theta)\,, \qquad L_N(\theta) = \theta^\intercal W \theta + \sum_{k=0}^{N-1} [\gamma(k) - \Upsilon(k)^\intercal \theta]^2 \tag{5.33}$$

This defines the approximation of the Q-function $Q^{\theta_N^*} = d + \sum_i \theta_N^*(i)\psi_i$.

The objective is a positive definite quadratic, so the solution to (5.33) is obtained on setting ∇L_N to zero:

Proposition 5.2. *$\theta_N^* = [N^{-1}W + \widehat{A}_N]^{-1}\widehat{b}_N$, with*

$$\widehat{A}_N = \frac{1}{N} \sum_{k=0}^{N-1} \Upsilon(k)\Upsilon(k)^\intercal\,, \quad \widehat{b}_N = \frac{1}{N} \sum_{k=0}^{N-1} \Upsilon(k)\gamma(k)$$

□

On uniqueness of θ^* The regularizer $\theta^\intercal W\theta$ is required to get a unique solution, since there is little theory available to tell us if $\widehat{A}_N$ is invertible for any N, or if the loss function L defined in (5.31) has a unique minimizer θ^*.

It is worth investigating the implications if $\widehat{A}_N$ is not invertible. It then follows that there is a non-zero vector v satisfying $v^\intercal \widehat{A}_N v = 0$, which means

$$0 = v^\intercal \widehat{A}_N v = \frac{1}{N}\sum_{k=0}^{N-1} \big(v^\intercal \Upsilon(k)\big)^2$$

That is, $v^\intercal \Upsilon(k) = 0$ for every observed sample, which means

$$v^\intercal \psi(x(k), u(k)) = v^\intercal \psi(x(k+1), \phi(x(k+1)))\,, \qquad 0 \le k \le N-1 \tag{5.34}$$

One appeal of on-policy methods is that the question of invertibility has a simple answer. If $u(k) = \phi(x(k))$ for all k then,

$$\Upsilon(k+1) = \psi(x(k), u(k)) - \psi(x(k+1), u(x(k+1)))$$

That is, from the representation of $\Upsilon(k)$ in this special case,

$$v^\intercal \psi(x(k), u(k)) = v^\intercal \psi(x(k+1), u(x(k+1)))\,, \qquad 0 \le k \le N-1$$

This does not seem likely for any reasonable basis. If it holds true for *every* k, and if the policy is stable, then

$$v^\intercal \psi(x(0), u(0)) = v^\intercal \psi(x(k), u(k)) = \lim_{j\to\infty} v^\intercal \psi(x(j), u(x(j))) = v^\intercal \psi(x^e, u^e) = 0$$

where the final equation uses (5.29). Hence your basis falls into the "redundant" category discussed in Section 5.1.5. These conclusions are summarized in the following:

Proposition 5.3. *Suppose that $\widehat{A}_N$ has rank less than d. Then, there is a non-zero parameter θ^n for which the following hold, for each $0 \le k \le N-1$:*

(i) $Q^{\theta^n}(x(k), u(k)) - d(x(k), u(k)) = \underline{Q}_\phi^{\theta^n}(x(k+1)) - d(x(k+1), \phi(x(k+1)))$

(ii) *For the on-policy implementation,*

$$\theta^{n\intercal} \psi(x(0), u(0)) = \theta^{n\intercal} \psi(x(k), u(k))$$

Proof. In both parts, the vector θ^n is any non-zero vector v in the null space of $\widehat{A}_N$. Part (ii) was stated before the proposition, and part (i) is a re-interpretation of (5.34). ☐

Given the desirable properties of the on-policy setting, in some examples it may be best to go with the re-start option (2.51):

Least Squares Temporal Difference Learning (on-policy, with re-start)

For a given $d \times d$ matrix $W > 0$, integers N and M, and observed samples

$$\{u^i(k), x^i(k) : 0 \le i \le N\,, 1 \le i \le M\}\,,$$

with user-defined initial conditions $\{x^i(0) : 1 \le i \le M\}$, and with $u^i(k) = \phi(x^i(k))$ (on-policy).

The approximation of the Q-function $Q^{\theta^*_N} = d + \psi^\intercal \theta^*_N$ is obtained, in which the optimal parameter is defined by the following steps:

(i) Introduce a per-batch loss function $L^i_N(\theta)$: defined by (5.33) using the ith batch, $B^i = \{u^i(k), x^i(k) : 0 \le k \le N\}$.

(ii) Define $\theta^*_N = \arg\min_\theta L_N(\theta)$, with

$$L_N(\theta) = \frac{1}{M} \sum_{i=1}^{M} L^i_N(\theta) \tag{5.35}$$

What do you do if $d = 10^6$? There is not yet consensus. Practitioners in machine learning often face high dimensional optimization problems, and claim that dimensions of one million are no longer a concern. This success story is attributed to advances in optimization theory, computer engineering, and computing power. In the RL research community it is typical to modify the objective in order to reduce computational complexity.

5.3.2 ODE analysis and Zap

5.4 Projected Dynamic Programming and TD Algorithms

Recall that in motivating the value iteration algorithm we began with the interpretation of the Bellman equation (3.2) as a fixed point equation in the "variable" J. VIA is simply the successive approximation technique for solving such fixed point equations. This section concerns approximation of such fixed point equations, and how they motivate the oldest approaches to RL.

The style of this section is very different from the previous sections in this chapter: it contains a survey of popular algorithms, often "translated" to the deterministic control setting of this part of the book. Stochastic analogs will be defined in later chapters. These algorithms are successfully applied in practice, but unfortunately there is not much theory to explain success or failure. So, consider this section more of a cookbook, and less of a guide to understanding.

The motivation behind these algorithms requires a bit more background on function approximation, described here using the notation of Section 5.1. We begin with an abstraction: find a function h^* that solves a fixed point equation:

$$h^* = T(h^*) \tag{5.36}$$

The specifics of the domain and range of h^*, and the meaning of the mapping T, depends on the problem we would like to solve. The Bellman equation (3.5) is one example, with $h^* = J^\star$. Solving (5.36) is intractable, so we seek an approximation.

We choose a function class $\mathcal{H}$, which in this section is defined by a basis (though this is not essential). A projection operator $P_{\mathcal{H}}$ is constructed, which takes any function h to $P_{\mathcal{H}}(h) \in \mathcal{H}$, and we then turn to the approximation of (5.36):

$$\hat{h} = \widehat{T}(\hat{h}) := P_{\mathcal{H}}\{T(\hat{h})\} \tag{5.37}$$

The solution $\hat{h}$ must lie in $\mathcal{H}$ because of the constraints we impose on $P_{\mathcal{H}}$.

In some cases this approach fails or is too complex, so we consider the alternative: given a second function class $\mathcal{G}$, find a function $\hat{h} \in \mathcal{H}$ solving

$$0 = P_{\mathcal{G}}\{\hat{h} - T(\hat{h})\} \tag{5.38}$$

This is a generalization of (5.37):

Proposition 5.4. *If $\mathcal{H} = \mathcal{G}$ then the solutions to (5.37) and (5.38) coincide.* □

When we put these ideas to practice in control, $\widehat{T}$ will define the *projected Bellman operator*, and (5.38) the *projected Bellman equation*.

5.4.1 Galerkin relaxations and projection

We begin with a bit more detail about the meaning of the projection $P_{\mathcal{G}}$, with each $g \in \mathcal{G}$ a function $g\colon \mathsf{Z} \to \mathbb{R}$, with Z the larger state space used in Assumption (A$\boldsymbol{\xi}$).

It is assumed at the start that the function class is linear: we choose d functions $\{\gamma_i : 1 \le i \le d\}$, stack these together to define a function $\gamma\colon \mathsf{Z} \to \mathbb{R}^d$, and then define the finite-dimensional linear function class $\mathcal{G} = \{g = \theta^{\intercal}\gamma : \theta \in \mathbb{R}^d\}$. We will denote $\zeta_k = \gamma(\Phi(k))$, and call this the sequence of *eligibility vectors*, since they will play a role in a Galerkin relaxation (first introduced in (5.10)).

The expectation introduced in (A$\boldsymbol{\xi}$) is used to define an inner product and norm on functions $h_1, h_2\colon \mathsf{Z} \to \mathbb{R}$:

$$\langle h_1, h_2\rangle_{\varpi} = \mathsf{E}_{\varpi}[h_1(\Phi)h_2(\Phi)]\,, \qquad \|h_1\|_{\varpi} = \sqrt{\mathsf{E}_{\varpi}[(h_1(\Phi))^2]} = \sqrt{\langle h_1, h_1\rangle_{\varpi}}$$

The function class $L_2(\varpi)$ is defined to be all functions h for which $\|h\|_{\varpi}$ is finite. With this background, we can define the projection: for any $h \in L_2(\varpi)$, the projection $\hat{h} \in \mathcal{G}$ is

$$\hat{h} := \arg\min_{g} \{\|g - h\|_{\varpi} : g \in \mathcal{G}\}$$

Proposition 5.5. *Suppose that $\gamma_i \in L_2(\varpi)$ for each i, and that these functions are* linearly independent *in $L_2(\varpi)$. That is, $\|\theta^{\intercal}\gamma\|_{\varpi} = 0$ implies that $\theta = 0$.*

For each $h \in L_2(\varpi)$ the projection exists, is unique, and given by $\hat{h} = \theta^{\intercal}\gamma$ with

$$\theta = \Sigma_{\gamma}^{-1} b_h \tag{5.39}$$

where $b_h \in \mathbb{R}^d$ and the $d \times d$ matrix Σ_{γ} are defined by

$$\begin{aligned} b_h(i) &= \langle \gamma_i, h\rangle_{\varpi} \\ \Sigma_{\gamma}(i,j) &= \langle \gamma_i, \gamma_j\rangle_{\varpi}\,, 1 \le i, j \le d \end{aligned} \tag{5.40}$$

The proof is via the orthogonality principle:

$$\langle h - \hat{h}, \gamma_i \rangle_\varpi = 0\,, \qquad 1 \le i \le d$$

and the fact that necessarily $\hat{h} = \theta^\intercal \gamma$ for some θ since $\hat{h} \in \mathcal{G}$ by assumption.

