AA203: Optimal and Learning-based Control Course Notes

James Harrison

April 8, 2019

2 Dynamic Programming and the Linear Quadratic Regulator

2.1 The Optimal Control Problems

In this section, we will outline the deterministic continuous-time optimal control problem that we will aim to solve. We will denote the state at time t as $\boldsymbol{x}(t) \in \mathbb{R}^n$, and the control as $\boldsymbol{u}(t) \in \mathbb{R}^m$. We will also occasionally write these as $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$, respectively. We will write the continuous-time systems dynamics as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t). \tag{1}$$

We will refer to a history of control input values during an interval $[t_0, t_f]$ as a control history, and we will refer to a history of state values over this interval as a state trajectory.

Different control problems may call for various constraints. For example, we may constrain a quadrotor to only fly in space not occupied by obstacles. Examples of constraints we will see are

- Initial and final conditions, $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$
- Trajectory constraints, $\underline{\boldsymbol{x}} \leq \boldsymbol{x}(t) \leq \bar{\boldsymbol{x}}$
- Control limits, $\underline{\boldsymbol{u}} \leq \boldsymbol{u}(t) \leq \bar{\boldsymbol{u}}$.

A state trajectory and control history that satisfy the constraints during the entire time interval $[t_0, t_f]$ are called admissible trajectories and admissible controls, respectively.

Finally, we will define the performance measure,

$$J = c_f(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} c(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \tag{2}$$

where c is the instantaneous cost function, and c_f is the terminal state cost. We are now able to state the continuous-time optimal control problem. We aim to find an admissible control,

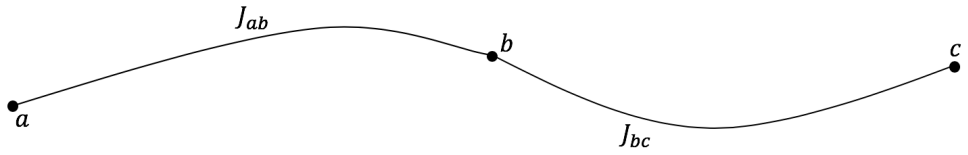

Figure 1: An optimal trajectory connecting point a to point c. There are no better (lower cost) trajectories than the sub-trajectory connecting b and c, by the principle of optimality.

$\boldsymbol{u}^*$, which causes the system (1) to follow an admissible trajectory, $\boldsymbol{x}^*$, that minimizes the performance measure given by (2). The minimizer $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ is called an optimal trajectory-control pair.

Note, first of all, that this is an extremely general problem formulation. We have not fixed our system dynamics, cost function, or specific constraints. We can't, in general, guarantee the existence or uniqueness of the optimal solution.

There are two possible solution forms for the optimal control. The first, $\boldsymbol{u}^* = e(\boldsymbol{x}(t_0), t)$ is referred to as an open-loop solution. This is an input function that is applied to the system, without using feedback. Practically, such solutions usually require augmentation with a feedback controller, as small model mismatch may lead to compounding errors. The second possible solution form is a feedback policy, $\boldsymbol{u}^* = \pi(\boldsymbol{x}(t), t)$. This feedback law maps all state-time pairs to an action and thus is usually more robust to possible model mismatch. However, depending on the particular problem formulation, open-loop solutions may be easier to compute.

2.2 Dynamic Programming and the Principle of Optimality

In this chapter we will outline the principle of optimality, and the method of dynamic programming (DP), one of two main approaches to solving the optimal control problem. The second, so-called variational approaches based on Pontryagin's Maximum Principle (PMP) will be discussed in future chapters. Dynamic programming has the strong advantage of yielding a feedback policy, however, exactly solving the dynamic programming problem is infeasible for many systems. We will address special cases in which the DP problem can be solved exactly, and approximate methods that work for a wide variety of systems.

Despite having just introduced the optimal control problem in continuous time, we will be operating in discrete time here, in which we aim to minimize

$$J_f(\boldsymbol{x}_0) = c_f(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} c(\boldsymbol{x}_k, \boldsymbol{u}_k, k). \tag{3}$$

We will extend the methods we develop in this chapter to continuous time in the next chapter.

