Contents

3 LQR **2**

3.1 Optimal control . . . 3

3.1.1 A first attempt: Discretization . . . 4

3.2 The Linear Quadratic Regulator . . . 5

3.3 Optimality and the Riccati Equation . . . 6

3.3.1 Expected state at time h . . . 10

3.4 Extensions . . . 11

3.4.1 Time-dependency . . . 11

3.4.2 General quadratic cost . . . 12

3.4.3 Tracking a predefined trajectory . . . 12

3.5 Approximating nonlinear dynamics . . . 12

3.5.1 Local linearization . . . 13

3.5.2 Finite differencing . . . 14

3.5.3 Local convexification . . . 14

3.5.4 Iterative LQR . . . 15

3.6 Summary . . . 17

Chapter 3

Linear Quadratic Regulators

Up to this point, we have considered decision problems with finitely many states and actions. However, in many applications, states and actions may take on continuous values. For example, consider autonomous driving, controlling a robot's joints, and automated manufacturing. How can we teach computers to solve these kinds of problems? This is the task of **continuous control**.

(a) Solving a Rubik's Cube with a robot hand.

(b) Boston Dynamics's Spot robot.

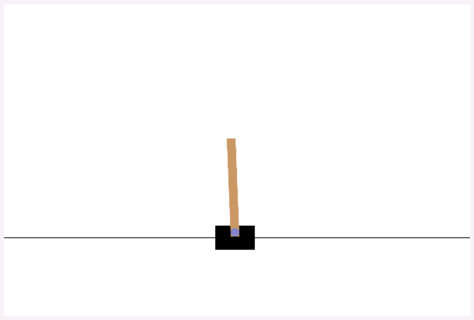
Figure 3.1: Examples of control tasks.

Aside from the change in the state and action spaces, the general problem setup remains the same: we seek to construct an *optimal policy* that outputs actions to solve the desired task. We will see that many key ideas and algorithms, in particular dynamic programming algorithms, carry over to this new setting.

This chapter introduces a fundamental tool to solve a simple class of continuous control problems: the **linear quadratic regulator**. We will then extend this basic method to more complex settings.

Example 3.0.1: CartPole

Try to balance a pencil on its point on a flat surface. It's much more difficult than it may first seem: the position of the pencil varies continuously, and the state transitions governing the system, i.e. the laws of physics, are highly complex. This task is equivalent to the classic control problem known as *CartPole*:



The state $s \in \mathbb{R}^4$ can be described by:

1. the position of the cart;
2. the velocity of the cart;
3. the angle of the pole;
4. the angular velocity of the pole.

We can *control* the cart by applying a horizontal force $a \in \mathbb{R}$.

Goal: Stabilize the cart around an ideal state and action $(s^\star, a^\star)$.

3.1 Optimal control

Recall that an MDP is defined by its state space $\mathcal{S}$, action space $\mathcal{A}$, state transitions P, reward function r, and discount factor γ or time horizon H. These have equivalents in the control setting:

- The state and action spaces are *continuous* rather than finite. That is, $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{n_a}$, where n_s and n_a are the corresponding dimensions of these spaces, i.e. the number of coordinates to specify a single state or action respectively.
- We call the state transitions the **dynamics** of the system. In the most general case, these might change across timesteps and also include some stochastic **noise** w_h at each timestep. We denote these dynamics as the function f_h such that $s_{h+1} = f_h(s_h, a_h, w_h)$. Of course, we can simplify to cases where the dynamics are *deterministic/noise-free* (no w_h term) and/or *time-homogeneous* (the same function f across timesteps).
- Instead of maximizing the reward function, we seek to minimize the **cost function** c_h :

$\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Often, the cost function describes *how far away* we are from a **target state-action pair** $(s^\star, a^\star)$. An important special case is when the cost is *time-homogeneous*; that is, it remains the same function c at each timestep h.

- We seek to minimize the *undiscounted* cost within a *finite time horizon* H. Note that we end an episode at the final state s_H – there is no a_H, and so we denote the cost for the final state as $c_H(s_H)$.

With all of these components, we can now formulate the **optimal control problem:** *compute a policy to minimize the expected undiscounted cost over H timesteps.* In this chapter, we will only consider *deterministic, time-dependent* policies $\pi = (\pi_0, \dots, \pi_{H-1})$ where $\pi_h : \mathcal{S} \to \mathcal{A}$ for each $h \in [H]$.

Definition 3.1.1: General optimal control problem

$$\begin{aligned}
\min_{\pi_0,\dots,\pi_{H-1}:\mathcal{S}\to\mathcal{A}} \quad & \mathbb{E}\left[\left(\sum_{h=0}^{H-1} c_h(s_h, a_h)\right) + c_H(s_H)\right] \\
\text{where} \quad & s_{h+1} = f_h(s_h, a_h, w_h), \\
& a_h = \pi_h(s_h) \\
& s_0 \sim \mu_0 \\
& w_h \sim \text{noise}
\end{aligned} \tag{3.1}$$

3.1.1 A first attempt: Discretization

Can we solve this problem using tools from the finite MDP setting? If $\mathcal{S}$ and $\mathcal{A}$ were finite, then we'd be able to work backwards using the DP algorithm for computing the optimal policy in an MDP (**??**). This inspires us to try *discretizing* the problem.

