Lecture 4

LQR and Continuous Dynamic Programming

In the previous lectures, we discussed dynamic programming for discrete states, $s \in \mathbb{S}$, and actions $a \in \mathbb{A}$. However, this is quite limiting, since in robotics we often have to reason about continuous state and action spaces.

4.1. Value Iteration on a Mesh

One way to extend value iteration to continuous state spaces is to do interpolation on a mesh. Here we assume we still discretize the action space and time.

What we would like to do is apply the value iteration update as follows:

$$\hat{J}^*(x) \Leftarrow \min_a \left[g(x,a) + \gamma \hat{J}^*(f(x,a)) \right] = \min_a \left[g(x,a) + \gamma \hat{J}^*(x') \right] \tag{4.1}$$

where

$$x_{k+1} = f(x_k, a_k). \tag{4.2}$$

However, due to the continuous nature of the state space, x_{k+1} may not be a point on a discretized grid. One solution to this problem is to use an interpolation scheme (e.g., barycentric interpolation).

Consider the interpolation strategy:

$$\hat{J}^*(x) = \sum_i \beta_i(x) \hat{J}^*(x_i), \quad \sum_i \beta_i = 1 \quad \beta_i > 0. \tag{4.3}$$

Here β_i is a weight on the i^{th} mesh point, x_i. Note that we are violating our typical notation where the subscript is not indicating time. Substituting this into our value iteration update gives

$$\hat{J}^*(x) \Leftarrow \min_a \left[g(x,a) + \gamma \sum_i \beta_i(x') \hat{J}^*(x_i) \right], \tag{4.4}$$

or, in vector form

$$\vec{\hat{J}}^* \Leftarrow \min_a \left[\vec{g} + \gamma P \vec{\hat{J}}^* \right]. \tag{4.5}$$

This final expression should look familiar!

4.2. Minimum-Time Control

Consider the simple double-integrator system

$$\ddot{q} = u, \tag{4.6}$$

with bounds on the input $|u| \leq 1$.

Suppose we want to solve for the minimum time policy

$$\min_{\pi} t_f \tag{4.7}$$
$$s.t. \quad q(t_0) = q0, \tag{4.8}$$
$$q(t_f) = 0, \tag{4.9}$$
$$\ddot{q} = u \tag{4.10}$$
$$|u| \leq 1. \tag{4.11}$$

We end up with a bang-bang controller given as

$$\begin{cases} 1 & (\dot{q} < 0 \text{ and } q \leq \frac{1}{2}\dot{q}^2) \text{ or } (\dot{q} \geq 0 \text{ and } q < -\frac{1}{2}\dot{q}^2) \\ 0 & \text{if } q = 0 \wedge \dot{q} = 0 \\ -1 & \text{if } otherwise \end{cases} . \tag{4.12}$$

For a detailed proof, see Kirk, Example 5.4-4.

4.3. Minimum-Time Control using Value Iteration

Can we approximate the minimum time solution using value iteration on mesh? See MATLAB example.

4.4. Discrete-Time Finite-Horizon LQR

With regards to our computational dynamic programming based approaches (e.g. value iteration), is there something better we can do in terms of handling continuous states and actions? There is, in the case of linear systems.

The original cost function (now rewritten for continuous state, x_k, and control, u_k) is given as

$$\min_{x_k, u_k} \quad J = \sum_{k=0}^{N-1} g_k(x_k, u_k) + g_N(x_N) \tag{4.13}$$
$$s.t. \quad x_{k+1} = f(u_k, u_k) \quad i = 0, 1...N-1. \tag{4.14}$$

Recall the Bellman Equation

$$J_k^*(x_k) = \min_{u_k \in U} \left[g(x_k, u_k) + J_{k+1}^*(f(x_k, u_k)) \right]. \tag{4.15}$$

Assume

$$x_{k+1} = Ax_k + Bu_k \tag{4.16}$$
$$g(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k \tag{4.17}$$
$$g(x_N) = x_N^T Q_N x_N, \tag{4.18}$$

where $Q \geq 0$,$Q_N \geq 0$,$R > 0$.

Here, we introduce the notion of a function being "positive-definite". This notion first appears with regards to quadrtic functions of the form $V(x) = x^T M x$.
A scalar continuous function $V(x)$ is said to be locally positive definite if

$$V(x) > 0, \quad \forall x \in \mathcal{D} - \{0\}, \quad V(0) = 0. \tag{4.19}$$

For functions, we will denote this $V(x) > 0, \quad \forall x \in D$. If this holds over the whole space, $V(x)$ is globally positive definite.
A function $V(x)$ is locally negative definite if $-V(x)$ is locally positive definite. This will be denoted

as $V(x) < 0, \quad \forall x \in D$.
A function V(x) is locally positive semi-definite if

$$V(x) \geq 0, \quad \forall x \in \mathcal{D} - \{0\}, \quad V(0) = 0. \tag{4.20}$$

This will be denoted as $V(x) \geq 0, \quad \forall x \in D$. A function $V(x)$ is locally negative semi-definite if $-V(x)$ is locally positive semi-definite and will be denoted $V(x) \leq 0, \quad \forall x \in D$. The global properities follow as in the case of a positive definite function above.