The principle of optimality is as follows. Figure 1 shows a trajectory from point a to c. If the cost of the trajectory, $J_{ac} = J_{ab} + J_{bc}$, is minimal, then J_{bc} is also a minimum cost trajectory connecting b and c. The proof of this principle, stated informally, is simple.

Assume there exists an alternative trajectory connecting b and c, for which we will write the cost as $\tilde{J}_{bc}$, that achieves $\tilde{J}_{bc} < J_{bc}$. Then, we have

$$\begin{aligned} \tilde{J}_{ac} &= J_{ab} + \tilde{J}_{bc} && (4)\\ &< J_{ab} + J_{bc} && (5)\\ &= J_{ac}, && (6) \end{aligned}$$

and thus J_{ac} isn't minimal. More formally,

Theorem 2.1 (Discrete-time Principle of Optimality)**.** *Let $\pi^* = (\pi_0^*, \ldots, \pi_{N-1}^*)$ be an optimal policy. Assume state $\boldsymbol{x}_k$ is reachable. Consider the subproblem whereby we are at $\boldsymbol{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $(\pi_k^*, \ldots, \pi_{N-1}^*)$ is optimal for the subproblem.*

Dynamic programming, intuitively, proceeds backwards in time, first solving simpler shorter horizon problems. If we have found the optimal policy for times $k+1$ to $N-1$, along with the associated cost-to-go for each state, choosing the optimal policy for time k is a one step optimization problem. More concretely, we will assume we have dynamics of the form $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k)$ with $\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)$, and the cost given by (3). Then, dynamic programming iterates backward in time, from $N-1$ to 0, with

$$\begin{aligned} J_N(\boldsymbol{x}_N) &= c_T(\boldsymbol{x}_N) && (7)\\ J_k(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k \in \mathcal{U}(\boldsymbol{x}_k)} \left\{ c_k(\boldsymbol{x}_k, \boldsymbol{u}_k, k) + J_{k+1}(f(\boldsymbol{x}_k, \boldsymbol{u}_k, k)) \right\}. && (8) \end{aligned}$$

Note that here we have considered only deterministic dynamical systems (there is no stochastic disturbance).

Practically, dynamic programming raises many practical issues if one were to attempt to apply it directly. To perform the recursion, J_{k+1} must be known for all $\boldsymbol{x}_{k+1}$ (or more precisely, all $\boldsymbol{x}_{k+1}$ that are reachable from $\boldsymbol{x}_k$). If the state space is discrete (and relatively small), this is tractable as the cost-to-go may just be maintained in tabular form. In the next subsection, we will discuss an extremely important case in continuous space in which the cost-to-go can be computed exactly for all states. However, for general systems, we can not expect to be able to compute the cost-to-go for all states. Possible approaches to make the DP approach tractable are discretizing the state space, approximating the cost-to-go (i.e. restricting the family of functions that J_{k+1} may be in), or interpolating between cost-to-go computed for a finite set of states.

2.3 Discrete LQR

An important instance in which dynamic programming can be solved analytically for continuous state-action systems is the *linear quadratic regulator* problem. We will fix the dynamics of the system to be (possibly time-varying) linear,

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k \tag{9}$$

and the cost function is quadratic

$$c(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}(\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \boldsymbol{u}_k^T R_k \boldsymbol{u}_k) \tag{10}$$

$$c_N(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_N \boldsymbol{x}_k \tag{11}$$

where $Q_k \in \mathbb{R}^{n\times n}$ is positive semi-definite and $R_k \in \mathbb{R}^{m\times m}$ is positive definite for all $k = 0, \ldots, N$. Importantly, we assume $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are unconstrained for all k. To perform DP recursion, we initialize

$$J_N^*(\boldsymbol{x}_N) = \frac{1}{2}\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N := \frac{1}{2}\boldsymbol{x}_N^T V_N \boldsymbol{x}_N. \tag{12}$$