Suppose $\mathcal{S}$ and $\mathcal{A}$ are bounded, that is, $\max_{s\in\mathcal{S}} \|s\| \le B_s$ and $\max_{a\in\mathcal{A}} \|a\| \le B_a$. To make $\mathcal{S}$ and $\mathcal{A}$ finite, let's choose some small positive ϵ, and simply round each coordinate to the nearest multiple of ϵ. For example, if $\epsilon = 0.01$, then we round each element of s and a to two decimal spaces.

However, the discretized $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{A}}$ may be finite, but they may be infeasibly large: we must divide *each dimension* into intervals of length ε, resulting in $|\widetilde{\mathcal{S}}| = (B_s/\varepsilon)^{n_s}$ and $|\widetilde{\mathcal{A}}| = (B_a/\varepsilon)^{n_a}$. To get a sense of how quickly this grows, consider $\varepsilon = 0.01, n_s = n_a = 10$. Then the number of elements in the transition matrix would be $|\widetilde{\mathcal{S}}|^2|\widetilde{\mathcal{A}}| = (100^{10})^2(100^{10}) = 10^{60}$! (That's a trillion trillion trillion trillion trillion.)

What properties of the problem could we instead make use of? Note that by discretizing the state and action spaces, we implicitly assumed that rounding each state or action vector by some tiny amount ε wouldn't change the behavior of the system by much; namely, that the cost and dynamics were relatively *continuous*. Can we use this continuous structure in other ways? This leads us to the **linear quadratic regulator**.

3.2 The Linear Quadratic Regulator

The optimal control problem (3.1.1) seems highly complex in its general case. Is there a relevant simplification that we can analyze?

Let us consider *linear dynamics* and an *upward-curved quadratic cost function* (in both arguments). We will also consider a time-homogenous cost function that targets $(s^\star, a^\star) = (0, 0)$. This model is called the **linear quadratic regulator** (LQR) and is a fundamental tool in control theory. Solving the LQR problem will additionally enable us to *locally approximate* more complex setups using *Taylor approximations*.

Definition 3.2.1: The linear quadratic regulator

Linear, time-homogeneous dynamics:

$$s_{h+1} = f(s_h, a_h, w_h) = As_h + Ba_h + w_h$$

Upward-curved quadratic, time-homogeneous cost function:

$$c(s_h, a_h) = \begin{cases} s_h^\top Q s_h + a_h^\top R a_h & h < H \\ s_h^\top Q s_h & h = H \end{cases}$$

We require Q and R to both be positive definite matrices so that c has a well-defined unique minimum. We can furthermore assume without loss of generality that they are both symmetric (see exercise below).

Spherical Gaussian noise:

$$w_h \sim \mathcal{N}(0, \sigma^2 I) \quad \forall h \in [H]$$

This results in the LQR optimization problem:

$$\begin{aligned} \min_{\pi_0, \dots, \pi_{H-1} : \mathcal{S} \to \mathcal{A}} \quad & \mathbb{E}\left[\left(\sum_{h=0}^{H-1} s_h^\top Q s_h + a_h^\top R a_h \right) + s_H^\top Q s_H \right] \\ \text{where} \quad & s_{h+1} = As_h + Ba_h + w_h \\ & a_h = \pi_h(s_h) \\ & w_h \sim \mathcal{N}(0, \sigma^2 I) \\ & s_0 \sim \mu_0. \end{aligned}$$

Exercise: We've set Q and R to be *symmetric* positive definite (SPD) matrices. Here we'll show that the symmetry condition can be imposed without loss of generality. Show that replacing Q with $(Q + Q^\top)/2$ (which is symmetric) yields the same cost function.

It will be helpful to reintroduce the *value function* notation for a policy to denote the average cost it incurs. These will be instrumental in constructing the optimal policy via **dynamic programming**.

Definition 3.2.2: Value functions for LQR

Given a policy $\pi = (\pi_0, \dots, \pi_{H-1})$, we can define its value function $V_h^\pi : \mathcal{S} \to \mathbb{R}$ at time $h \in [H]$ as the average **cost-to-go** incurred by that policy:

$$\begin{aligned} V_h^\pi(s) &= \mathbb{E}\left[\left(\sum_{i=h}^{H-1} c(s_i, a_i)\right) + c(s_H) \mid s_h = s, a_i = \pi_i(s_i) \quad \forall h \le i < H\right] \\ &= \mathbb{E}\left[\left(\sum_{i=h}^{H-1} s_i^\top Q s_i + a_i^\top R a_i\right) + s_H^\top Q s_H \mid s_h = s, a_i = \pi_i(s_i) \quad \forall h \le i < H\right] \end{aligned}$$

The Q-function additionally conditions on the first action we take:

$$\begin{aligned} Q_h^\pi(s, a) &= \mathbb{E}\Bigg[\left(\sum_{i=h}^{H-1} c(s_i, a_i)\right) + c(s_H) \\ &\qquad \mid (s_h, a_h) = (s, a), a_i = \pi_i(s_i) \quad \forall h \le i < H\Bigg] \\ &= \mathbb{E}\Bigg[\left(\sum_{i=h}^{H-1} s_i^\top Q s_i + a_i^\top R a_i\right) + s_H^\top Q s_H \\ &\qquad \mid (s_h, a_h) = (s, a), a_i = \pi_i(s_i) \quad \forall h \le i < H\Bigg] \end{aligned}$$

3.3 Optimality and the Riccati Equation

In this section, we'll compute the optimal value function $V_h^\star$, Q-function $Q_h^\star$, and policy $\pi_h^\star$ in the LQR setting using **dynamic programming** in a very similar way to the DP algorithms in the MDP setting (**??**):

1. We'll compute $V_H^\star$ (at the end of the horizon) as our base case.
2. Then we'll work backwards in time, using $V_{h+1}^\star$ to compute $Q_h^\star$, $\pi_h^\star$, and $V_h^\star$.