Prop. 5.5 is the motivation for Galerkin approaches to root finding. Regardless of the meaning of the operator T appearing in (5.37), this equation holds if and only if $b_e = 0$ with $e = \hat{h} - T(\hat{h})$:

$$0 = \langle \gamma_i, \hat{h} - T(\hat{h}) \rangle_\varpi\,, \qquad 1 \le i \le d \tag{5.41}$$

This is by definition a Galerkin relaxation of (5.36).

5.4.2 TD(λ)-learning

The fixed-point equation (5.24) is precisely of the form (5.36): define for any function $h\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}$,

$$T(h)\Big|_{(x,u)} = c(x,u) + h(x^+, u^+)\,, \qquad x^+ = \mathrm{F}(x,u)\,, \quad u^+ = \phi(x^+)$$

so that $Q^\phi = T(Q^\phi)$.

Galerkin relaxations lead to the oldest and most celebrated RL algorithms. Consider specification of $\mathcal{H}$ as a finite dimensional function class $\{h = \theta^\intercal \psi : \theta \in \mathbb{R}^d\}$, where $\psi_i\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}$ for each i. We arrive at the projected Bellman equation by applying the approximation (5.37) to this problem, in its equivalent form (5.41): for each i,

$$0 = \mathsf{E}_\varpi\big[\zeta_k(i)\{\hat{h}(x(k),u(k)) - [c(x(k),u(k)) + \hat{h}(x(k+1),\phi(x(k+1)))]\}\big]$$

where we have used the definition of the inner product, along with the notation $\zeta_k = \gamma(\Phi_k)$. The solution of this root finding problem defines $Q^{\theta^*} = \hat{h}$, since $\hat{h} \in \mathcal{H}$.

Recalling the definition (5.27), the projected Bellman equation is equivalently expressed

$$0 = \mathsf{E}_\varpi\big[\zeta_k \mathcal{D}_{k+1}(Q^\theta)\big]\Big|_{\theta=\theta^*} \tag{5.42}$$

Given N observations, an approximation is obtained via

$$0 = \frac{1}{N}\sum_{k=0}^{N-1} \zeta_k \mathcal{D}_{k+1}(Q^\theta)\Big|_{\theta=\theta^*} \tag{5.43}$$

which is precisely of the form (5.10).

From this we can now define the meaning of "λ" in TD(λ) learning, which depends entirely on the choice of $\mathcal{G}$. The special case TD(0) is simple: $\mathcal{G} = \mathcal{H}$, so that we substitute $\zeta_k = \psi(x(k),u(k))$ in (5.42). For general $\lambda \in (0,1)$ we require the following abstraction:

$$\zeta_k = \sum_{j=0}^{\infty} \lambda^j \psi(x(k-j), u(k-j))$$

Of course, this is not physical: the control system has not been running for eternity! However, this is the only way to describe what the TD(λ) algorithm is attempting to compute. The state process $\mathbf{\Phi}$ can be defined on the two-sided interval, so this abstract definition is valid. Finally,

we can assume that ζ_k is a linear function of $\Phi(k)$, by extending the definition of this state process.

To obtain a practical algorithm we use the fact that this eligibility vector sequence evolves according to the first-order linear system equation,

$$\zeta_k = \lambda\zeta_{k-1} + \psi(x(k), u(k)), \qquad k \in \mathbb{Z}$$

Denote $\overline{f}_\lambda(\theta) = \mathsf{E}_\varpi\big[\zeta_k \mathcal{D}_{k+1}(Q^\theta)\big]$, and consider the ODE

$$\tfrac{d}{dt}\vartheta = \overline{f}_\lambda(\vartheta)$$

In the on-policy setting, subject to linear independence of ψ in $L_2(\varpi)$, it can be shown that this ODE is globally asymptotically stable. This motivates the celebrated algorithm:

TD(λ) Learning

For a given $\lambda \in [0,1]$, non-negative stepsize sequence $\{\alpha_n\}$, initial conditions θ_0, ζ_0, and observed samples $\{u(k), (x(k) : 0 \le k \le N\}$, the sequence of estimates are defined by the coupled equations:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \tag{5.44a}$$

$$\zeta_{n+1} = \lambda\zeta_n + \psi(x(n+1), u(n+1)) \tag{5.44b}$$

with $\mathcal{D}_{n+1} = \mathcal{D}_{k+1}(Q^\theta)\big|_{\theta=\theta_n}$.
This defines the approximation of the Q-function $Q^{\theta^*_N} = \sum_i \theta^*_N(i)\psi_i$.

The ODE introduced to motivate the algorithm is linear, $\overline{f}_\lambda(\theta) = A(\theta - \theta^*)$, in which

$$A = \mathsf{E}_\varpi\big[\zeta_k[-\psi(x(k), u(k) + \psi(x(k+1), \phi(x(k+1))]\big]$$

For on-policy implementation we have $u(k+1) = \phi(x(k+1))$ which simplifies analysis, but then poses a challenge because we are not likely to obtain the required linear independence of the basis functions.

Stability of TD(λ) learning is not guaranteed in the off-policy setting, even with $\lambda = 0$, Ideas found in the proof of Prop. 5.3 can be used to obtain conditions under which A is Hurwitz for the "near" on-policy setting, in which $\boldsymbol{u}$ is defined using a small perturbation of the policy ϕ.

In the remainder of this section we change course, turning to the Q-function associated with the optimal control problem: $Q^\star$, defined in (3.6).

5.4.3 Projected Bellman operator and Q-learning

The Q-function for the total cost optimal control problem solves the fixed point equation (3.9), copied here for convenience:

$$Q^\star(x,u) = c(x,u) + \underline{Q}^\star(\mathrm{F}(x,u))$$

with $\underline{Q}(x) = \min_u Q(x,u)$ for any Q. For a parameterized family of approximations $\{Q^\theta : \theta \in \mathbb{R}^d\}$, recall that for each θ we define a policy via (3.43):

$$\phi^\theta(x) \in \arg\min_u Q^\theta(x,u), \qquad x \in \mathsf{X}$$

We obtain an algorithm for approximation via "pattern matching" with (5.44):

Q(λ) Learning

For a given $\lambda \in [0,1]$, non-negative stepsize sequence $\{\alpha_n\}$, initial conditions θ_0, ζ_0, and observed samples $\{u(k), (x(k) : 0 \le k \le N\}$, the sequence of estimates are defined by the coupled equations:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \tag{5.45a}$$
$$\zeta_{n+1} = \lambda\zeta_n + \psi(x(n+1), u(n+1)) \tag{5.45b}$$
$$\mathcal{D}_{n+1} = c(x(n), u(n)) + Q^{\theta_n}(x(n+1), \phi^{\theta_n}(x(n+1))) \tag{5.45c}$$

The major change is (5.45c), where we use the current policy estimate ϕ^{θ_n} rather than the fixed policy ϕ in TD(λ). Note that in (5.45c) we can substitute

$$Q^{\theta_n}(x(n+1), \phi^{\theta_n}(x(n+1))) := \underline{Q}^{\theta_n}(x(n+1))$$

And ODE theory predicts that a limit θ^* for Q(λ)-learning will solve $\overline{f}(\theta^*) = 0$, with

$$\overline{f}(\theta) = \mathsf{E}\big[\mathcal{D}_{k+1}(\theta)\zeta_k\big] \tag{5.46}$$

The Q(0)-learning algorithm and variants are commonly applied because they are the natural extension of Watkins' Q-learning algorithm for controlled Markov chains. With $\lambda = 0$ we can apply Prop. 5.4 to conclude that Q^{θ^*} solves the projected Bellman equation

$$Q^{\theta^*} = P_{\mathcal{H}}\{T(Q^{\theta^*})\}$$

in which the Bellman operator is redefined:

$$T(h)\Big|_{(x,u)} = c(x,u) + \min_u h(x^+, u)\,, \qquad x^+ = \mathrm{F}(x,u)$$

There is a firm theory for convergence of Q(0)-learning in the special case considered by Watkins, in part because it is assumed that the function class is "complete" (contains every possible function on $\mathsf{X} \times \mathsf{U}$). Convergence is established using ODE methods in Chapter 11.

We can attempt to apply the same reasoning here: an ODE analysis of Q(λ)-learning requires that we look at global asymptotic stability of the ODE with vector field $\overline{f}$ defined in (5.46). As a first step, we must find conditions under which the root finding problem admits a solution! Unfortunately, very little is known, even in the case $\lambda = 0$. And, if an equilibrium does exist, stability theory is nearly absent.

5.4.4 GQ-learning

If we are concerned that $\overline{f}(\theta^*) = 0$ does not admit a solution, then we might turn to the next best option: solve the optimization problem:

$$\min_\theta L(\theta) = \min_\theta \tfrac{1}{2}\overline{f}(\theta)^\intercal M \overline{f}(\theta), \qquad \text{with} \quad M > 0 \tag{5.47}$$

We can then apply the ODE method to devise an algorithm. One approach is gradient descent:

$$\tfrac{d}{dt}\vartheta_t = -[\partial_\theta \overline{f}\,(\vartheta_t)]^\intercal M \overline{f}(\vartheta_t) \tag{5.48}$$

The GQ-learning algorithm of [136] can be regarded as a direct discrete-time translation of this ODE, using $M = \mathsf{E}[\zeta_n \zeta_n^\intercal]^{-1}$.

To do: insert pretty algorithm that doesn't require matrix inversion

There are several important challenges and questions:

(i) The function $\overline{f}$ is not convex, so we may have difficulty obtaining a global minimum of (5.47).

(ii) Suppose that in fact $\overline{f}(\theta^*) = 0$ *does have a solution.* One might turn to (5.47) for computation of θ^*. We are then faced with numerical challenges beyond lack of convexity. Nesterov discusses this approach to root finding in his monograph. In [160, Section 4.4.1] he warns that it can lead to numerical instability: "*...if our system of equations is linear, then such a transformation squares the condition number of the problem*". He goes on to warn that it can lead to a "*squaring the number of iterations*" to obtain the desired error bound. To see this, consider the second-order approximation of the loss function L at θ^*:

$$L(\theta) \approx L(\theta^*) + (\theta - \theta^*)^\intercal [A^* M A^{*\intercal}](\theta - \theta^*)$$

which uses $\overline{f}(\theta^*) = 0$, and the error is $o(\|\theta - \theta^*\|^2)$ provided $\partial_\theta \overline{f}$ is Lipschitz continuous. The appearance of $A^* M A^{*\intercal}$ is the "squaring" that Nesterov warns about. If A^* has a large condition number, then we may make things much worse by squaring. The numerical challenge can be addressed with a good choice for M, provided this can be introduced without introducing additional complexity.

