16-745 Optimal Control Lecture 9

Reid Graves

February 16, 2025

1 Last time

- LQR via Shooting
- LQR as a QP
- LQR via Riccati
- Infinite Horizon LQR

2 Today

- Controllability
- Dynamic Programming

3 Controllability

- How do we know if LQR will work?
- We already know $Q \geq 0$, $R > 0$
- For the time-invariant case there is a simple answer
- For any initial state x_0, x_N is given by:

$$\begin{aligned}
x_n &= Ax_{n-1} + Bu_{n-1} \\
&= A(A_{x_{n-2}+Bu_{n-2}} + Bu_{n-1} \\
&\vdots \\
&= A^N x_0 + A^{N-1}Bu_0 + A^{n-2}Bu_1 + Bu_{N-1} \\
&= \begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix} \begin{bmatrix} u_{N-1} \\ u_{N-2} \\ u_{N-3} \\ \vdots \\ u_0 \end{bmatrix} + A^N x_0
\end{aligned}$$

- $\begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix}$ is C

- Without loss of generality, we solve for $x_N = 0$
- This is equivalent to a least-squares problem for $u_{0:N-1}$:

$$\begin{bmatrix} u_{N-1} \\ u_{N-2} \\ \vdots \\ u_0 \end{bmatrix} = \left[C^T(CC^T)^{-1}\right](x_N - A^N)x_0$$

- $C^T(CC^T)^{-1}$ is the "pseudo-inverse":
- For CC^T to be invertable:

$$\Rightarrow rank(C) = n, \quad n = dim(x)$$

- I can stop at n time steps in C because the Cayley-Hamilton theorem says that "A^N" can be written in terms of a linear combination of lower powers of A up to N

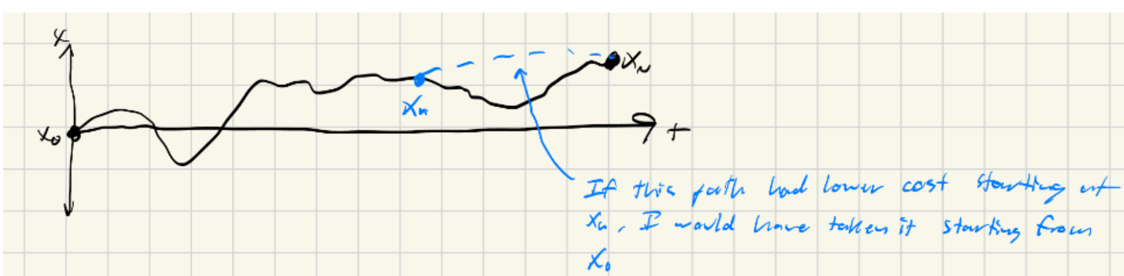
$$A^N = \sum_{k=0}^{N-1} \alpha_k A^k \quad \text{(for some } \alpha_k\text{)}$$

- therefore adding more time steps/columns to C can't increase the rank:

$$\Rightarrow C = \begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix} \quad \text{“Controllability Matrix”}$$

4 Bellman's Principle

- Optimal control problems have an inherently sequential structure
- Past control inputs can only affect future states. Future inputs can't affect past states.
- Bellman's Principle ("Princible of Optimality") formulates this



- Sub trajectories of optimal trajectories have to be optimal for the appropriately defined sub problem.

4.1 Dynamic Programming

- Bellman's Principle suggests starting from the end of the trajectory and working backwards.
- We've already seen this with Riccati and Pontryagin
- Define "Optimal cost to go" aka "Value function" $V_k(x)$
- Encodes cost incurred starting from state x at time k if we act optimally
- For LQR

$$V_N(x) = \frac{1}{2}x^T Q_N x = \frac{1}{2}x^T P_N x$$

- Back up one step and calculate $V_{N-1}(x)$:

$$V_{N-1} = \min_u \frac{1}{2}x_{N-1}^T Q x_{N-1} + \frac{1}{2}u^T R u + V_N(Ax_{N-1} + Bu_{N-1})$$

  We want to minimize for the control input u_{N-1}. Substituting in our value for V_N:

$$\Rightarrow \min_u \frac{1}{2}u^T R u + \frac{1}{2}(Ax_{N-1} + Bu)^T P_N (Ax_{N-1} + Bu)$$

  Taking the gradient and setting to 0:

$$\Rightarrow Ru + B^T P_N (Ax_{N-1} + Bu) = 0$$

  Then solving for u_{N-1} :

$$\Rightarrow u_{N-1} = -(R + B^T P_N B)^{-1} B P_N A x_{N-1} \quad \text{term before } x_{N-1} \text{is } (K_{N-1})$$

- Plug $u = -Kx$ back into

$$\Rightarrow V_{N-1}(x) = \frac{1}{2}x^T \left[Q + K^T R K + (A - BK)^T P_N (A - BK)\right] x$$

$$P_{N-1} = \left[Q + K^T R K + (A - BK)^T P_N (A - BK)\right]$$

$$\Rightarrow V_{N-1}(x) = \frac{1}{2}x^T P_{N-1} x$$

- Now we have a backward recursion for K and P that we iterate until $K = 0$

4.2 Dynamic Programming Algorithm

$$\begin{aligned}
&V_N(x) \leftarrow l_N(x) \\
&k \leftarrow N \\
&\text{while } k > 1 \\
&\quad V_{k-1}(x) = \min_{u \in \mathcal{U}} \left[l(x,u) + V_k(f(x,u)\right] \\
&k \leftarrow k - 1 \\
&\text{end}
\end{aligned}$$

- If we know $V_k(x)$, the optimal policy is:

$$u_k(x) = \arg\min_{u \in \mathcal{U}} \left[l(x,u) + V_{k+1}(f(x,u))\right]$$

- DP equations can be written equivalently written in terms of “action-value” or “Q” function:

$$S_k(x, u) = l(x, u) + V_{k+1}(f(x, u))$$

- Usually denoted $Q(x, u)$, but we’ll use $S(x, u)$
- Avoids need for explicit dynamics model

4.3 The curse of dimensionality

- DP is sufficient for global optimum
- Only tractable for simple problems (LQR, low dimensional)
- $V(x)$ stays quadratic for LQR but becomes impossible to write analytically, even for simple nonlinear problems
- Even if we could, $\min_u S(x, u)$ will be non-convex and possibly hard to solve
- Cost of DP blows up with state dimension due to cost of representing $V(x)$

4.4 Why do we care?

- Approximate DP with a function approximator for $V(x)$ or $S(x, u)$ is very powerful
- Forms basis for modern RL
- DP generalizes to stochastic problem (Just wrap everything in expectations). Pontryagin does not.

4.5 Finally: What are the Lagrange Multipliers?

- Recall Ricatti derivation from QP:

$$\lambda_k = p_k x_k$$

- From Dynamic Programing:

$$V(x) = \frac{1}{2} x^T P x$$
$$\Rightarrow \lambda_k = \nabla_x V_k(x)$$

- Dynamics multipliers are cost-to-go gradients
- Carries over to nonlinear setting (Not just LQR)