Along the way, we will prove the striking fact that the solution has very simple structure: $V_h^\star$ and $Q_h^\star$ are *upward-curved quadratics* and $\pi_h^\star$ is *linear*.

Definition 3.3.1: Optimal value functions for LQR

The **optimal value function** is the one that, at any time and in any state, achieves *minimum cost* across *all policies*:

$$\begin{aligned} V_h^\star(s) &= \min_{\pi_h, \dots, \pi_{H-1}} V_h^\pi(s) \\ &= \min_{\pi_h, \dots, \pi_{H-1}} \mathbb{E} \Bigg[\left(\sum_{i=h}^{H-1} s_h^\top Q s_h + a_h^\top R a_h \right) + s_H^\top Q s_H \\ &\qquad\qquad \mid s_h = s, a_i = \pi_i(s_i) \quad \forall h \le i < H \Bigg] \end{aligned}$$

Theorem 3.3.1: Optimal value function in LQR is a upward-curved quadratic

At each timestep $h \in [H]$,

$$V_h^\star(s) = s^\top P_h s + p_h$$

for some symmetric positive definite matrix $P_h \in \mathbb{R}^{n_s \times n_s}$ and vector $p_h \in \mathbb{R}^{n_s}$.

Theorem 3.3.2: Optimal policy in LQR is linear

At each timestep $h \in [H]$,

$$\pi_h^\star(s) = -K_h s$$

for some $K_h \in \mathbb{R}^{n_a \times n_s}$. (The negative is due to convention.)

Base case: At the final timestep, there are no possible actions to take, and so $V_H^\star(s) = c(s) = s^\top Q s$. Thus $V_H^\star(s) = s^\top P_h s + p_h$ where $P_H = Q$ and p_H is the zero vector.

Inductive hypothesis: We seek to show that the inductive step holds for both theorems: If $V_{h+1}^\star(s)$ is a upward-curved quadratic, then $V_h^\star(s)$ must also be a upward-curved quadratic, and $\pi_h^\star(s)$ must be linear. We'll break this down into the following steps:

Step 1. Show that $Q_h^\star(s, a)$ is a upward-curved quadratic (in both s and a).

Step 2. Derive the optimal policy $\pi_h^\star(s) = \arg\min_a Q_h^\star(s, a)$ and show that it's linear.

Step 3. Show that $V_h^\star(s)$ is a upward-curved quadratic.

We first assume the inductive hypothesis that our theorems are true at time $h+1$. That is,

$$V_{h+1}^\star(s) = s^\top P_{h+1} s + p_{h+1} \quad \forall s \in \mathcal{S}.$$

Step 1. We aim to show that $Q_h^\star(s)$ is a upward-curved quadratic. Recall that the definition of $Q_h^\star : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is

$$Q_h^\star(s, a) = c(s, a) + \mathop{\mathbb{E}}_{s' \sim f(s, a, w_{h+1})} [V_{h+1}^\star(s')].$$

Recall $c(s, a) = s^\top Q s + a^\top R a$. Let's consider the average value over the next timestep. The only randomness in the dynamics comes from the noise $w_{h+1} \sim \mathcal{N}(0, \sigma^2 I)$, so we can write out this expected value as:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s'}[V^\star_{h+1}(s')] \\
&= \mathop{\mathbb{E}}_{w_{h+1}}[V^\star_{h+1}(As + Ba + w_{h+1})] && \text{definition of } f \\
&= \mathop{\mathbb{E}}_{w_{h+1}}[(As + Ba + w_{h+1})^\top P_{h+1}(As + Ba + w_{h+1}) + p_{h+1}]. && \text{inductive hypothesis}
\end{aligned}$$

Summing and combining like terms, we get

$$\begin{aligned}
Q^\star_h(s, a) &= s^\top Q s + a^\top R a + \mathop{\mathbb{E}}_{w_{h+1}}[(As + Ba + w_{h+1})^\top P_{h+1}(As + Ba + w_{h+1}) + p_{h+1}] \\
&= s^\top(Q + A^\top P_{h+1} A)s + a^\top(R + B^\top P_{h+1} B)a + 2s^\top A^\top P_{h+1} B a \\
&\qquad + \mathop{\mathbb{E}}_{w_{h+1}}[w_{h+1}^\top P_{h+1} w_{h+1}] + p_{h+1}.
\end{aligned}$$