(iii) Last but not least: is solving $\overline{f}(\theta^*) = 0$ a worthwhile goal? The answer to that question is currently a topic for research.

5.4.5 Batch Methods and DQN

The *Deep Q Network* (DQN) algorithm was designed for neural network function approximation; the term "deep" refers to a large number of hidden layers. The basic algorithm is summarized here, without imposing any particular form for Q^θ.

An important component of this approach is to abandon the purely recursive form of the preceding RL algorithms. In a batch RL algorithm, the time-horizon N is broken into B batches of more reasonable size, defined by the sequence of intermediate times $T_0 = 0 < T_1 < T_2 < \cdots < T_{B-1} < T_B = N$. The architecture is designed to reduce complexity, and there are many other potential benefits that are more obvious when we come to RL design for stochastic control systems.

DQN

With $\theta_0 \in \mathbb{R}^d$ given, along with a sequence of positive scalars $\{\alpha_n\}$, define recursively,

$$\theta_{n+1} = \arg\min_\theta \Big\{ \mathcal{E}_n^\varepsilon(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|^2 \Big\} \tag{5.49a}$$

where for each n:

$$\mathcal{E}_n^{\varepsilon}(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[-Q^{\theta}(x(k),u(k)) + c(x(k),u(k)) + \underline{Q}^{\theta_n}(x(k+1))\big]^2 \tag{5.49b}$$

with $r_n = 1/(T_{n+1} - T_n)$.
Output of the algorithm: θ_B, to define the final approximation Q^{θ_B}.

The elegance and simplicity of DQN is clear. Most significant: if Q^{θ} is defined via linear function approximation, then the minimization (5.49a) is the unconstrained minimum of a quadratic. On denoting

$$y_n(k) = c((x(k),u(k)) + \underline{Q}^{\theta_n}(x(k+1))$$

we obtain θ_{n+1} by taking the gradient of the left hand side of (5.49a), and setting it equal to zero:

$$0 = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \psi(x(k),u(k))\big[Q^{\theta}(x(k),u(k)) - y_n(k)\big]\Big|_{\theta=\theta_{n+1}} + \frac{1}{\alpha_{n+1}}[\theta_{n+1} - \theta_n]$$

where $Q^{\theta} = \psi^{\intercal}\theta$. After simplifying terms this becomes

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big\{A_n\theta_{n+1} - b_n\big\} \tag{5.50a}$$

$$\textit{with} \quad A_n = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \psi(x(k),u(k))\psi(x(k),u(k))^{\intercal} \tag{5.50b}$$

$$b_n = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} y_n(k)\psi(x(k),u(k)) \tag{5.50c}$$

This simplicity does not come for free, since the convex structure of the overall function approximation problem has been abandoned.

Proposition 5.6. *Consider the DQN algorithm with possibly nonlinear function approximation, and with* $\zeta_k = \nabla_{\theta} Q^{\theta}(x(k),u(k))\Big|_{\theta=\theta_k}$. *Assume that* Q^{θ} *is continuously differentiable, and its gradient* $\nabla Q^{\theta}(x,u)$ *is globally Lipschitz continuous, with Lipschitz constant independent of* (x,u). *Suppose that* $B = \infty$, *the non-negative step-size sequence satisfies* $\alpha_n = \alpha_1/n$, *with* $\alpha_1 > 0$, *and suppose that the sequence* $\{\theta_n\}$ *defined by the DQN algorithm is convergent to some* $\theta_{\infty} \in \mathbb{R}^d$.

Then, this limit is a solution to (5.46), *and moreover the algorithm admits the ODE approximation* $\frac{d}{dt}\vartheta_t = \overline{f}(\vartheta_t)$ *using the vector field* $\overline{f}$ *defined in* (5.46). ☐

The assumption on the step-size is to facilitate a simple proof. This can be replaced by the standard assumptions:

$$\sum \alpha_n = \infty\,, \qquad \sum \alpha_n^2 < \infty$$

The proof of Prop. 5.6 is contained at the end of Section 5.5. Its conclusion should raise a warning, since we do not know if (5.46) has a solution, or if a solution has desirable properties.

5.5 Convex Q-learning

So far in this chapter we have surveyed algorithms that have been popular for the past decade and have been successfully applied, but at this stage there is little supporting theory. In particular, we currently lack strong motivation for focusing on the projected Bellman equation as a guide to approximating the optimal policy.

The following transformation was presented in Section 5.1.1: For any function $Q\colon \mathsf{X}\times\mathsf{U}\to\mathbb{R}$, denote $\underline{Q}(x)=\min_u Q(x,u)$. Then (3.5) implies $\underline{Q^\star}=J^\star$, and hence also the fixed point equation

$$Q^\star(x,u)=c(x,u)+\underline{Q^\star}(\mathrm{F}(x,u)) \tag{5.51}$$

and the sample path representation, based on (3.1):

$$Q^\star(x(k),u(k))=c((x(k),u(k))+\underline{Q^\star}(x(k+1)) \tag{5.52}$$

Consider a parameterized family of functions $\{Q^\theta:\theta\in\mathbb{R}^d\}$. It is common to use gradient descent to find a local minimum of $\mathcal{E}(Q^\theta)$, with $\mathcal{E}(h)$defined in (5.5) for any $h\colon \mathsf{X}\times\mathsf{U}\to\mathbb{R}$. This (with various refinements) is how the complex Go and chess games were tamed in [184].

A challenge is that $\mathcal{E}(Q^\theta)$ may be a horrible function of θ, even for linear function approximation. Gradient descent may not find a good value of θ even if there is a value θ^* satisfying $Q^{\theta^*}=Q^\star$. This difficulty is easily resolved by applying the convex representations of the dynamic programming equations to obtain a convex loss function. For this, we need a joint parameterization: in the notation of Section 5.1, $H^\theta=(J^\theta,Q^\theta)$.

Consider the direct translation of (3.34) based on a parameterized family:

$$\begin{aligned}\max_\theta\ & \langle\mu,Q^\theta\rangle\\ \text{s.t.}\ & Q^\theta(x,u)\le c(x,u)+J^\theta(\mathrm{F}(x,u))\\ & Q^\theta(x,u)\ge J^\theta(x)\qquad x\in\mathsf{X},\ u\in\mathsf{U}(x)\end{aligned} \tag{5.53}$$

We can if we wish strengthen the first constraint to equality:

$$\begin{aligned}\max_\theta\ & \langle\mu,Q^\theta\rangle\\ \text{s.t.}\ & Q^\theta(x,u)=c(x,u)+J^\theta(\mathrm{F}(x,u))\\ & Q^\theta(x,u)\ge J^\theta(x)\qquad x\in\mathsf{X},\ u\in\mathsf{U}(x)\end{aligned} \tag{5.54}$$

The latter may be reasonable if we have a good model. In this case, we might first create a parameterized family $\{J^\theta\}$, and then *define* $Q^\theta(x,u)=c(x,u)+J^\theta(\mathrm{F}(x,u))$ for each θ, x,u.

However, if we don't have a highly accurate model, it is then better to relax our constraints. Several options are surveyed in the following. In each case, the value function approximation algorithm is designed to make use of input-state pairs $\{(u(k),x(k):0\le k\le N\}$.

LP Q-Learning

Choose a linear function class for $H^\theta=(J^\theta,Q^\theta)$, and use (5.53) to define the optimal parameter:

$$\begin{aligned}\theta^*=\arg\max_\theta\ & \langle\mu,Q^\theta\rangle\\ \text{s.t.}\ & Q^\theta(x(k),u(k))\le c(x(k),u(k))+J^\theta(x(k+1))\\ & Q^\theta(x(k),u(k))\ge J^\theta(x(k))\qquad 0\le k\le N-1\end{aligned} \tag{5.55}$$

An obvious challenge with this algorithm is that there are N constraints, where in many cases N will be over one million.

The RL algorithms introduced in this paper are all motivated by the "DPLP" (3.34). We search for an approximate solution among a finite-dimensional family $\{J^\theta\,,\ Q^\theta : \theta \in \mathbb{R}^d\}$. The value θ_i might represent the ith weight in a neural network function approximation architecture, but to justify the adjective *convex* we require a linearly parameterized family:

$$J^\theta(x) = \theta^\intercal \psi^J(x)\,, \qquad Q^\theta(x,u) = \theta^\intercal \psi(x,u) \tag{5.56}$$

The function class is normalized with $J^\theta(x^e) = 0$ for each θ. For the linear approximation architecture this requires $\psi_i^J(x^e) = 0$ for each $1 \le i \le d$; for a neural network architecture, this normalization is imposed through definition of the output of the network. Convex Q-learning based on a *reproducing kernel Hilbert space* (RKHS) are contained in Section 5.5.2.

Recall the MSE loss (**??**) presents challenges because it is not convex for linear function approximation. The quadratic program (3.37) obtained from the DPLP motivates the variation of (**??**):

$$\mathcal{E}^\varepsilon(\theta) = \frac{1}{N}\sum_{k=0}^{N-1} \big[\mathcal{D}^\circ_{k+1}(\theta)\big]^2 \tag{5.57}$$

with temporal difference defined by a modification of (**??**):

$$\mathcal{D}^\circ_{k+1}(\theta) := -Q^\theta(x(k),u(k)) + c(x(k),u(k)) + J^\theta(x(k+1)) \tag{5.58}$$

The algorithms are designed so that J^θ approximates $\underline{Q}^\theta$, and hence $\mathcal{D}^\circ_{k+1}(\theta) \approx \mathcal{D}_{k+1}(\theta)$ (recall from below (**??**), $\underline{Q}(x) = \min_u Q(x,u)$ for any function Q).

