Last Time:

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

Today:

- Controllability
- Dynamic Programming

---

* Controllability:

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

$$\begin{aligned} x_N &= A x_{N-1} + B u_{N-1} \\ &= A(A x_{N-2} + B u_{N-2}) + B u_{N-1} \\ &\vdots \\ &= A^N x_0 + A^{N-1} B u_0 + A^{N-2} B u_1 + \cdots B u_{N-1} \\ &= \underbrace{\begin{bmatrix} B & AB & A^2B & \cdots & A^{N-1}B \end{bmatrix}}_{C} \begin{bmatrix} u_{N-1} \\ u_{N-2} \\ u_{N-3} \\ \vdots \\ u_0 \end{bmatrix} + A^N x_0 \end{aligned}$$

- Without loss of generality, we solve for $x_N = 0$

- This is equivalent to a least-squares problem for $u_{0:N-1}$:

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

- For CC^T to be invertable:

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

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

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

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

$$\Rightarrow \quad C = \underbrace{\left[B \;\; AB \;\; \cdots \;\; A^{n-1}B\right]}_{\text{"Controllability Matrix"}}$$

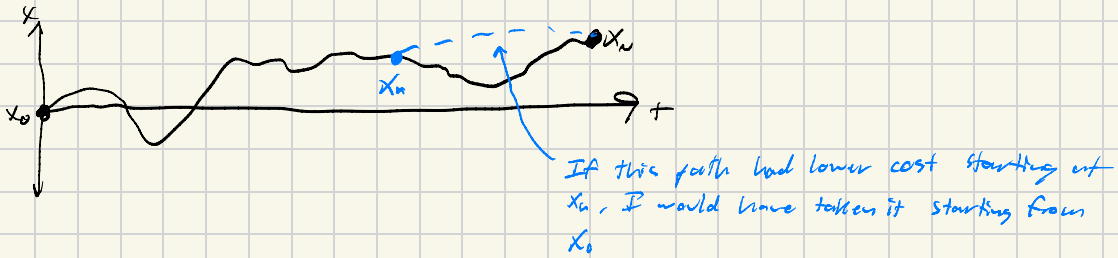
---

* 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 ("Principle of Optimality") formalizes this.



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

* Dynamic Programming

  - Bellman's Principle suggest starting from the end of the trajectory and working backwards.
  - We've already seen this with Riccati + 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 $V_{N-1}(x)$:

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

$$\Rightarrow \min_u \frac{1}{2} u^T R u + \frac{1}{2} (Ax_{N-1} + Bu)^T P_N (Ax_{N-1} + Bu) \qquad \nabla_u = 0$$

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

$$\Rightarrow u_{N-1} = -\underbrace{(R + B^T P_N B)^{-1} B P_N A}_{K_{N-1}} x_{N-1}$$

- Plug $u = -Kx$ back into *

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

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

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

* Dynamic Programming Algorithm:

$V_N(x) \leftarrow l_N(x)$

$k \leftarrow N$

while $k > 1$

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

$k \leftarrow k-1$

end

- If we know $V_0(x)$, the optimal policy is:

$$u_0(x) = \operatorname*{argmin}_{u \in \mathcal{U}} \left[l(x,u) + V_{0+1}(f(x,u)) \right]$$

- DP equations can be written equivalently written in terms of "action-value" or "Q" functions:

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

- Usually denoted $Q(x,u)$ but we'll use $S(x,u)$

- Avoids need for explicit dynamics model

* The Curse

- 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, $\min S(x,u)$ will be non-convex and possibly hard to solve
- Cost of DP blows up with state dimension due to cost of representing $V(x)$

* 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 problems (just wrap everything in expectations). Pontryagin does not.

* Finally: What are the Lagrange Multipliers?

- Recall Ricatti derivation from QP:

$$\lambda_n = P_n x_n$$

- From DP:

$$V(x) = \frac{1}{2} x^T P x$$

$$\Rightarrow \lambda_n = \nabla_x V_n(x)$$

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