Note that the terms that are linear in w_h have mean zero and vanish. Now consider the remaining expectation over the noise. By expanding out the product and using linearity of expectation, we can write this out as

$$\mathop{\mathbb{E}}_{w_{h+1}}[w_{h+1}^\top P_{h+1} w_{h+1}] = \sum_{i=1}^d \sum_{j=1}^d (P_{h+1})_{ij} \mathop{\mathbb{E}}_{w_{h+1}}[(w_{h+1})_i (w_{h+1})_j].$$

When dealing with these *quadratic forms*, it's often helpful to consider the terms on the diagonal ($i = j$) separately from those off the diagonal. On the diagonal, the expectation becomes

$$(P_{h+1})_{ii} \mathbb{E}(w_{h+1})_i^2 = \sigma^2 (P_{h+1})_{ii}.$$

Off the diagonal, since the elements of w_{h+1} are independent, the expectation factors, and since each element has mean zero, the term disappears:

$$(P_{h+1})_{ij} \mathbb{E}[(w_{h+1})_i] \mathbb{E}[(w_{h+1})_j] = 0.$$

Thus, the only terms left are the ones on the diagonal, so the sum of these can be expressed as the trace of $\sigma^2 P_{h+1}$:

$$\mathop{\mathbb{E}}_{w_{h+1}}[w_{h+1}^\top P_{h+1} w_{h+1}] = \mathrm{Tr}(\sigma^2 P_{h+1}).$$

Substituting this back into the expression for $Q^\star_h$, we have:

$$\boxed{\begin{aligned} Q^\star_h(s, a) = s^\top(Q + A^\top P_{h+1} A)s + a^\top(R + B^\top P_{h+1} B)a + 2s^\top A^\top P_{h+1} B a \\ + \mathrm{Tr}(\sigma^2 P_{h+1}) + p_{h+1}. \end{aligned}} \tag{3.2}$$

As we hoped, this expression is quadratic in s and a. Furthermore, we'd like to show that it also has *positive curvature* with respect to a so that its minimum with respect to a is well-defined. We can do this by proving that the **Hessian matrix** of second derivatives is positive definite:

$$\nabla_{aa} Q^\star_h(x, u) = R + B^\top P_{h+1} B$$

This is fairly straightforward: recall that in our definition of LQR, we assumed that R is SPD (see Definition 3.2.1). Also note that since P_{h+1} is SPD (by the inductive hypothesis), so too must be $B^\top P_{h+1} B$. (If this isn't clear, try proving it as an exercise.) Since the sum of two SPD matrices is also SPD, we have that $R + B^\top P_{h+1} B$ is SPD, and so $Q^\star_h$ is indeed a upward-curved quadratic with respect to a.

Step 2. Now we aim to show that $\pi^\star_h$ is linear. Since $Q^\star_h$ is a upward-curved quadratic, finding its minimum over a is easy: we simply set the gradient with respect to a equal to zero and solve for a. First, we calculate the gradient:

$$\begin{aligned}\nabla_a Q^\star_h(s, a) &= \nabla_a [a^\top (R + B^\top P_{h+1} B) a + 2 s^\top A^\top P_{h+1} B a] \\ &= 2(R + B^\top P_{h+1} B) a + 2 (s^\top A^\top P_{h+1} B)^\top\end{aligned}$$

Setting this to zero, we get

$$\begin{aligned}0 &= (R + B^\top P_{h+1} B) \pi^\star_h(s) + B^\top P_{h+1} A s \\ \pi^\star_h(s) &= (R + B^\top P_{h+1} B)^{-1} (-B^\top P_{h+1} A s) \\ &= - K_h s, \qquad (3.3)\end{aligned}$$

where $K_h = (R + B^\top P_{h+1} B)^{-1} B^\top P_{h+1} A$. Note that this optimal policy doesn't depend on the starting distribution μ_0. It's also fully **deterministic** and isn't affected by the noise terms $w_0, \dots, w_{H-1}$.

Step 3. To complete our inductive proof, we must show that the inductive hypothesis is true at time h; that is, we must prove that $V^\star_h(s)$ is a upward-curved quadratic. Using the identity $V^\star_h(s) = Q^\star_h(s, \pi^\star(s))$, we have:

$$\begin{aligned}V^\star_h(s) &= Q^\star_h(s, \pi^\star(s)) \\ &= s^\top (Q + A^\top P_{h+1} A) s + (-K_h s)^\top (R + B^\top P_{h+1} B)(-K_h s) + 2 s^\top A^\top P_{h+1} B (-K_h s) \\ &\qquad + \mathrm{Tr}(\sigma^2 P_{h+1}) + p_{h+1}\end{aligned}$$

Note that with respect to s, this is the sum of a quadratic term and a constant, which is exactly what we were aiming for! The constant term is clearly $p_h = \mathrm{Tr}(\sigma^2 P_{h+1}) + p_{h+1}$. We can simplify the quadratic term by substituting in K_h. Notice that when we do this, the $(R + B^\top P_{h+1} B)$ term in the expression is cancelled out by its inverse, and the remaining terms combine to give the **Riccati equation**:

Definition 3.3.2: Riccati equation

$$P_h = Q + A^\top P_{h+1} A - A^\top P_{h+1} B (R + B^\top P_{h+1} B)^{-1} B^\top P_{h+1} A.$$

There are several nice properties to note about the Riccati equation:

1. It's defined **recursively.** Given the dynamics defined by A and B, and the state cost matrix Q, we can recursively calculate P_h across all timesteps starting from $P_H = Q$.
2. P_h often appears in calculations surrounding optimality, such as $V^\star_h, Q^\star_h$, and $\pi^\star_h$.