For global positive definite quadratic functions of the form $V(x) = x^T M x > 0, \quad \forall x \in \mathbb{R}^n$, we sometimes refer to the matrix M as being a "positive definite matrix". This is often denoted $M > 0$. Negative definite, positive semi-definite, and negative semi-definite matrices are are represented as $M < 0$, $M \geq 0$, and $M \leq 0$ respectively.

Substituting in we have

$$J_k^*(x_k) = \min_{u_k} \left[x_k^T Q x_k + u_k^T R u_k + J_{k+1}^*(x_k, u_k) \right]. \tag{4.21}$$

Starting at the last time step $k = N$, we can write

$$J_N^*(x_N) = x_N^T Q_N x_N, \tag{4.22}$$

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \left[x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + x_N^T Q_N x_N \right] \tag{4.23}$$

and

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \left[x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + \right. \tag{4.24}$$

$$\left. (A_{N-1} x_{N-1} + B_{N-1} u_{N-1})^T Q_N (A_{N-1} x_{N-1} + B_{N-1} u_{N-1}) \right]. \tag{4.25}$$

Taking the derivative with respect to the control inputs gives

$$\frac{\partial J_{N-1}^*(x_{N-1})}{\partial u_{N-1}} = 2 u_{N-1}^T R_{N-1} + 2 (A_{N-1} x_{N-1} + B_{N-1} u_{N-1})^T Q_N B_{N-1}. \tag{4.26}$$

A useful identity for taking the partial derivative of the matrix quadratic form $x^T P x$ with respect to x can be derived as follows. Let $y = Px$, then

$$x^T P x = x^T y. \tag{4.27}$$

Since this is a scalar $x^T y = y^T x$. We can apply the product rule to take the partial derivative as

$$\frac{\partial}{\partial x} x^T y = y^T + x^T P \tag{4.28}$$

$$= x^T P^T + x^T P \tag{4.29}$$

$$= x^T (P^T + P). \tag{4.30}$$

If P is symmetric positive definite,

$$\frac{\partial}{\partial x} x^T P x = 2 x^T P. \tag{4.31}$$

Setting the derivative equal to zero yields the control inputs resulting in the optimal cost-to-go given as

$$u_{N-1}^* = -\left[R_{N-1} + B_{N-1}^T Q_N B_{N-1} \right]^{-1} B_{N-1}^T Q_N A_{N-1} x_{N-1} \tag{4.32}$$

$$= -K_{N-1} x_{N-1}. \tag{4.33}$$

Now substitute back in the optimal policy into the dynamic programming recursion to get

$$J^*_{N-1}(x_{N-1}) = x^T_{N-1}Qx_{N-1} + x^T_{N-1}K_{N-1}RK_{N-1}x_{N-1}+ \tag{4.34}$$

$$(A_{N-1}x_{N-1} - B_{N-1}K_{N-1}x_{N-1})^T Q_N(A_{N-1}x_{N-1} - B_{N-1}K_{N-1}x_{N-1}). \tag{4.35}$$

Finally, collect the x_{N-1} terms and write

$$J^*_{N-1}(x_{N-1}) = x^T_{N-1}(Q_{N-1} + K^T_{N-1}RK_{N-1}+ \tag{4.36}$$

$$(A_{N-1} - B_{N-1}K_{N-1})^T Q_N(A_{N-1} - B_{N-1}K_{N-1}))x_{N-1} \tag{4.37}$$

$$= x^T_{N-1}S_{N-1}x_{N-1}. \tag{4.38}$$

Now we can assume and then verify that the same properties hold at $k-1$ and k and N and $N-1$. We have

$$u^*_k = -K_k x_k \tag{4.39}$$

and

$$J^*_k(x_k) = x^T_k S_k x_k \tag{4.40}$$

where

$$S_k = Q + K^T_k RK_k + (A_k - B_k K_k)^T S_{k+1}(A_k - B_k K_k) \tag{4.41}$$

and

$$K_k = (R + B^T_k S_{k+1}B_k)^{-1}B^T_k S_{k+1}A_k. \tag{4.42}$$

4.5. Discrete-Time Infinite Horizon LQR

In the case of value iteration, we observed how the cost-to-go and and the policy converged to a fixed-point. If A and B are not functions of time, K_k and S_k converge if the system is stabilizable. Dropping the time index, we have

$$S = Q + K^T RK + (A - BK)^T S(A - BK) \tag{4.43}$$

and

$$K = (R + B^T SB)^{-1}B^T SA. \tag{4.44}$$

4.6. Continuous-Time LQR

4.6.1 The Hamilton-Jacobi-Bellman Equation

We have seen the dynamic programming recursion

$$J^*_k(x_k) = \min_{u_k \in U} \left[g(x_k, u_k) + J^*_{k+1}(f(x_k, u_k))\right]. \tag{4.45}$$

This is useful for discrete time scenarios. However, if we take the limit as $t \to 0$, we arrive at the Hamilton-Jacobi-Bellman equation.