The first of several versions of "CQL" involves a Galerkin relaxation of the constraints in the DPLP (3.34). This requires specification of two vector valued sequences $\{\zeta_k, \zeta_k^+\}$ based on the data, and denote

$$z^\varepsilon(\theta) = \frac{1}{N}\sum_{k=0}^{N-1} \mathcal{D}^\circ_{k+1}(\theta)\zeta_k \tag{5.59a}$$

$$z^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1} \big[J^\theta(x(k)) - Q^\theta(x(k),u(k))\big]\zeta_k^+ \tag{5.59b}$$

with temporal difference sequence $\{\mathcal{D}^\circ_{k+1}(\theta)\}$ defined in (5.58). It is assumed that ζ_k takes values in $\mathbb{R}^d$, and that the entries of the vector ζ_k^+ are non-negative for each k.

LP Convex Q-Learning

$$\theta^* = \arg\max_\theta \ \ \langle \mu, J^\theta\rangle \tag{5.60a}$$

$$\text{s.t.} \quad z^\varepsilon(\theta) = 0 \tag{5.60b}$$

$$z^+(\theta) \le 0 \tag{5.60c}$$

This algorithm is introduced mainly because it is the most obvious translation of the general DPLP (3.34). A preferred algorithm described next is motivated by the quadratic program (3.37), which directly penalizes Bellman error. The objective function is modified to include the empirical mean-square error (5.57) and a second loss function:

$$\mathcal{E}^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\{J^\theta(x(k)) - Q^\theta(x(k),u(k))\}_+\big]^2 \tag{5.61a}$$

$$\textit{or}\quad \mathcal{E}^+(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\big[\{J^\theta(x(k)) - \underline{Q}^\theta(x(k))\}_+\big]^2 \tag{5.61b}$$

where $\{z\}_+ = \max(z,0)$. The second option (5.61b) more strongly penalizes deviation from the constraint $\underline{Q}^\theta \geq J^\theta$. The choice of definition (5.61a) or (5.61b) will depend on the relative complexity, which is application-specific.

Convex Q-Learning

For positive scalars κ^ε and κ^+, and a tolerance $\text{Tol} \geq 0$,

$$\theta^* = \arg\min_\theta \big\{-\langle\mu, J^\theta\rangle + \kappa^\varepsilon\mathcal{E}^\varepsilon(\theta) + \kappa^+\mathcal{E}^+(\theta)\big\} \tag{5.62a}$$

$$\text{s.t.}\quad z^\varepsilon(\theta) = 0 \tag{5.62b}$$

$$z^+(\theta) \leq \text{Tol} \tag{5.62c}$$

To understand why the optimization problem (5.60) or (5.62) may present challenges, consider their implementation based on a kernel. Either of these optimization problems is a convex program. However, due to the Representer Theorem [32], the dimension of θ is equal to the number of observations N. Even in simple examples, the value of N for a reliable estimate may be larger than one million.

The following batch RL algorithm is designed to reduce complexity, and there are many other potential benefits [127]. The time-horizon N is broken into B batches of more reasonable size, defined by the sequence of intermediate times $T_0 = 0 < T_1 < T_2 < \cdots < T_{B-1} < T_B = N$. Also required are a sequence of *regularizers*: $\mathcal{R}_n(J,Q,\theta)$ is a convex functional of J, Q, θ, that may depend on θ_n. Examples are provided below.

Batch Convex Q-Learning

With $\theta_0 \in \mathbb{R}^d$ given, along with a sequence of positive scalars $\{\kappa_n^\varepsilon, \kappa_n^+\}$, define recursively,

$$\theta_{n+1} = \arg\min_\theta \Big\{-\langle\mu, J^\theta\rangle + \kappa_n^\varepsilon\mathcal{E}_n^\varepsilon(\theta) + \kappa_n^+\mathcal{E}_n^+(\theta) + \mathcal{R}_n(J^\theta, Q^\theta, \theta)\Big\} \tag{5.63a}$$

where for $0 \le n \le B-1$,

$$\mathcal{E}_n^{\varepsilon}(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[-Q^\theta(x(k),u(k)) + c(x(k),u(k)) + J^\theta(x(k+1))\big]^2 \tag{5.63b}$$

$$\mathcal{E}_n^{+}(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[\{J^\theta(x(k)) - Q^\theta(x(k),u(k))\}_+\big]^2 \tag{5.63c}$$

$$\textit{or} \quad \mathcal{E}_n^{+}(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[\{J^\theta(x(k)) - \underline{Q}^\theta(x(k))\}_+\big]^2 \tag{5.63d}$$

with $r_n = 1/(T_{n+1} - T_n)$.
Output of the algorithm: θ_B, to define the final approximation Q^{θ_B}.

The constraints (5.62b, 5.62c) are relaxed in BCQL only to streamline the discussion that follows.

How to choose a regularizer? It is expected that design of $\mathcal{R}_n$ will be inspired by proximal algorithms, so that it will include a term of the form $\|\theta-\theta_n\|^2$ (most likely a weighted norm—see discussion in Section 5.5.1). With a simple scaled norm, the recursion becomes

$$\theta_{n+1} = \arg\min_\theta \Big\{ -\langle \mu, J^\theta\rangle + \kappa_n^{\varepsilon}\mathcal{E}_n^{\varepsilon}(\theta) + \kappa_n^{+}\mathcal{E}_n^{+}(\theta) + \frac{1}{\alpha_{n+1}} \tfrac{1}{2}\|\theta-\theta_n\|^2 \Big\} \tag{5.64}$$

where $\{\alpha_n\}$ plays a role similar to the step-size in stochastic approximation.

Convergence in this special case is established in the the following result, based on the steady-state expectations (recall (5.19)):

$$\bar{\mathcal{E}}^{\varepsilon}(\theta) = \mathsf{E}_\varpi\big[\{-Q^\theta(x(k),u(k)) + c(x(k),u(k)) + J^\theta(x(k+1))\}^2\big] \tag{5.65a}$$

$$\bar{\mathcal{E}}^{+}(\theta) = \mathsf{E}_\varpi[\{J^\theta(x(k)) - Q^\theta(x(k),u(k))\}_+^2] \tag{5.65b}$$

Proposition 5.7. *Consider the BCQL algorithm (5.64) subject to the following assumptions:*

(i) *The parameterization (J^θ, Q^θ) is linear, and $\bar{\mathcal{E}}^{\varepsilon}$ is strongly convex.*

(ii) *The non-negative step-size sequence is of the form $\alpha_n = \alpha_1/n$, with $\alpha_1 > 0$.*

(iii) *The parameters reach steady-state limits:*

$$r := \lim_{n\to\infty} r_n\,, \quad \kappa^{\varepsilon} := \lim_{n\to\infty} \kappa_n^{\varepsilon}\,, \quad \kappa^{+} := \lim_{n\to\infty} \kappa_n^{+}$$

Then, the algorithm is consistent: $\theta_n \to \theta^*$ as $n \to \infty$, where the limit is the unique optimizer:

$$\theta^* = \arg\min_\theta \Big\{ -\langle \mu, J^\theta\rangle + \kappa^{\varepsilon}\bar{\mathcal{E}}^{\varepsilon}(\theta) + \kappa^{+}\bar{\mathcal{E}}^{+}(\theta) \Big\} \tag{5.66}$$

To do: migrate proof from [143] Strong convexity of $\bar{\mathcal{E}}^\varepsilon$ is obtained by design, which is not difficult since it is a quadratic function of θ:

$$\bar{\mathcal{E}}^\varepsilon(\theta) = \theta^\intercal M\theta + 2\theta^\intercal b + k$$

with $b \in \mathbb{R}^d$, $k \geq 0$, and

$$M = \mathsf{E}_\varpi\left[\Upsilon_{k+1}\Upsilon_{k+1}^\intercal\right], \qquad \textit{with} \quad \Upsilon_{k+1} = \psi(x(k), u(k)) - \psi^J(x(k+1))$$

A few words on constraints It is found in experiments that a pure penalty approach leads to numerical instability.

The pd-BCQL algorithm defined in (5.70) is motivated by a relaxation of the inequality constraint (3.34b) required in the DPLP:

$$z^\varepsilon(\theta) \geq -\text{Tol} \tag{5.67}$$

Recall that this is a valid relaxation only if we impose positivity on the entries of ζ_k. In experiments it is found that imposing hard constraints in the convex program is more effective than the pure penalty approach used in BCQL.

The CQL algorithms introduce data-driven relaxations of the constraint (3.34c) in the DPLP: consider the constraint (5.60c) or (5.62c), or the penalty $\mathcal{E}_n^+(\theta)$ in BCQL. A data driven approach may be convenient, *but it is unlikely that this is the best option.* The purpose of these constraints is to enforce non-negativity of the difference

$$A^\theta(x,u) = Q^\theta(x,u) - J^\theta(x) \tag{5.68}$$

This is known as the *advantage function*. There are at least two options to enforce or approximate the inequality $A^\theta \geq 0$:

1. Choose a parameterization, along with constraints on θ, so that non-negativity of A^θ is automatic. Consider for example a linear parameterization in which $d = d^J + d^Q$, and with basis functions $\{\psi^J, \psi^A\}$ satisfying the following constraints:

$$\psi_i^J(x) = 0 \text{ for all } x, \text{ and all } i > d^J$$
$$\psi_i^A(x,u) = 0 \text{ for all } x, u, \text{ and all } i \leq d^J$$

subject to the further constraint that $\psi_i^A(x,u) \geq 0$ for all i, x, u.

We then define $\psi = \psi^J + \psi^A$, $J^\theta(x) = \theta^\intercal\psi^J(x,u)$, $Q^\theta(x,u) = \theta^\intercal\psi(x,u)$, and $A^\theta(x,u) = \theta^\intercal\psi^A(x,u)$, so that for any $\theta \in \mathbb{R}^d$,

$$Q^\theta(x,u) = J^\theta(x) + A^\theta(x,u)$$

It follows that $Q^\theta(x,u) \geq J^\theta(x)$ for all x, u, provided we impose the constraint $\theta_i \geq 0$ for $i > d^J$. Using this approach, the penalty term $\kappa^+\mathcal{E}^+(\theta)$ may be eliminated from (5.62), as well as the constraint $z^+(\theta) \leq \text{Tol}$. It was found that this approach was most reliable in experiments conducted so far, along with the relaxation (5.67), since the algorithm reduces to a convex program (much like DQN).