3. Together with the dynamics given by A and B, and the action coefficients R, it fully defines the optimal policy.

Now we've shown that $V^\star_h(s) = s^\top P_h s + p_h$, which is a upward-curved quadratic, and this concludes our proof. ■

In summary, we just demonstrated that at each timestep $h \in H$, the optimal value function $V^\star_h$ and optimal Q-function $Q^\star_h$ are both upward-curved quadratics and the optimal policy $\pi^\star_h$ is linear. We also showed that all of these quantities can be calculated using a sequence of symmetric matrices $P_0, \dots, P_H$ that can be defined recursively using the Riccati equation (3.3.2).

Before we move on to some extensions of LQR, let's consider how the state at time h behaves when we act according to this optimal policy.

3.3.1 Expected state at time h

How can we compute the expected state at time h when acting according to the optimal policy? Let's first express s_h in a cleaner way in terms of the history. Note that having linear dynamics makes it easy to expand terms backwards in time:

$$\begin{aligned}
s_h &= A s_{h-1} + B a_{h-1} + w_{h-1} \\
&= A(A s_{h-2} + B a_{h-2} + w_{h-2}) + B a_{h-1} + w_{h-1} \\
&= \cdots \\
&= A^h s_0 + \sum_{i=0}^{h-1} A^i (B a_{h-i-1} + w_{h-i-1}).
\end{aligned}$$

Let's consider the *average state* at this time, given all the past states and actions. Since we assume that $\mathbb{E}[w_h] = 0$ (this is the zero vector in d dimensions), when we take an expectation, the w_h term vanishes due to linearity, and so we're left with

$$\mathbb{E}[s_h \mid s_{0:(h-1)}, a_{0:(h-1)}] = A^h s_0 + \sum_{i=0}^{h-1} A^i B a_{h-i-1}.$$

If we choose actions according to our optimal policy, this becomes

$$\mathbb{E}[s_h \mid s_0, a_i = -K_i s_i \quad \forall i \le h] = \left(\prod_{i=0}^{h-1} (A - B K_i) \right) s_0.$$

Exercise: Verify this.

This introdces the quantity $A - BK_i$, which shows up frequently in control theory. For example, one important question is: will s_h remain bounded, or will it go to infinity as time goes on? To answer this, let's imagine for simplicity that these K_is are equal (call this matrix K). Then the expression above becomes $(A - BK)^h s_0$. Now consider the maximum eigenvalue $\lambda_{\max}$ of $A - BK$. If $|\lambda_{\max}| > 1$, then there's some nonzero initial state $\bar{s}_0$, the corresponding eigenvector, for which

$$\lim_{h \to \infty} (A - BK)^h \bar{s}_0 = \lim_{h \to \infty} \lambda_{\max}^h \bar{s}_0 = \infty.$$

Otherwise, if $|\lambda_{\max}| < 1$, then it's impossible for your original state to explode as dramatically.

3.4 Extensions

We've now formulated an optimal solution for the time-homogeneous LQR and computed the expected state under the optimal policy. However, real world tasks rarely have such simple dynamics, and we may wish to design more complex cost functions. In this section, we'll consider more general extensions of LQR where some of the assumptions we made above are relaxed. Specifically, we'll consider:

1. **Time-dependency**, where the dynamics and cost function might change depending on the timestep.
2. **General quadratic cost**, where we allow for linear terms and a constant term.
3. **Tracking a goal trajectory** rather than aiming for a single goal state-action pair.

Combining these will allow us to use the LQR solution to solve more complex setups by taking *Taylor approximations* of the dynamics and cost functions.

3.4.1 Time-dependent dynamics and cost function

So far, we've considered the *time-homogeneous* case, where the dynamics and cost function stay the same at every timestep. However, this might not always be the case. As an example, in many sports, the rules and scoring system might change during an overtime period. To address these sorts of problems, we can loosen the time-homogeneous restriction, and consider the case where the dynamics and cost function are *time-dependent.* Our analysis remains almost identical; in fact, we can simply add a time index to the matrices A and B that determine the dynamics and the matrices Q and R that determine the cost.

Exercise: Walk through the above derivation to verify this claim.

The modified problem is now defined as follows:

Definition 3.4.1: Time-dependent LQR

$$\begin{aligned}
\min_{\pi_0, \dots, \pi_{H-1}} \quad & \mathbb{E} \left[\left(\sum_{h=0}^{H-1} (s_h^\top Q_h s_h) + a_h^\top R_h a_h \right) + s_H^\top Q_H s_H \right] \\
\text{where} \quad & s_{h+1} = f_h(s_h, a_h, w_h) = A_h s_h + B_h a_h + w_h \\
& s_0 \sim \mu_0 \\
& a_h = \pi_h(s_h) \\
& w_h \sim \mathcal{N}(0, \sigma^2 I).
\end{aligned}$$

The derivation of the optimal value functions and the optimal policy remains almost exactly the same, and we can modify the Riccati equation accordingly:

Definition 3.4.2: Time-dependent Riccati Equation

$$P_h = Q_h + A_h^\top P_{h+1} A_h - A_h^\top P_{h+1} B_h (R_h + B_h^\top P_{h+1} B_h)^{-1} B_h^\top P_{h+1} A_h.$$

Note that this is just the time-homogeneous Riccati equation (Definition 3.3.2), but with the time index added to each of the relevant matrices.