Divide the time-horizon in $\delta = T/N$ with $x_k = x(k\delta)$ and $u(k\delta)$ for $k = 0, 1, ..., N$.

The continuous-time dynamics are approximated by

$$x_{k+1} = x_k + f(x_k, u_k)\delta \tag{4.46}$$

and the cost-function is approximated by

$$h(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k)\delta. \tag{4.47}$$

Using the DP recursion, we have

$$\hat{J}(N\delta, x) = h(x) \tag{4.48}$$

$$\hat{J}(k\delta, x) = \min_{u \in U} \left[g(x,u)\delta + \hat{J}((k+1)\delta, x + f(x,u)\delta) \right]. \tag{4.49}$$

Do a Taylor series expansion of $\hat{J}$ around $(k\delta, x)$, and we have

$$\hat{J}((k+1)\delta, x + f(x,u)\delta) = \hat{J}(k\delta, x) + \frac{\partial \hat{J}}{\partial t}\delta + \frac{\partial \hat{J}}{\partial x}^T f(x,u)\delta + o(\delta). \tag{4.50}$$

Substituting in, we have

$$\hat{J}(k\delta, x) = \min_{u \in U} \left[g(x,u)\delta + \hat{J}(k\delta, x) + \frac{\partial \hat{J}}{\partial t}\delta + \frac{\partial \hat{J}}{\partial x}^T f(x,u)\delta + o(\delta) \right]. \tag{4.51}$$

If we divide by δ and let $\delta \to 0$, we get the HJB equations:

$$0 = \min_u \left[g(x,u) + \frac{\partial J^*}{\partial t} + \frac{\partial J^*}{\partial x} f(x,u) \right], \tag{4.52}$$

$$\pi^*(x) = \arg\min_u \left[g(x,u) + \frac{\partial J^*}{\partial t} + \frac{\partial J^*}{\partial x} f(x,u) \right] \tag{4.53}$$

4.7. Continuous-Time Finite Horizon LQR

Consider the linear, time-varying system

$$\dot{x} = A(t)x + B(t)u. \tag{4.54}$$

We desire to minimize the cost

$$J = g_f(x(t_f)) + \int_0^{t_f} g(x(t), u(t))dt. \tag{4.55}$$

Here

$$g_f(x) = x^T Q_f x, \quad Q_f = Q_f^T \geq 0 \tag{4.56}$$

$$g(x,u) = x^T Q x + u^T R u, \quad Q = Q^T \geq 0, \quad R = R^T > 0. \tag{4.57}$$

Starting with the HJB

$$0 = \min_u \left[g(x,u) + \frac{\partial J^*}{\partial x} f(x,u) + \frac{\partial J^*}{\partial t} \right], \tag{4.58}$$

$$\tag{4.59}$$

let's assume a particular form of the cost-to-go given as

$$J^*(x,t) = x^T S(t) x, \quad S(t) = S(t)^T \geq 0. \tag{4.60}$$

Substituting into the HJB, we have

$$0 = \min_u \left[g(x,u) + 2x^T S(t) f(x,u) + x^T \dot{S(t)} x \right], \tag{4.61}$$

$$= \min_u \left[x^T Q x + u^T R u + 2x^T S(t)(Ax + Bu) + x^T \dot{S}(t) x \right], \tag{4.62}$$

$$= \min_u \left[x^T Q x + u^T R u + 2x^T S(t) A x + 2x^T S(t) B u + x^T \dot{S}(t) x \right] \tag{4.63}$$

Now take the minimization with respect to u and set equal to zero as follows:

$$2u^T R + 2x^T S(t)B = 0 \tag{4.64}$$

$$Ru + B^T S(t)x = 0. \tag{4.65}$$

Solving for u gives

$$u = -R^{-1}B^T S(t)x \tag{4.66}$$

$$= -Kx. \tag{4.67}$$

Substitute in for u and get

$$x^T Qx + x^T K^T RKx + 2x^T S(t)Ax - 2x^T S(t)BKx + x^T \dot{S}(t)x = 0, \tag{4.68}$$

$$Q + K^T RK + 2S(t)A - 2S(t)BK = -\dot{S}(t), \tag{4.69}$$

and

$$S(t_f) = Q_f. \tag{4.70}$$

Bibliography

[1] Russ Tedrake. *Underactuated Robotics*. 2023.

[2] Dimitri Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[3] Drew Bagnell, Byron Boots, and Sanjiban Choudhury. *Modern Adaptive Control and Reinforcement Learning*. 2022.