2. Choose a grid of points $G \subset \mathsf{X} \times \mathsf{U}$, and replace (5.62c) with the simple inequality constraint

$$A^\theta(x^i, u^i) \geq -\text{Tol for } (x^i, u^i) \in G.$$

For example, this approach is reasonable for the LQR problem and similar control problems that involve linear matrix inequalities (such as eq. (3.42b)). In particular, the constraints in (3.42) cannot be captured by a linear parameterization of the matrices M, so application of approach 1 can only approximate a subset of the constraint set.

Based on these considerations we introduce a modification of BCQL: the definition of $\mathcal{E}_n^\varepsilon$ from (5.63b) is maintained, and we bring back *inequality* constraints on the temporal difference, consistent with the DPLP constraint (3.34b). The batch version of (5.59a) is denoted

$$z_n^\varepsilon(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \mathcal{D}_{k+1}^\circ(\theta)\zeta_k \tag{5.69}$$

We require $\zeta_k(i) \geq 0$ for all k, i to ensure that the inequality constraint $z_n^\varepsilon(\theta) \geq 0$ is consistent with (3.34b)

With $\theta_0 \in \mathbb{R}^d$ given, along with a positive scalar κ^ε, consider the primal dual variant of BCQL (pd-BCQL):

$$\theta_{n+1} = \arg\min_{\theta\in\Theta} \Big\{ -\langle \mu, J^\theta \rangle + \kappa^\varepsilon \mathcal{E}_n^\varepsilon(\theta) - \lambda_n^\intercal z_n^\varepsilon(\theta) + \frac{1}{\alpha_{n+1}} \tfrac{1}{2}\|\theta - \theta_n\|^2 \Big\} \tag{5.70a}$$

$$\lambda_{n+1} = \big[\lambda_n - \alpha_{n+1} z_n^\varepsilon(\theta)\big]_+ \tag{5.70b}$$

where the subscript "+" in the second recursion is a component-wise projection: for each i, the component $\lambda_{n+1}(i)$ is constrained to an interval $[0, \lambda^{\max}(i)]$.

This algorithm is designed to solve the convex program

$$\begin{aligned} \min_\theta \ & -\langle \mu, J^\theta \rangle + \kappa^\varepsilon \mathsf{E}\big[\big\{\mathcal{D}_{k+1}^\circ(\theta)\big\}^2\big] \\ \text{s.t. } & \mathsf{E}\big[\mathcal{D}_{k+1}^\circ(\theta)\zeta_k(i)\big] \geq 0\,, \qquad i \geq 1 \end{aligned} \tag{5.71}$$

Corollary 5.8. *Suppose that assumptions (i) and (ii) of Prop. 5.7 hold. In addition, assume that $A^\theta(x,u) = Q^\theta(x,u) - J^\theta(x)$ is non-negative valued for $\theta \in \Theta$, where Θ is a polyhedral cone with non-empty interior. Then, the sequence $\{\theta_n, \lambda_n\}$ obtained from the pd-BCQL algorithm (5.70) is convergent to a pair (θ^*, λ^*), and the following hold:*

(i) *The limit θ^* is the solution of the convex program (5.71), and λ^* is the Lagrange multiplier for the linear inequality constraint in (5.71).*

(ii) *Consider the special case: the state space and action space are finite, μ has full support, and the dimension of ζ_k is equal to $d_\zeta = |\mathsf{X}| \times |\mathsf{U}|$, with*

$$\zeta_k(i) = \mathbf{1}\{(x(k), u(k)) = (x^i, u^i)\}$$

where $\{(x^i, u^i)\}$ is an enumeration of all state action pairs. Suppose moreover that $(J^\star, Q^\star)$ is contained in the function class. Then,

$$(J^\theta, Q^\theta)\Big|_{\theta=\theta^*} = (J^\star, Q^\star)$$

□

Corollary 5.8 is a corollary to Prop. 5.7, in the sense that it follows the same proof for the joint sequence $\{\theta_n, \lambda_n\}$. Both the proposition and corollary start with a proof that $\{\theta_n\}$ is a bounded sequence, using the "Borkar-Meyn" Theorem [39, 36, 169, 170] (based on a scaled ODE). The cone assumption on Θ is imposed to simplify the proof that the algorithm is stable. Once boundedness is established, the next step is to show that the algorithm (5.70) can be approximated by a primal-dual ODE for the saddle point problem

$$\max_{\lambda\geq 0}\min_{\theta\in\Theta}\bar{L}(\theta,\lambda)\,,\qquad \bar{L}(\theta,\lambda):=-\langle\mu,J^\theta\rangle+\kappa^\varepsilon\bar{\mathcal{E}}^\varepsilon(\theta)-\lambda^\intercal\bar{z}_n^\varepsilon(\theta) \tag{5.72}$$

The function $\bar{L}$ is quadratic and strictly convex in θ, and linear in λ. A suitable choice for the upper bound $\lambda^{\max}$ can be found through inspection of this saddle-point problem.

This approximation suggests improvements to the algorithm. For example, the use of an augmented Lagrangian to ensure strict convexity in λ. We might also make better use of the solution to the quadratic program (5.70a) to improve estimation of λ^*.

The choice of regularizer is more subtle when we consider kernel methods, so that the "parameterization" is infinite dimensional. Discussion on this topic is postponed to Section 5.5.2.

5.5.1 Gain selection and SA approximations

Consider the introduction of a weighted norm in (5.64):

$$\begin{aligned}\theta_{n+1}&=\arg\min_\theta\big\{\mathcal{E}_n(\theta)+\frac{1}{\alpha_{n+1}}\tfrac{1}{2}\|\theta-\theta_n\|^2_{W_n}\big\}\\ \mathcal{E}_n(\theta)&=-\langle\mu,J^\theta\rangle+\kappa_n^\varepsilon\mathcal{E}_n^\varepsilon(\theta)+\kappa_n^+\mathcal{E}_n^+(\theta)\end{aligned} \tag{5.73}$$

where $\|\vartheta\|^2_{W_n}=\frac{1}{2}\vartheta^\intercal W_n\vartheta$ for $\vartheta\in\mathbb{R}^d$, with $W_n>0$.

The recursion can be represented in a form similar to stochastic approximation (SA):

Lemma 5.9. *Suppose that $\{\mathcal{E}_n(\theta)\}$ are continuously differentiable in θ. Then, the parameter update in BCQL is the solution to the fixed point equation:*

$$\theta_{n+1}=\theta_n-\alpha_{n+1}W_n^{-1}\nabla\mathcal{E}_n(\theta_{n+1}) \tag{5.74}$$

Suppose in addition that $\nabla\mathcal{E}_n$ is Lipschitz continuous (uniformly in n), and the sequences $\{\theta_n\}$ and $\{\operatorname{trace}(W_n^{-1})\}$ are uniformly bounded. Then,

$$\theta_{n+1}=\theta_n-\alpha_{n+1}\{W_n^{-1}\nabla\mathcal{E}_n(\theta_n)+\varepsilon_{n+1}\} \tag{5.75}$$

where $\|\varepsilon_{n+1}\|=O(\alpha_{n+1})$.

Proof. The fixed point equation (5.74) follows from the first-order condition for optimality:

$$0=\nabla\big\{\mathcal{E}_n(\theta)+\frac{1}{\alpha_{n+1}}\tfrac{1}{2}\|\theta-\theta_n\|^2_{W_n}\big\}\big|_{\theta=\theta_{n+1}}=\nabla\mathcal{E}_n(\theta_{n+1})+\frac{1}{\alpha_{n+1}}\tfrac{1}{2}W_n[\theta_{n+1}-\theta_n]$$

To obtain the second conclusion, note that (5.74) implies $\|\theta_{n+1}-\theta_n\|=O(\alpha_{n+1})$ under the boundedness assumptions, and then Lipschitz continuity of $\{\nabla\mathcal{E}_n\}$ in θ then gives the desired representation (5.75). ∎

The Zap SA algorithm of [70] is designed for recursions of the form (5.75) so that

$$W_n \approx \nabla^2 \bar{\mathcal{E}}(\theta_n)$$

in which the bar designates a steady-state expectation (recall (5.19)).

For the problem at hand, this is achieved in the following steps. First, define for each n,

$$A_{n+1} = \nabla^2 \mathcal{E}_n(\theta_n)$$

For the linear parametrization using (5.63c) we obtain simple expressions. To compress notation, denote

$$\psi_{(k)} = \psi(x(k), u(k)), \qquad \psi^J_{(k)} = \psi^J(x(k)).$$

so that (5.73) gives

$$\nabla \mathcal{E}_n(\theta) = 2\frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \Big\{ -\langle \mu, \psi^J \rangle + \kappa_n^{\varepsilon} \mathcal{D}^{\circ}_{k+1}(\theta) \big[\psi^J_{(k+1)} - \psi_{(k)}\big] + \kappa_n^{+} \{J^\theta(x(k)) - \underline{Q}^\theta(x(k))\}_+ \big[\psi^J_{(k)} - \psi_{(k)}\big] \Big\}$$

where $\langle \mu, \psi^J \rangle$ is the column vector whose ith entry is $\langle \mu, \psi^J_i \rangle$. Taking derivatives once more gives

$$A_{n+1} = 2\frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \Big\{ \big[\psi_{(k)} - \psi^J_{(k+1)}\big]\big[\psi_{(k)} - \psi^J_{(k+1)}\big]^\intercal + \kappa_n^{+} \big[\psi_{(k)} - \psi^J_{(k)}\big]\big[\psi_{(k)} - \psi^J_{(k)}\big]^\intercal \mathbf{1}\{J^\theta(x(k)) > \underline{Q}^\theta(x(k))\} \Big\}$$

We then take $W_0 > 0$ arbitrary, and for $n \geq 0$,

$$W_{n+1} = W_n + \beta_{n+1}[A_{n+1} - W_n] \tag{5.76}$$

in which the step-size for this matrix recursion is relatively large:

$$\lim_{n\to\infty} \frac{\alpha_n}{\beta_n} = 0$$

In [70] the choice $\beta_n = \alpha_n^\eta$ is proposed, with $\frac{1}{2} < \eta < 1$.

The original motivation in [71, 72, 67] was to minimize algorithm variance It is now known that the "Zap gain" (5.76) often leads to a stable algorithm, even for nonlinear function approximation (such as neural networks) [48].