Additionally, by allowing the dynamics to vary across time, we gain the ability to *locally approximate* nonlinear dynamics at each timestep. We'll discuss this later in the chapter.

3.4.2 More general quadratic cost functions

Our original cost function had only second-order terms with respect to the state and action, incentivizing staying as close as possible to $(s^\star, a^\star) = (0, 0)$. We can also consider more general quadratic cost functions that also have first-order terms and a constant term. Combining this with time-dependent dynamics results in the following expression, where we introduce a new matrix M_h for the cross term, linear coefficients q_h and r_h for the state and action respectively, and a constant term c_h:

$$c_h(s_h, a_h) = (s_h^\top Q_h s_h + s_h^\top M_h a_h + a_h^\top R_h a_h) + (s_h^\top q_h + a_h^\top r_h) + c_h. \tag{3.4}$$

Similarly, we can also include a constant term $v_h \in \mathbb{R}^{n_s}$ in the dynamics (note that this is *deterministic* at each timestep, unlike the stochastic noise w_h):

$$s_{h+1} = f_h(s_h, a_h, w_h) = A_h s_h + B_h a_h + v_h + w_h.$$

Exercise: Derive the optimal solution. (You will need to slightly modify the above proof.)

3.4.3 Tracking a predefined trajectory

Consider applying LQR to a task like autonomous driving, where the target state-action pair changes over time. We might want the vehicle to follow a predefined *trajectory* of states and actions $(s_h^\star, a_h^\star)_{h=0}^{H-1}$. To express this as a control problem, we'll need a corresponding time-dependent cost function:

$$c_h(s_h, a_h) = (s_h - s_h^\star)^\top Q (s_h - s_h^\star) + (a_h - a_h^\star)^\top R (a_h - a_h^\star).$$

Note that this punishes states and actions that are far from the intended trajectory. By expanding out these multiplications, we can see that this is actually a special case of the more general quadratic cost function above (Equation 3.4):

$$M_h = 0, \qquad q_h = -2Q s_h^\star, \qquad r_h = -2R a_h^\star, \qquad c_h = (s_h^\star)^\top Q (s_h^\star) + (a_h^\star)^\top R (a_h^\star).$$

3.5 Approximating nonlinear dynamics

The LQR algorithm solves for the optimal policy when the dynamics are *linear* and the cost function is an *upward-curved quadratic*. However, real settings are rarely this simple! Let's return to the CartPole example from the start of the chapter (Example 3.0.1). The dynamics (physics) aren't linear. How can we approximate this by an LQR problem?

Concretely, let's consider a *noise-free* problem since, as we saw, the noise doesn't factor into the optimal policy. Let's assume the dynamics and cost function are stationary, and ignore the terminal state for simplicity:

Definition 3.5.1: Nonlinear control problem

$$\begin{aligned}
\min_{\pi_0, \dots, \pi_{H-1} : \mathcal{S} \to \mathcal{A}} \quad & \mathop{\mathbb{E}}_{s_0} \left[\sum_{h=0}^{H-1} c(s_h, a_h) \right] \\
\text{where} \quad & s_{h+1} = f(s_h, a_h) \\
& a_h = \pi_h(s_h) \\
& s_0 \sim \mu_0 \\
& c(s, a) = d(s, s^\star) + d(a, a^\star).
\end{aligned}$$

Here, d denotes a function that measures the "distance" between its two arguments.

This is now only slightly simplified from the general optimal control problem (see 3.1.1). Here, we don't know an analytical form for the dynamics f or the cost function c, but we assume that we're able to *query/sample/simulate* them to get their values at a given state and action. To clarify, consider the case where the dynamics are given by real world physics. We can't (yet) write down an expression for the dynamics that we can differentiate or integrate analytically. However, we can still *simulate* the dynamics and cost function by running a real-world experiment and measuring the resulting states and costs. How can we adapt LQR to this more general nonlinear case?

3.5.1 Local linearization

How can we apply LQR when the dynamics are nonlinear or the cost function is more complex? We'll exploit the useful fact that we can take a function that's *locally continuous* around $(s^\star, a^\star)$ and approximate it nearby with low-order polynomials (i.e. its Taylor approximation). In particular, as long as the dynamics f are differentiable around $(s^\star, a^\star)$ and the cost function c is twice differentiable at $(s^\star, a^\star)$, we can take a linear approximation of f and a quadratic approximation of c to bring us back to the regime of LQR.

