DATA 37200: Learning, Decisions, and Limits
(Winter 2026)

Lecture 13: Linear Quadratic Regulator (LQR)

Instructor: Frederic Koehler

Reference

A good textbook reference for this and previous class is "Dynamic Programming and Optimal Control" by Bertsekas. See `https://www.mit.edu/~dimitrib/dpbook.html` for a lot of free supplementary content.

Elaborating on value iteration

Last time we introduced the idea of iterating the Bellman equations and showed that this map is a contraction in $\|\cdot\|_\infty$.
Then we used that every contraction map has a unique fixed point which it converges to.
For completeness, I will tell you the formal statement of this result (Banach fixed point theorem).

The Banach Fixed Point Theorem

The convergence of Value Iteration relies on the Bellman operator being a "contraction."

Definition: Contraction Mapping

Let (X, d) be a complete metric space. A function $T : X \to X$ is a γ-**contraction** if there exists a constant $\gamma \in [0, 1)$ such that for all $x, y \in X$:

$$d(T(x), T(y)) \leq \gamma \cdot d(x, y)$$

Banach Fixed Point Theorem

If T is a γ-contraction on a complete metric space, then:

1. **Existence & Uniqueness:** There exists a **unique** fixed point $x^* \in X$ such that $T(x^*) = x^*$.
2. **Convergence:** For any initial guess x_0, the sequence defined by $x_{n+1} = T(x_n)$ converges to x^*.
3. **Rate:** The convergence is geometric: $d(x_n, x^*) \leq \frac{\gamma^n}{1-\gamma} d(x_0, x_1)$.

Introduction

When can we actually solve the MDP for the optimal policy? This is difficult. Two important cases:

- **Tabular MDP**, i.e. small $|\mathcal{S}|$ and $|\mathcal{A}|$. You just use the Bellman equations ("value iteration").
  - Even if MDP is unknown, a UCB variant of value iteration learns a nearly-optimal policy from $poly(|\mathcal{S}|, \mathcal{A}|, 1/\gamma)$ episodes of play. Will discuss next class.
- **Optimal control** of linear dynamical systems. What we will cover today.
  - Infinite state space, so not suitable for tabular approach. Very practically important.
  - Remark: in the control theory convention, the objective is to find a policy which *minimizes cost* rather than *maximizes reward*. (cost = -reward.)

LQR as an MDP

The solution of the **Linear Quadratic Regulator (LQR)** is one of the most important results in control theory. LQR is an MDP:

- **State Space** $\mathcal{S}$: Continuous vector space $\mathbb{R}^n$. State is x_t.
- **Action Space** $\mathcal{A}$: Continuous vector space $\mathbb{R}^m$. Action is u_t.
- **Dynamics (Linear)**:

$$x_{t+1} = Ax_t + Bu_t \quad (+\epsilon_t)$$

  with $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. (Optional Gaussian noise ϵ_t.)
- **Reward (Quadratic Cost)**: Instead of maximizing reward, we minimize **Cost**.

$$c(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t$$

- $Q \succeq 0$ (PSD): Penalty for state deviation (e.g., error).
- $R \succ 0$ (PD): Penalty for control effort (e.g., energy).

The Objective Function

We seek a policy $\pi : \mathbb{R}^n \to \mathbb{R}^m$ to minimize the infinite horizon value function:

$$J(x_0) = \sum_{t=0}^{\infty} \gamma^t \left(x_t^\top Q x_t + u_t^\top R u_t \right)$$

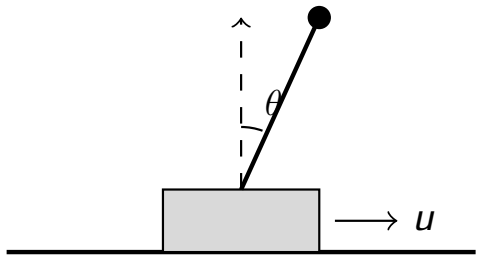
Key Differences from Previous Examples:

1. **Continuous Spaces**: We cannot use a lookup table for $Q(s, a)$ or $V(s)$.
2. **Structure**: The dynamics are linear, and cost is quadratic. This specific structure allows us to solve the Bellman equation **exactly** and **analytically**.
3. Remark: in RL context it is common to take $\gamma < 1$, but in control theory they typically consider $\gamma = 1$ (no discount).