It may be advisable to simply use the recursive form of the batch algorithm: make the change of notation $\widehat{A}_n = W_n$ (to highlight the similarity with the Zap algorithms in [70]), and define recursively

$$\theta_{n+1} = \theta_n - \alpha_{n+1} \widehat{A}_n^{-1} \nabla \mathcal{E}_n(\theta_n)$$
$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}[A_{n+1} - \widehat{A}_n]$$

In preliminary experiments it is found that this leads to much higher variance, but this may be offset by the reduced complexity.

5.5.2 BCQL and kernel methods

The reader is referred to other sources, such as [32], for the definition of a reproducing kernel Hilbert space (RKHS) and surrounding theory. In this subsection, the Hilbert space $\mathcal{H}$ defines a two dimensional function class that defines approximations (J, Q). One formulation of this method is to choose a kernel $\Bbbk$ on $(\mathsf{X}\times\mathsf{U})^2$, in which $\Bbbk((x,u),(x',u'))$ is a symmetric and positive definite matrix for each $x, x' \in \mathsf{X}$ and $u, u' \in \mathsf{U}$. The function J does not depend on u, so for $(f, g) \in \mathcal{H}$ we associate g with Q, but take

$$J(x) = f(x, u^\circ)$$

for some distinguished $u^\circ \in \mathsf{U}$.

A candidate regularizer in this case is

$$\mathcal{R}_n(J,Q) = \frac{1}{\alpha_{n+1}} \tfrac{1}{2}\|(J,Q) - (J^n, Q^n)\|^2_{\mathcal{H}} \tag{5.77}$$

where $(J^n, Q^n) \in \mathcal{H}$ is the estimate at stage n based on the kernel BCQL method.

The notation must be modified in this setting:

Kernel Batch Convex Q-Learning

With $(J^0, Q^0) \in \mathcal{H}$ given, along with a sequence of positive scalars $\{\kappa_n^\varepsilon, \kappa_n^+\}$, define recursively,

$$(J^{n+1}, Q^{n+1}) = \arg\min_{J,Q} \Big\{ -\langle \mu, J\rangle + \kappa_n^\varepsilon \mathcal{E}_n^\varepsilon(J,Q) + \kappa_n^+ \mathcal{E}_n^+(J,Q) + \mathcal{R}_n(J,Q) \Big\} \tag{5.78a}$$

where for $0 \le n \le B-1$,

$$\mathcal{E}_n^\varepsilon(J,Q) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[-Q(x(k),u(k)) + c(x(k),u(k)) + J(x(k+1)) \big]^2 \tag{5.78b}$$

$$\mathcal{E}_n^+(J,Q) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[\{J(x(k)) - Q(x(k),u(k))\}_+ \big]^2 \tag{5.78c}$$

$$\textit{or} \quad \mathcal{E}_n^+(J,Q) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \big[\{J(x(k)) - \underline{Q}(x(k))\}_+ \big]^2 \tag{5.78d}$$

The regularizer (5.77) is chosen so that we can apply the Representer Theorem to solve this infinite-dimensional optimization problem (5.78). The theorem states that computation of (J^n, Q^n) reduces to a finite dimensional setting, with a linearly parameterized family similar to (5.56). However, the "basis functions" ψ^J and ψ depend upon n: for some $\{\theta_i^{n*}\} \subset \mathbb{R}^2$,

$$J^n(x) = \sum_i \{\theta_i^{n\star\intercal} \Bbbk((x_i,u_i),(x,u^\circ))\}_1$$

$$Q^n(x,u) = \sum_i \{\theta_i^{n\star\intercal} \Bbbk((x_i,u_i),(x,u))\}_2$$

where $\{x_i, u_i\}$ are the state-input pairs observed on the time interval $\{T_{n-1} \le k < T_n\}$.

List of Figures

Bibliography

[1] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[2] L. L. Andrew, M. Lin, and A. Wierman. Optimality, fairness, and robustness in speed scaling designs. *SIGMETRICS Perform. Eval. Rev.*, 38(1):37–48, June 2010.

[3] C. Andrieu, E. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM J. Control Optim.*, 44(1):283–312, 2005.

[4] O. Anschel, N. Baram, and N. Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *Proc. of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 176–185. JMLR.org, 2017.

[5] A. Arapostathis, V. S. Borkar, E. Fernandez-Gaucherand, M. K. Ghosh, and S. I. Marcus. Discrete-time controlled Markov processes with average cost criterion: a survey. *SIAM J. Control Optim.*, 31:282–344, 1993.

[6] K. B. Ariyur and M. Krstić. *Real Time Optimization by Extremum Seeking Control*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[7] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, New York, 2007.

[8] K. Åström and K. Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287 – 295, 2000.

[9] K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, USA, 2008.

[10] B. Awerbuch and R. Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97 – 114, 2008. Learning Theory 2004.

[11] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. Kappen. Speedy Q-learning. In *Advances in Neural Information Processing Systems*, 2011.

[12] F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $o(1/n)$. In *Advances in Neural Information Processing Systems 26*, pages 773–781. Curran Associates, Inc., 2013.

[13] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In A. Prieditis and S. Russell, editors, *Machine Learning Proceedings 1995*, pages 30 – 37. Morgan Kaufmann, San Francisco (CA), 1995.

[14] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, Mar. 2007.

[15] A. Barto, R. Sutton, and C. Anderson. Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man and Cybernetics*, 13(5):835–846, 1983.

[16] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In *LEARNING AND COMPUTATIONAL NEUROSCIENCE*, pages 539–602. MIT Press, 1989.

[17] T. Basar, S. Meyn, and W. R. Perkins. Lecture notes on control system theory and design. UIUC lecture notes, 2010.

[18] N. Bäuerle and J. Ott. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011.

[19] N. Bäuerle and U. Rieder. Partially observable risk-sensitive markov decision processes. *Mathematics of Operations Research*, 42(4):1180–1196, 2017.

[20] J. Beck. *Strong Uniformity and Large Dynamical Systems*. World Scientific, 2017.

[21] R. Bellman. The stability of solutions of linear differential equations. *Duke Math. J.*, 10(4):643–647, 12 1943.

[22] M. Benaïm. Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités, XXXIII*, pages 1–68. Springer, Berlin, 1999.

[23] A. Benveniste, M. Métivier, and P. Priouret. *Adaptive algorithms and stochastic approximations*. Springer, 2012.

[24] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. Optimal rate of convergence for quasi-stochastic approximation. *arXiv:1903.07228*, 2019.

[25] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. Quasi-stochastic approximation and off-policy reinforcement learning. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 5244–5251, Mar 2019.

[26] D. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Atena Scientific, Cambridge, Mass, 1996.

[27] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4th edition, 2012.

[28] J. Bhandari, D. Russo, and R. Singal. A finite time analysis of temporal difference learning with linear function approximation. *arXiv preprint arXiv:1806.02450*, 2018.

[29] S. Bhatnagar and V. S. Borkar. Multiscale chaotic spsa and smoothed functional algorithms for simulation optimization. *Simulation*, 79(10):568–580, 2003.

[30] S. Bhatnagar, M. C. Fu, S. I. Marcus, and I.-J. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(2):180–209, 2003.

[31] S. Bhatnagar, D. Precup, D. Silver, R. S. Sutton, H. R. Maei, and C. Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in neural information processing systems*, pages 1204–1212, 2009.

[32] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[33] J. R. Blum. Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics*, pages 737–744, 1954.

[34] V. S. Borkar. Convex analytic methods in Markov decision processes. In *Handbook of Markov decision processes*, volume 40 of *Internat. Ser. Oper. Res. Management Sci.*, pages 347–375. Kluwer Acad. Publ., Boston, MA, 2002.

[35] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency and Cambridge University Press (jointly), Delhi, India and Cambridge, UK, 2008.

[36] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint (2nd ed., to appear)*. Hindustan Book Agency, Delhi, India and Cambridge, UK, 2020.

[37] V. S. Borkar and V. Gaitsgory. Linear programming formulation of long-run average optimal control problem. *Journal of Optimization Theory and Applications*, 181(1):101–125, 2019.

[38] V. S. Borkar, V. Gaitsgory, and I. Shvartsman. LP formulations of discrete time long-run average optimal control problems: The non ergodic case. *SIAM Journal on Control and Optimization*, 57(3):1783–1817, 2019.

[39] V. S. Borkar and S. P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.*, 38(2):447–469, 2000. (see also *IEEE CDC*, 1998).

[40] J. A. Boyan. Technical update: Least-squares temporal difference learning. *Mach. Learn.*, 49(2-3):233–246, 2002.

[41] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15. SIAM, 1994.

[42] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.*, 22(1-3):33–57, 1996.

[43] W. L. Brogan. *Modern control theory*. Pearson, 3rd edition, 1990.

[44] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[45] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1–122, 2012.

[46] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[47] D. Chatterjee, A. Patra, and H. K. Joglekar. Swing-up and stabilization of a cart–pendulum system under restricted cart track length. *Systems & control letters*, 47(4):355–364, 2002.

[48] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. Zap Q Learning with nonlinear function approximation. Submitted for publication NeurIPS and arXiv e-prints 1910.05405, 2019.

[49] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. Explicit mean-square error bounds for Monte-Carlo and linear stochastic approximation. *AISTATS*, page arXiv:2002.02584, Feb. 2020.

[50] T. Chen and G. B. Giannakis. Bandit convex optimization for scalable and dynamic iot management. *IEEE Internet of Things Journal*, 6(1):1276–1286, Feb 2019.

[51] W. Chen, D. Huang, A. A. Kulkarni, J. Unnikrishnan, Q. Zhu, P. Mehta, S. Meyn, and A. Wierman. Approximate dynamic programming using fluid and diffusion approximations with applications to power management. In *Proc. of the 48th IEEE Conf. on Dec. and Control; held jointly with the 2009 28th Chinese Control Conference*, pages 3575–3580, 2009.

[52] Y. Chen, A. Bernstein, A. Devraj, and S. Meyn. Model-Free Primal-Dual Methods for Network Optimization with Application to Real-Time Optimal Power Flow. In *Proc. of the American Control Conf.*, pages 3140–3147, Sept. 2019.