Linearizing the dynamics around $(s^\star, a^\star)$ gives:

$$f(s, a) \approx f(s^\star, a^\star) + \nabla_s f(s^\star, a^\star)(s - s^\star) + \nabla_a f(s^\star, a^\star)(a - a^\star)$$

$$(\nabla_s f(s, a))_{ij} = \frac{d f_i(s, a)}{d s_j}, \quad i, j \le n_s \qquad (\nabla_a f(s, a))_{ij} = \frac{d f_i(s, a)}{d a_j}, \quad i \le n_s, j \le n_a$$

and quadratizing the cost function around $(s^\star, a^\star)$ gives:

$$c(s, a) \approx c(s^\star, a^\star) \quad \text{constant term}$$

$$
\begin{aligned}
&+ \nabla_s c(s^\star, a^\star)(s - s^\star) + \nabla_a c(s^\star, a^\star)(a - a^\star) \quad \text{linear terms} \\
&\left.\begin{aligned}
&+ \frac{1}{2}(s - s^\star)^\top \nabla_{ss} c(s^\star, a^\star)(s - s^\star) \\
&+ \frac{1}{2}(a - a^\star)^\top \nabla_{aa} c(s^\star, a^\star)(a - a^\star) \\
&+ (s - s^\star)^\top \nabla_{sa} c(s^\star, a^\star)(a - a^\star)
\end{aligned}\right\} \text{quadratic terms}
\end{aligned}
$$

where the gradients and Hessians are defined as

$$
\begin{aligned}
(\nabla_s c(s, a))_i &= \frac{dc(s, a)}{ds_i}, \quad i \le n_s & (\nabla_a c(s, a))_i &= \frac{dc(s, a)}{da_i}, \quad i \le n_a \\
(\nabla_{ss} c(s, a))_{ij} &= \frac{d^2 c(s, a)}{ds_i ds_j}, \quad i, j \le n_s & (\nabla_{aa} c(s, a))_{ij} &= \frac{d^2 c(s, a)}{da_i da_j}, \quad i, j \le n_a \\
(\nabla_{sa} c(s, a))_{ij} &= \frac{d^2 c(s, a)}{ds_i da_j}. \quad i \le n_s, j \le n_a
\end{aligned}
$$

Exercise: Note that this cost can be expressed in the general quadratic form seen in Equation 3.4. Derive the corresponding quantities Q, R, M, q, r, c.

3.5.2 Finite differencing

To calculate these gradients and Hessians in practice, we use a method known as **finite differencing** for numerically computing derivatives. Namely, we can simply use the limit definition of the derivative, and see how the function changes as we add or subtract a tiny δ to the input.

$$
\frac{d}{dx} f(x) = \lim_{\delta \to 0} \frac{f(x + \delta) - f(x)}{\delta}
$$

Note that this only requires us to be able to *query* the function, not to have an analytical expression for it, which is why it's so useful in practice.

3.5.3 Local convexification

However, simply taking the second-order approximation of the cost function is insufficient, since for the LQR setup we required that the Q and R matrices were positive definite, i.e. that all of their eigenvalues were positive.

One way to naively *force* some symmetric matrix D to be positive definite is to set any non-positive eigenvalues to some small positive value $\varepsilon > 0$. Recall that any real symmetric matrix $D \in \mathbb{R}^{n \times n}$ has an basis of eigenvectors $u_1, \dots, u_n$ with corresponding eigenvalues $\lambda_1, \dots, \lambda_n$ such that $Du_i = \lambda_i u_i$. Then we can construct the positive definite approximation by

$$
\widetilde{D} = \left(\sum_{i=1,\dots,n \mid \lambda_i > 0} \lambda_i u_i u_i^\top \right) + \varepsilon I.
$$

Exercise: Convince yourself that $\widetilde{D}$ is indeed positive definite.

Note that Hessian matrices are generally symmetric, so we can apply this process to Q and R to obtain the positive definite approximations $\widetilde{Q}$ and $\widetilde{R}$. Now that we have a upward-curved quadratic approximation to the cost function, and a linear approximation to the state transitions, we can simply apply the time-homogenous LQR methods from section 3.3.

But what happens when we enter states far away from $s^\star$ or want to use actions far from $a^\star$? A Taylor approximation is only accurate in a *local* region around the point of linearization, so the performance of our LQR controller will degrade as we move further away. We'll see how to address this in the next section using the **iterative LQR** algorithm.

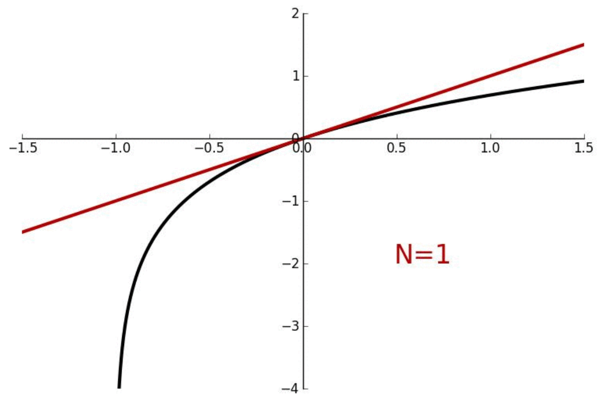


Figure 3.2: Local linearization might only be accurate in a small region around the point of linearization.

