Lecture 17 2025-10-21

Today: linear quadratic regulator

So far, the optimization problems we've studied have been agnostic to the temporal nature of the optimal control problem

Today: how do we leverage the temporal nature of the problem to solve the problem more effectively?

Consider what's known as the discrete-time, finite horizon linear quadratic regulator:

$$\min_{x_{0:N},\, u_{0:N-1}} \quad x_N^T Q_g x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k$$

$$\text{subj. to:} \quad x_{k+1} = A x_k + B u_k$$

$$x_0 = x_{init}$$

$$x_k \in \mathbb{R}^{n_x}, \; u_k \in \mathbb{R}^{n_u}$$

$$Q, Q_g \in S_+^{n_x} \quad R \in S_+^{n_u}$$

This is "simply" a quadratic program without inequality constraints.

What structure can we exploit here?

↳ Note: the state trajectory implicitly follows from the "control tape" $u_{0:N-1}$

$$x_0 = x_{init}$$

$$x_1 = A x_0 + B u_0 = A x_{init} + B u_0$$

$$x_2 = A x_1 + B u_1 = A(A x_{init} + B u_0) + B u_1 = A^2 x_{init} + AB u_0 + B u_1$$

$$x_3 = A x_2 + B u_2 = A(A^2 x_{init} + AB u_0 + B u_1) + B u_2$$
$$= A^3 x_{init} + A^2 B u_0 + AB u_1 + B u_2$$

$$\vdots$$

$$x_N = A^N x_{init} + A^{N-1} B u_0 + \dots + B u_{N-1}$$

⇒ rewrite in matrix form:

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} I \\ A \\ A^2 \\ \vdots \end{pmatrix} x_{init} + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \ddots & \vdots \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & & \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

$$\underbrace{\begin{pmatrix}\vdots\\ x_N\end{pmatrix}}_{\vec{X}\in\mathbb{R}^{(N+1)n_x}} \quad \underbrace{\begin{pmatrix}\vdots\\ A^N\end{pmatrix}}_{\bar{A}\in\mathbb{R}^{(N+1)n_x\times(N+1)n_x}} \quad \underbrace{\begin{pmatrix}\vdots & \vdots & & \\ A^{N-1}B & A^{N-2}B & \cdots & B\end{pmatrix}}_{\bar{B}\in\mathbb{R}^{(N+1)n_x\times Nn_u}} \quad \underbrace{\begin{pmatrix}\vdots\\ u_{N-1}\end{pmatrix}}_{\vec{U}\in\mathbb{R}^{Nn_u}}$$

$$\rightarrow \quad \vec{X} = \bar{A}x_{init} + \bar{B}\vec{U}$$

Now we can plug this back into LQR equation:

$$\rightarrow J(\vec{U}) = (\bar{A}x_{init} + \bar{B}\vec{U})^T \begin{pmatrix} Q & & \\ & \ddots & \\ & & Q_g \end{pmatrix} (\bar{A}x_{init} + \bar{B}\vec{U})$$

$$+ \vec{U} \begin{pmatrix} R & & \\ & \ddots & \\ & & R \end{pmatrix} \vec{U}$$

$$= \| \mathrm{diag}(Q^{1/2}, \dots, Q_g^{1/2})(\bar{A}x_{init} + \bar{B}\vec{U}) \|_2^2 + \| \mathrm{diag}(R^{1/2}, \dots, R^{1/2})\vec{U} \|_2^2$$

So, writing the LQR problem in terms of $\vec{U}$ yields an unconstrained optimal control problem that can be solved analytically.

But can we do better than inverting a large (but sparse) matrix?

Bellman's Principle of Optimality:

"An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions."

R. Bellman, "An Introduction to the Theory of Dynamic Programming", 1953.

Informally, we can now consider optimizing the trajectory in terms of "tail optimality" and introduce the

definition of a value function:

Value function: $V_k : \mathbb{R}^{n_x} \to \mathbb{R}_+$

$$V_k(z) = \min_{u_{k:N-1}} x_N^T Q_g x_N + \sum_{t=k}^{N-1} x_t^T Q x_t + u_t^T R u_t$$

$$u_{k:N-1}$$

$$\text{subj. to: } \quad x_k = z$$

$$x_{t+1} = Ax_t + Bu_t, \quad t = k, \dots, N-1$$

This defines the optimum cost-to-go at a state z and starting at time k.

How do we leverage this? From Bellman principle, we can introduce the idea of tail optimality:

$$V_k(z) = \min_w \underbrace{z^T Q z + w^T R w}_{\text{stage cost}} + \underbrace{V_{k+1}(Az + Bw)}_{\text{optimum cost-to-go from next state}}$$

How can we solve this?

→ We (1) apply recursion and

(2) make use of an ansatz (i.e., guess)

We assume: $V_{k+1}(z) = z^T P_{k+1} z$ where $P_{k+1} \in S_+^{n_x}$

i.e. the cost-to-go is a quadratic function of state.

$$\rightarrow V_k(z) = \min_w z^T Q z + w^T R w + \underbrace{(Az + Bw)^T P_{k+1} (Az + Bw)}_{V_{k+1}(Az + Bw)}$$

to solve for optimal control, impose necessary conditions to solve for optimal control, impose necessary conditions of optimality:

$$\frac{\partial V_k(z)}{\partial w} = 2Rw + 2B^T P_{k+1} B w^* + 2B^T P_{k+1} A z := 0$$

$$= 2(R + B^T P_{k+1} B) w^* + 2B^T P_{k+1} A z = 0$$

$$\rightarrow \quad (R + B^T P_{k+1} B) w^* = -B^T P_{k+1} A z$$

$$\rightarrow \quad w^* = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A z$$

so $w^* = -K_k z$ where $K_k = (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A$

i.e. the optimal policy is a linear state feedback.

Note: the optimal policy is a function of time k

If we plug this expression back into $V_k(z)$:

$$V_k(z) = z^T P_k z$$

where $P_k = Q + A^T P_{k+1} A - A^T P_{k+1} B (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A$

and by construction: $P_k \in S_+^{n_x}$

How do we implement this?

LQR algorithm:

base case is: $V_N(z) = z^T Q_g z$ so $P_N = Q_g$

1. $P_N = Q_g$

2. for $k = N-1, \ldots, 0$

$$P_k = Q + A^T P_{k+1} A - A^T P_{k+1} B (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A$$

$$K_k = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A$$