[53] Z. Chen, S. Zhang, T. T. Doan, S. T. Maguluri, and J.-P. Clarke. Performance of Q-learning with linear function approximation: Stability and finite-time analysis. *arXiv: Optimization and Control*, 2019.

[54] A. Chorin, O. Hald, and R. Kupferman. Optimal prediction and the Mori-Zwanzig representation of irreversible processes. *Proc. Nat. Acad. Sci.*, 97:2968–2973, 2000.

[55] A. J. Chorin. Conditional expectations and renormalization. *J. Multiscale Modeling Simul.*, 1(1):105–118, 2003.

[56] K. L. Chung et al. On a stochastic approximation method. *The Annals of Mathematical Statistics*, 25(3):463–483, 1954.

[57] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song. SBEED: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pages 1133–1142, 2018.

[58] J. G. Dai. On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models. *Ann. Appl. Probab.*, 5(1):49–77, 1995.

[59] J. G. Dai and S. P. Meyn. Stability and convergence of moments for multiclass queueing networks via fluid limit models. *IEEE Trans. Automat. Control*, 40:1889–1904, November 1995.

[60] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor. Concentration bounds for two timescale stochastic approximation with applications to reinforcement learning. *Proceedings of the Conference on Computational Learning Theory, and ArXiv e-prints*, pages 1–35, 2017.

[61] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Res.*, 51(6):850–865, 2003.

[62] D. P. De Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, 29(3):462–478, 2004.

[63] D. P. de Farias and B. Van Roy. A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. *Math. Oper. Res.*, 31(3):597–620, 2006.

[64] A. Dembo and O. Zeitouni. *Large Deviations Techniques And Applications*. Springer-Verlag, New York, second edition, 1998.

[65] C. Derman. *Finite State Markovian Decision Processes*, volume 67 of *Mathematics in Science and Engineering*. Academic Press, Inc., 1970.

[66] J.-D. Deuschel and D. W. Stroock. *Large deviations*. Pure and applied mathematics. Academic Press, New York, London, 1989. édition révisée de : An introduction to the theory of large deviations / D.W. Stroock. cop.1984.

[67] A. M. Devraj. *Reinforcement Learning Design with Optimal Learning Rate*. PhD thesis, University of Florida, 2019.

[68] A. M. Devraj, A. Bušić, and S. Meyn. On matrix momentum stochastic approximation and applications to Q-learning. In *Allerton Conference on Communication, Control, and Computing*, pages 749–756, Sep 2019.

[69] A. M. Devraj, A. Bušić, and S. Meyn. Zap Q-Learning – a user's guide. In *Proc. of the Fifth Indian Control Conference*, January 9-11 2019.

[70] A. M. Devraj, A. Bušić, and S. Meyn. Fundamental design principles for reinforcement learning algorithms. In *Handbook on Reinforcement Learning and Control*. Springer, 2020.

[71] A. M. Devraj and S. P. Meyn. Fastest convergence for Q-learning. *ArXiv e-prints*, July 2017.

[72] A. M. Devraj and S. P. Meyn. Zap Q-learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[73] A. M. Devraj and S. P. Meyn. Q-learning with Uniformly Bounded Variance: Large Discounting is Not a Barrier to Fast Learning. *arXiv e-prints*, page arXiv:2002.10301, Feb. 2020.

[74] P. Diaconis. The Markov chain Monte Carlo revolution. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):179–205, 2009.

[75] R. Douc, E. Moulines, P. Priouret, and P. Soulier. *Markov Chains*. Springer, 2018.

[76] K. Duffy and S. Meyn. Large deviation asymptotics for busy periods. *Stochastic Systems*, 4(1):300–319, 2014.

[77] K. R. Duffy and S. P. Meyn. Most likely paths to error when estimating the mean of a reflected random walk. *Performance Evaluation*, 67(12):1290–1303, 2010.

[78] K. Dupree, P. M. Patre, M. Johnson, and W. E. Dixon. Inverse optimal adaptive control of a nonlinear Euler-Lagrange system, part i: Full state feedback. In *Proceedings of the 48h IEEE Conference on Decision and Control, held jointly with 2009 28th Chinese Control Conference*, pages 321–326, 2009.

[79] E. B. Dynkin and A. A. Yushkevich. *Controlled Markov processes*, volume 235 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1979. Translated from the Russian original by J. M. Danskin and C. Holland.

[80] E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5(Dec):1–25, 2003.

[81] E. Feinberg and A. Shwartz, editors. *Markov Decision Processes: Models, Methods, Directions, and Open Problems*. Kluwer Acad. Publ., Holland, 2001.

[82] E. A. Feinberg and A. Shwartz, editors. *Handbook of Markov decision processes*. International Series in Operations Research & Management Science, 40. Kluwer Academic Publishers, Boston, MA, 2002. Methods and applications.

[83] Y. Feng, L. Li, and Q. Liu. A kernel loss for solving the Bellman equation. In *Advances in Neural Information Processing Systems*, pages 15456–15467, 2019.

[84] A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 385–394, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[85] G. Fort, E. Moulines, S. P. Meyn, and P. Priouret. ODE methods for Markov chain stability with applications to MCMC. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 42, New York, NY, USA, 2006. ACM Press.

[86] K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In *Proceedings International Conference on Industrial Electronics, Control and Instrumentation*, pages 2193–2198. IEEE, 1991.

[87] P. A. Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[88] V. Gaitsgory, A. Parkinson, and I. Shvartsman. Linear programming formulations of deterministic infinite horizon optimal control problems in discrete time. *Discrete and Continuous Dynamical Systems - Series B*, 22(10):3821 – 3838, 2017.

[89] V. Gaitsgory and M. Quincampoix. On sets of occupational measures generated by a deterministic control system on an infinite time horizon. *Nonlinear Analysis: Theory, Methods and Applications*, 88:27 – 41, 2013.

[90] J. M. George and J. M. Harrison. Dynamic control of a queue with adjustable service rate. *Operations Res.*, 49(5):720–731, Sept. 2001.

[91] P. W. Glynn and S. P. Meyn. A Liapounov bound for solutions of the Poisson equation. *Ann. Probab.*, 24(2):916–931, 1996.

[92] G. J. Gordon. Reinforcement learning with function approximation converges to a region. In *Proc. of the 13th International Conference on Neural Information Processing Systems*, pages 996–1002, Cambridge, MA, USA, 2000. MIT Press.

[93] A. Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[94] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 1994.

[95] L. Greenemeier. AI versus AI: self-taught AlphaGo Zero vanquishes its predecessor. *Scientific American*, October 2017.

[96] A. Gupta, R. Jain, and P. W. Glynn. An empirical algorithm for relative value iteration for average-cost MDPs. In *IEEE Conference on Decision and Control*, pages 5079–5084, 2015.

[97] B. Hajek. *Random Processes for Engineers*. Cambridge University Press, 2015.

[98] H. V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[99] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag, second edition, 2001. Corr. 3rd printing, 2003.

[100] D. Hernández-Hernández, O. Hernández-Lerma, and M. Taksar. The linear programming approach to deterministic optimal control problems. *Applicationes Mathematicae*, 24(1):17–33, 1996.

[101] O. Hernández-Lerma and J. B. Lasserre. The linear programming approach. In *Handbook of Markov decision processes*, volume 40 of *Internat. Ser. Oper. Res. Management Sci.*, pages 377–407. Kluwer Acad. Publ., Boston, MA, 2002.

[102] O. Hernández-Lerma and J. B. Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

[103] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[104] D. Huang, W. Chen, P. Mehta, S. Meyn, and A. Surana. Feature selection for neuro-dynamic programming. In F. Lewis, editor, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley, 2011.

[105] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*, volume 44. Cambridge University Press, 2009.

[106] T. Jaakola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[107] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. *arXiv preprint arXiv:1907.05388*, 2019.

[108] G. L. Jones. On the Markov chain Central Limit Theorem. *Probab. Surv.*, 1:299–320 (electronic), 2004.

[109] R. E. Kalman. Contribution to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5:102–119, 1960.

[110] R. E. Kalman. When is a linear control system optimal? *Journal of Basic Engineering*, 86:51, 1964.

[111] A. Kamoutsi, T. Sutter, P. Mohajerin Esfahani, and J. Lygeros. On infinite linear programming and the moment approach to deterministic infinite horizon discounted optimal control problems. *IEEE Control Systems Letters*, 1(1):134–139, July 2017.

[112] P. Karmakar and S. Bhatnagar. Two time-scale stochastic approximation with controlled Markov noise and off-policy temporal-difference learning. *Math. Oper. Res.*, 43(1):130–151, 2018.

[113] H. K. Khalil. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition, 2002.

[114] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[115] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[116] V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4):1143–1166 (electronic), 2003.

[117] V. R. Konda and J. N. Tsitsiklis. Convergence rate of linear two-time-scale stochastic approximation. *Ann. Appl. Probab.*, 14(2):796–819, 2004.

[118] V. V. G. Konda. *Actor-critic algorithms*. PhD thesis, Massachusetts Institute of Technology, 2002.

[119] I. Kontoyiannis, L. A. Lastras-Montaño, and S. P. Meyn. Relative entropy and exponential deviation bounds for general Markov chains. In *Proc. of the IEEE International Symposium on Information Theory*, pages 1563–1567, Sept. 2005.

[120] I. Kontoyiannis and S. P. Meyn. Spectral theory and limit theorems for geometrically ergodic Markov processes. *Ann. Appl. Probab.*, 13:304–362, 2003.

[121] A. Krener. Feedback linearization. In *Mathematical control theory*, pages 66–98. Springer, 1999.

[122] V. Krishnamurthy. Structural results for partially observed Markov decision processes. *ArXiv e-prints*, page arXiv:1512.03873, 2015.

[123] M. Krstić and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595 – 601, 2000.

[124] H. J. Kushner and G. G. Yin. *Stochastic approximation algorithms and applications*, volume 35 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1997.

[125] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Wiley-Interscience, New York, NY, 1972.

[126] C. Lakshminarayanan and C. Szepesvari. Linear stochastic approximation: How far does constant step-size and iterate averaging go? In *International Conference on Artificial Intelligence and Statistics*, pages 1347–1355, 2018.