Application 1: Balancing Systems (Inverted Pendulum)

Balancing a pole on a cart (or a Segway/hoverboard).

- **State** x_t:
  $[\text{position}, \text{velocity}, \text{angle}, \text{angular velocity}]^\top$.
- **Goal**: Keep angle ≈ 0 (upright)
  and position ≈ 0.
- **Linearization**: For small angles
  ($\sin\theta \approx \theta$), the physics are linear:
  $x_{t+1} = Ax_t + Bu_t$.



Why LQR?

- We tune Q to prioritize the angle (avoid falling) over position.
- The optimal policy $u = -Kx$ automatically learns to "counter-steer" (move the cart *under* the center of mass) to recover balance.
- **Real World:** Segways, Self-balancing robots, VTVL rockets (vertical takeoff and vertical landing).

Application 2: Industrial Process Control

Regulating Steady States: Many real-world systems (chemical reactors, aircraft cruising, cruise control) are non-linear but operate near a stable setpoint (equilibrium).

The Strategy:

1. **Linearize**: Approximate the dynamics around the operating point (x_{target}, u_{target}) using Taylor expansion:

$$x_{t+1} \approx A(x_t - x_{target}) + B(u_t - u_{target})$$

2. **Define Deviation**: Let Δx be the error. LQR drives $\Delta x \to 0$.

Example: Aircraft Autopilot

- **State**: Pitch, Roll, Yaw, and their rates.
- **Action**: Aileron, Elevator, Rudder deflection.
- **Role of LQR**: LQR finds the optimal feedback gains to dampen turbulence and return the plane to level flight with minimal oscillation and control effort.

What is the optimal policy?

Solving via Bellman Equation

Let $V(x)$ be the optimal value function (minimum cost-to-go) from state x. The Bellman Optimality Equation is:

$$V(x) = \min_u \left[x^\top Q x + u^\top R u + \gamma V(Ax + Bu) \right]$$

The Quadratic Ansatz

Since the cost is quadratic and dynamics are linear, we hypothesize that the Value function is also quadratic:

$$V(x) = x^\top P x$$

for some symmetric positive definite matrix $P \in \mathbb{R}^{n \times n}$. Then we solve to see if this actually solves the Bellman equations.

Substituting the Ansatz

Substitute $V(x) = x^\top P x$ into the Bellman equation:

$$x^\top P x = \min_u \Big[\underbrace{x^\top Q x + u^\top R u}_{\text{Immediate Cost}} + \gamma \underbrace{(Ax + Bu)^\top P (Ax + Bu)}_{\text{Future Value } V(x')} \Big]$$

To find the optimal u, we take the gradient with respect to u and set it to zero:

$$\nabla_u (\dots) = 2Ru + 2\gamma B^\top P (Ax + Bu) = 0$$

The Optimal Control Law

Solving for u:

$$2Ru + 2\gamma B^\top PBu + 2\gamma B^\top PAx = 0$$

$$(R + \gamma B^\top PB)u = -\gamma B^\top PAx$$

Optimal Policy $\pi^*(x)$

The optimal action is linear in the state:

$$u^* = -\underbrace{(R + \gamma B^\top PB)^{-1}\gamma B^\top PA}_{K} x = -Kx$$

This matrix K is called the **LQR Gain**. Note that

$$Ru^* = -\gamma B^\top P(Ax + Bu^*)$$

Deriving the Riccati Equation

We found u^*, but it depends on the unknown matrix P. Using the formula for Ru^*, we have

$$\begin{aligned} x^\top Px &= x^\top Qx + (u^*)^\top Ru^* + \gamma(Ax + Bu^*)^\top P(Ax + Bu^*) \\ &= x^\top Qx - \gamma(u^*)^\top B^\top P(Ax + Bu^*) + \gamma(Ax + Bu^*)^\top P(Ax + Bu^*) \\ &= x^\top Qx + \gamma(Ax)^\top P(A - BK)x \end{aligned}$$

so using that the equation holds for arbitrary x, we must have

$$P = Q + \gamma A^\top P(A - BK)$$

Plugging in

$$K = (R + \gamma B^\top PB)^{-1}\gamma B^\top PA$$

this yields the **Discrete Algebraic Riccati Equation (DARE)**:

$$P = Q + \gamma A^\top PA - \gamma^2 A^\top PB(R + \gamma B^\top PB)^{-1}B^\top PA$$