3.5.4 Iterative LQR

To address these issues with local linearization, we'll use an iterative approach, where we repeatedly linearize around different points to create a *time-dependent* approximation of the dynamics, and then solve the resulting time-dependent LQR problem to obtain a better policy. This is known as **iterative LQR** or **iLQR**:

Definition 3.5.2: Iterative LQR (high-level)

For each iteration of the algorithm:

Step 1. Form a time-dependent LQR problem around the current candidate trajectory using local linearization.

Step 2. Compute the optimal policy using subsection 3.4.1.

Step 3. Generate a new series of actions using this policy.

Step 4. Compute a better candidate trajectory by interpolating between the current and proposed actions.

Now let's go through the details of each step. We'll use superscripts to denote the iteration of the algorithm. We'll also denote $\bar{s}_0 = \mathbb{E}_{s_0 \sim \mu_0}[s_0]$ as the expected initial state.

At iteration i of the algorithm, we begin with a **candidate** trajectory $\bar{\tau}^i = (\bar{s}^i_0, \bar{a}^i_0, \dots, \bar{s}^i_{H-1}, \bar{a}^i_{H-1})$.

Step 1: Form a time-dependent LQR problem. At each timestep $h \in [H]$, we use the techniques from section 3.5 to linearize the dynamics and quadratize the cost function around $(\bar{s}^i_h, \bar{a}^i_h)$:

$$\begin{aligned} f_h(s,a) &\approx f(\bar{s}^i_h, \bar{a}^i_h) + \nabla_s f(\bar{s}^i_h, \bar{a}^i_h)(s - \bar{s}^i_h) + \nabla_a f(\bar{s}^i_h, \bar{a}^i_h)(a - \bar{a}^i_h) \\ c_h(s,a) &\approx c(\bar{s}^i_h, \bar{a}^i_h) + \begin{bmatrix} s - \bar{s}^i_h & a - \bar{a}^i_h \end{bmatrix} \begin{bmatrix} \nabla_s c(\bar{s}^i_h, \bar{a}^i_h) \\ \nabla_a c(\bar{s}^i_h, \bar{a}^i_h) \end{bmatrix} \\ &\quad + \frac{1}{2} \begin{bmatrix} s - \bar{s}^i_h & a - \bar{a}^i_h \end{bmatrix} \begin{bmatrix} \nabla_{ss} c(\bar{s}^i_h, \bar{a}^i_h) & \nabla_{sa} c(\bar{s}^i_h, \bar{a}^i_h) \\ \nabla_{as} c(\bar{s}^i_h, \bar{a}^i_h) & \nabla_{aa} c(\bar{s}^i_h, \bar{a}^i_h) \end{bmatrix} \begin{bmatrix} s - \bar{s}^i_h \\ a - \bar{a}^i_h \end{bmatrix}. \end{aligned}$$

Step 2: Compute the optimal policy. We can now solve the time-dependent LQR problem using the Riccati equation from subsection 3.4.1 to compute the optimal policy $\pi^i_0, \dots, \pi^i_{H-1}$.

Step 3: Generate a new series of actions. We can then generate a new sample trajectory by taking actions according to this optimal policy:

$$\bar{s}^{i+1}_0 = \bar{s}_0, \qquad \widetilde{a}_h = \pi^i_h(\bar{s}^{i+1}_h), \qquad \bar{s}^{i+1}_{h+1} = f(\bar{s}^{i+1}_h, \widetilde{a}_h).$$

Note that the states are sampled according to the *true* dynamics, which we assume we have query access to.

Step 4: Compute a better candidate trajectory., Note that we've denoted these actions as $\widetilde{a}_h$ and aren't directly using them for the next iteration $\bar{a}^{i+1}_h$. Rather, we want to *interpolate* between them and the actions from the previous iteration $\bar{a}^i_0, \dots, \bar{a}^i_{H-1}$. This is so that the cost will *increase monotonically,* since if the new policy turns out to actually be worse, we can stay closer to the previous trajectory. (Can you think of an intuitive example where this might happen?)

Formally, we want to find $\alpha \in [0,1]$ to generate the next iteration of actions $\bar{a}^{i+1}_0, \dots, \bar{a}^{i+1}_{H-1}$ such that the cost is minimized:

$$\begin{aligned} \min_{\alpha \in [0,1]} \quad & \sum_{h=0}^{H-1} c(s_h, \bar{a}^{i+1}_h) \\ \text{where} \quad & s_{h+1} = f(s_h, \bar{a}^{i+1}_h) \\ & \bar{a}^{i+1}_h = \alpha \bar{a}^i_h + (1-\alpha)\widetilde{a}_h \\ & s_0 = \bar{s}_0. \end{aligned}$$

Note that this optimizes over the closed interval $[0,1]$, so by the Extreme Value Theorem, it's guaranteed to have a global maximum.

The final output of this algorithm is a policy $\pi^{n_{\text{steps}}}$ derived after n_{steps} of the algorithm. Though the proof is somewhat complex, one can show that for many nonlinear control problems, this solution converges to a locally optimal solution (in the policy space).

3.6 Summary

This chapter introduced some approaches to solving different variants of the optimal control problem (3.1.1). We began with the simple case of linear dynamics and an upward-curved quadratic cost. This model is called the LQR and we solved for the optimal policy using dynamic programming. We then extended these results to the more general nonlinear case via local linearization. We finally saw the iterative LQR algorithm for solving nonlinear control problems.