[127] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[128] B. Lapeybe, G. Pages, and K. Sab. Sequences with low discrepancy generalisation and application to Robbins-Monro algorithm. *Statistics*, 21(2):251–272, 1990.

[129] S. Laruelle and G. Pagès. Stochastic approximation with averaging innovation applied to finance. *Monte Carlo Methods and Applications*, 18(1):1–51, 2012.

[130] J.-B. Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2010.

[131] T. Lattimore and C. Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2020.

[132] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[133] S. Liu and M. Krstic. Introduction to extremum seeking. In *Stochastic Averaging and Stochastic Extremum Seeking*, Communications and Control Engineering. Springer, London, 2012.

[134] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.

[135] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

[136] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 719–726, USA, 2010. Omnipress.

[137] H. Mania, A. Guy, and B. Recht. Simple random search provides a competitive approach to reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

[138] A. S. Manne. Linear programming and sequential decisions. *Management Sci.*, 6(3):259–267, 1960.

[139] N. Matni, A. Proutiere, A. Rantzer, and S. Tu. From self-tuning regulators to reinforcement learning and back again. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3724–3740, 2019.

[140] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[141] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[142] P. G. Mehta and S. P. Meyn. Q-learning and Pontryagin's minimum principle. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3598–3605, Dec. 2009.

[143] P. G. Mehta and S. P. Meyn. Convex Q-learning, part 1: Deterministic optimal control. *ArXiv e-prints:2008.03559*, 2020.

[144] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 664–671, New York, NY, USA, 2008. ACM.

[145] M. Metivier and P. Priouret. Theoremes de convergence presque sure pour une classe d'algorithmes stochastiques a pas decroissants. *Prob. Theory Related Fields*, 74:403–428, 1987.

[146] S. P. Meyn. Large deviation asymptotics and control variates for simulating large functions. *Ann. Appl. Probab.*, 16(1):310–339, 2006.

[147] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007. Pre-publication edition available online.

[148] S. P. Meyn and G. Mathew. Shannon meets Bellman: Feature based Markovian models for detection and optimization. In *Proc. 47th IEEE CDC*, pages 5558–5564, 2008.

[149] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, Cambridge, second edition, 2009. Published in the Cambridge Mathematical Library. 1993 edition online.

[150] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. *Machine intelligence*, 2(2):137–152, 1968.

[151] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[152] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[153] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[154] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[155] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[156] A. W. Moore. *Efficient memory-based learning for robot control*. PhD thesis, University of Cambridge, Computer Laboratory, 1990.

[157] E. Moulines and F. R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24*, pages 451–459. Curran Associates, Inc., 2011.

[158] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[159] A. Nedic and D. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems: Theory and Applications*, 13(1-2):79–110, 2003.

[160] Y. Nesterov. *Lectures on Convex Optimization*. Springer Optimization and Its Applications. Springer International Publishing, 2018.

[161] J. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1997.

[162] E. Nummelin. *General Irreducible Markov Chains and Nonnegative Operators*. Cambridge University Press, Cambridge, 1984.

[163] D. Ormoneit and P. Glynn. Kernel-based reinforcement learning in average-cost problems. *IEEE Transactions on Automatic Control*, 47(10):1624–1636, Oct 2002.

[164] I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386, 2016.

[165] J. B. Park and J. Y. Lee. Nonlinear adaptive control based on Lyapunov analysis: Overview and survey. *Journal of Institute of Control, Robotics and Systems*, 20(3):261–269, 2014.

[166] B. T. Polyak. A new method of stochastic approximation type. *Avtomatika i telemekhanika (in Russian). translated in Automat. Remote Control, 51 (1991)*, pages 98–107, 1990.

[167] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, 1992.

[168] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[169] A. Ramaswamy and S. Bhatnagar. A generalization of the Borkar-Meyn Theorem for stochastic recursive inclusions. *Mathematics of Operations Research*, 42(3):648–661, 2017.

[170] A. Ramaswamy and S. Bhatnagar. Stability of stochastic approximations with 'controlled Markov' noise and temporal difference learning. *IEEE Transactions on Automatic Control*, pages 1–1, 2018.

[171] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[172] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report 166, Cambridge Univ., Dept. Eng., Cambridge, U.K. CUED/F-INENG/, 1994.

[173] D. Ruppert. A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 13(1):236–245, 1985.

[174] D. Ruppert. Efficient estimators from a slowly convergent Robbins-Monro processes. Technical Report Tech. Rept. No. 781, Cornell University, School of Operations Research and Industrial Engineering, Ithaca, NY, 1988.

[175] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. *A Tutorial on Thompson Sampling*. now, 2018.

[176] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *ArXiv*, abs/1911.08265, 2019.

[177] E. Seneta. *Non-Negative Matrices and Markov Chains*. Springer, New York, NY, 2nd edition, 1981.

[178] C. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[179] H. Sharma, R. Jain, and A. Gupta. An empirical relative value learning algorithm for non-parametric MDPs with continuous state space. In *European Control Conference*, pages 1368–1373. IEEE, 2019.

[180] S. D.-C. Shashua and S. Mannor. Kalman meets Bellman: Improving policy evaluation through value tracking. *arXiv preprint arXiv:2002.07171*, 2020.

[181] B. Shi, S. S. Du, W. Su, and M. I. Jordan. Acceleration via symplectic discretization of high-resolution differential equations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5744–5752. Curran Associates, Inc., 2019.

[182] S. Shirodkar and S. Meyn. Quasi stochastic approximation. In *Proc. of the 2011 American Control Conference (ACC)*, pages 2429–2435, July 2011.

[183] S. Shivam, I. Buckley, Y. Wardi, C. Seatzu, and M. Egerstedt. Tracking control by the newton-raphson flow: Applications to autonomous vehicles. *CoRR*, abs/1811.08033, 2018.

[184] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[185] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

[186] S. Smale. A convergent process of price adjustment and global Newton methods. *Journal of Mathematical Economics*, 3(2):107–120, July 1976.

[187] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.*, 21(5):1071–1088, Oct. 1973.

[188] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2003.

[189] M. W. Spong and D. J. Block. The pendubot: A mechatronic system for control research and education. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 1, pages 555–556. IEEE, 1995.

[190] M. W. Spong and L. Praly. Control of underactuated mechanical systems using switching and saturation. In *Control using logic-based switching*, pages 162–172. Springer, 1997.

[191] M. W. Spong and M. Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[192] R. Srikant and L. Ying. Finite-time error bounds for linear stochastic approximation and TD learning. *CoRR*, abs/1902.00923, 2019.

[193] W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in neural information processing systems*, pages 2510–2518, 2014.

[194] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press. Online edition at http://www.cs.ualberta.ca/~sutton/book/the-book.html, Cambridge, MA, 2nd edition, 2018.

[195] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, 1984.

[196] R. S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, 1988.

[197] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 1038–1044, Cambridge, MA, USA, 1995. MIT Press.

[198] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

[199] R. S. Sutton and A. G. Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review*, 88(2):135, 1981.

[200] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[201] R. S. Sutton, C. Szepesvári, and H. R. Maei. A convergent $o(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, pages 1609–1616, Red Hook, NY, USA, 2008. Curran Associates Inc.

[202] C. Szepesvári. The asymptotic convergence-rate of Q-learning. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, NIPS'97, pages 1064–1070, Cambridge, MA, USA, 1997. MIT Press.

[203] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[204] A. Tanzanakis and J. Lygeros. Data-driven control of unknown systems: A linear programming approach. *ArXiv*, abs/2003.00779, 2020.

[205] G. Thoppe and V. Borkar. A concentration bound for stochastic approximation via Alekseev's formula. *Stochastic Systems*, 9(1):1–26, 2019.

[206] J. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

[207] J. N. Tsitsiklis and B. V. Roy. Average cost temporal-difference learning. *Automatica*, 35(11):1799 – 1808, 1999.

[208] J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94, 1996.

[209] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Automat. Control*, 42(5):674–690, 1997.

[210] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, May 1997.

[211] J. N. Tsitsiklis and B. Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Trans. Automat. Control*, 44(10):1840–1851, 1999.

[212] L. Vandenberghe and S. Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 29(3):283 – 299, 1999. Proceedings of the Stieltjes Workshop on High Performance Optimization Techniques.

[213] J. Venter et al. An extension of the Robbins-Monro procedure. *The Annals of Mathematical Statistics*, 38(1):181–190, 1967.

[214] R. Vinter. Convex duality and nonlinear optimal control. *SIAM Journal on Control and Optimization*, 31(2):518–21, 03 1993.

[215] M. J. Wainwright. Stochastic approximation with cone-contractive operators: Sharp ℓ_∞-bounds for Q-learning. *CoRR*, abs/1905.06265, 2019.

[216] H.-H. Wang and M. Krstić. Extremum seeking for limit cycle minimization. *IEEE Transactions on Automatic Control*, 45(12):2432–2436, Dec 2000.

[217] Y. Wang and S. Boyd. Performance bounds for linear stochastic control. *Systems Control Lett.*, 58(3):178–182, 2009.

[218] Y. Wardi, C. Seatzu, M. Egerstedt, and I. Buckley. Performance regulation and tracking via lookahead simulation: Preliminary results and validation. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 6462–6468, 2017.

[219] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, Cambridge, UK, 1989.

[220] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[221] D. J. White. Real applications of markov decision processes. *Interfaces*, 15(6):73–83, 1985.

[222] D. J. White. A survey of applications of markov decision processes. *Journal of the operational research society*, 44(11):1073–1096, 1993.

[223] P. Whittle. *Risk-Sensitive Optimal Control*. John Wiley and Sons, Chichester, NY, 1990.

[224] A. Wibisono, A. C. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113:E7351 – E7358, 2016.

[225] V. G. Yaji and S. Bhatnagar. Stochastic recursive inclusions with non-additive iterate-dependent markov noise. *Stochastics*, 90(3):330–363, 2018.

[226] H. Yu and D. P. Bertsekas. Q-learning and policy iteration algorithms for stochastic shortest path problems. *Annals of Operations Research*, 208(1):95–132, 2013.

[227] J. Zhao and M. Spong. Hybrid control for global stabilization of the cart–pendulum system. *Automatica*, 37(12):1941 – 1951, 2001.

[228] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, 1996.

Index

Symbols and Notation