Value Iteration in LQR

The Zero Initialization (V_0):

- Set $V_0(x) = 0$ for all x, i.e.

$$P_0 = \mathbf{0}_{n \times n}$$

- $V_0(x) = 0$ corresponds to a problem with **zero time steps** remaining.
- If the game ends *now*, you pay no further cost.
- From here, keep iterating the DARE.

Value Iteration (DARE version):

$$P_{k+1} = \Phi(P_k) = Q + \gamma A^\top P_k A - \gamma^2 A^\top P_k B (R + \gamma B^\top P_k B)^{-1} B^\top P_k A$$

Here Φ is called the Riccati operator.

Variational inequality and convergence

We now prove convergence of the DARE recursion. Recall that we derived the DARE from the Bellman equation, so

$$x^\top \Phi(P)x = \min_u \left[x^\top Q x + u^\top R u + \gamma(Ax + Bu)^\top P(Ax + Bu) \right]$$

Recall that $P_0 = 0$. By the variational characterization, we have

$$P_0 = 0 \preceq \Phi(P_0) = P_1$$

and by induction that

$$P_t = \Phi(P_{t-1}) \preceq \Phi(P_t) \preceq P_{t+1}$$

for all times t. Hence the sequence of iterates is monotonically increasing, so either: (1) the sequence of iterates is unbounded ($\|P_t\| \to \infty$) or (2) they converge to a fixed point.

Under the assumption of controllability (see later slide), we can rule out situation (1).

LQR in the Context of RL

Standard LQR	**Standard RL**
Dynamics (A, B) known	Dynamics Unknown
Solve Riccati Equation	Learn $Q(s, a)$ from samples
Exact Solution found	Approximate Solution

Standard LQR assumes the dynamics are known, but if they are unknown we can first learn them ("system identification") and then do LQR. Analogous to Explore-Then-Commit strategy.

System Identification via Least Squares

Problem: What if the dynamics matrices A and B are **unknown**?

Data Collection: We run the system with random noise $u_t \sim \mathcal{N}(0, \sigma_u^2 I)$ to collect a dataset of transitions:

$$\mathcal{D} = \{(x_t, u_t, x_{t+1})\}_{t=0}^{T-1}$$

Objective (Least Squares): Find estimates $\hat{A}, \hat{B}$ that minimize the one-step prediction error:

$$(\hat{A}, \hat{B}) = \underset{A,B}{\operatorname{argmin}} \sum_{t=0}^{T-1} \|x_{t+1} - (Ax_t + Bu_t)\|_2^2$$

This is known to succeed under a standard assumption called controllability.

Controllability

Definition: A linear system (A, B) is **Controllable** if, for any initial state x_0 and any target state x^*, there exists a sequence of control inputs $u_0, \ldots, u_{k-1}$ that steers the system to x^* in finite time.

The Controllability Matrix: To check this, we construct the matrix $\mathcal{C} \in \mathbb{R}^{n \times nm}$:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \ldots & A^{n-1}B \end{bmatrix}$$

Kalman Rank Condition: The system is controllable if and only if $\mathcal{C}$ has full row rank:

$$\text{rank}(\mathcal{C}) = n$$

Why it matters for LQR:

- If the system is *not* controllable, there are "unreachable" parts of the state space.
- It may not be possible to identify A from samples if the system is not controllable. (ex: $B = 0$).

Summary

- LQR generalizes to a setting where the state is latent and we noisy linear observations. This generalization is called **LQG** (Linear-Quadratic-Gaussian).
- Basically LQG = LQR + Kalman filtering. Kalman filter updates are a version of recursive least squares.
- Handles this type of problem: a projectile has position + velocity, but only position is directly measured. (e.g. via GPS)
- Other approaches such as H_∞ control also well-studied.