Then, applying (8), we have

$$J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \left\{\boldsymbol{x}_{N-1}^T Q_{N-1}\boldsymbol{x}_{N-1} + \boldsymbol{u}_{N-1}^T R_{N-1}\boldsymbol{u}_{N-1} + \boldsymbol{x}_N^T V_N \boldsymbol{x}_N\right\} \tag{13}$$

which, applying the dynamics,

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \{&\boldsymbol{x}_{N-1}^T Q_{N-1}\boldsymbol{x}_{N-1} + \boldsymbol{u}_{N-1}^T R_{N-1}\boldsymbol{u}_{N-1} \\ &+ (A_{N-1}\boldsymbol{x}_{N-1} + B_{N-1}\boldsymbol{u}_{N-1})^T V_N (A_{N-1}\boldsymbol{x}_{N-1} + B_{N-1}\boldsymbol{u}_{N-1})\}. \end{aligned} \tag{14}$$

Rearranging, we have

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) = \frac{1}{2}\min_{\boldsymbol{u}_{N-1}\in\mathbb{R}^m} \{&\boldsymbol{x}_{N-1}^T (Q_{N-1} + A_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \\ &+ \boldsymbol{u}_{N-1}^T (R_{N-1} + B_{N-1}^T V_N B_{N-1})\boldsymbol{u}_{N-1} \\ &+ 2\boldsymbol{u}_{N-1}^T (B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1}\}. \end{aligned} \tag{15}$$

Note that this optimization problem is convex in $\boldsymbol{u}_{N-1}$ as $R_{N-1} + B_{N-1}^T V_N B_{N-1} > 0$. Therefore, any local minima is a global minima, and therefore we can simply apply the first order optimality conditions. Differentiating,

$$\frac{\partial J_{N-1}^*(\boldsymbol{x}_{N-1})}{\partial \boldsymbol{u}_{N-1}} = (R_{N-1} + B_{N-1}^T V_N B_{N-1})\boldsymbol{u}_{N-1} + (B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \tag{16}$$

and setting this to zero yields

$$\boldsymbol{u}_{N-1}^* = -(R_{N-1} + B_{N-1}^T V_N B_{N-1})^{-1}(B_{N-1}^T V_N A_{N-1})\boldsymbol{x}_{N-1} \tag{17}$$

which we write

$$\boldsymbol{u}_{N-1}^* = L_{N-1}\boldsymbol{x}_{N-1} \tag{18}$$

which is a time-varying linear feedback policy. Plugging this feedback policy into (14),

$$\begin{aligned} J_{N-1}^*(\boldsymbol{x}_{N-1}) =&\boldsymbol{x}_{N-1}^T (Q_{N-1} + L_{N-1}^T R_{N-1} L_{N-1} \\ &+ (A_{N-1} + B_{N-1}L_{N-1})^T V_N (A_{N-1} + B_{N-1}L_{N-1}))\boldsymbol{x}_{N-1}. \end{aligned} \tag{19}$$

Critically, this implies that the cost-to-go is always a positive semi-definite quadratic function of the state. Because the optimal policy is always linear, and the optimal cost-to-go is always quadratic, the DP recursion may be recursively performed backward in time and the minimization may be performed analytically.

Following the same procedure, we can write the DP recursion for the discrete-time LQR controller:

1. $V_N = Q_N$
2. $L_k = -(R_k + B_k^T V_{k+1} B_k)^{-1}(B_k^T V_{k+1} A_k)$
3. $V_k = Q_k + L_k^T R_k L_k + (A_k + B_k L_k)^T V_{k+1}(A_k + B_k L_k)$
4. $\boldsymbol{u}_k^* = L_k \boldsymbol{x}_k$
5. $J_k^*(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k$

There are several implications of this recurrence relation. First, even if A, B, Q, R are all constant (not time-varying), the policy is still time-varying. Why is this the case? Control effort invested early in the problem will yield dividends over the remaining length of the horizon, in terms of lower state cost for all future time steps. However, as the remaining length of the episode becomes shorter, this tradeoff is increasingly imbalanced, and the control effort will decrease. However, for a linear time-invariant system, if (A, B) controllable, the feedback gain L_k approach a constant as the episode length approaches infinity. This time-invariant policy is practical for long horizon control problems, and may be approximately computed by running the DP recurrence relation until approximate convergence.

2.4 Iterative LQR and Differential Dynamic Programming

2.5 LQG

References