Last Time:

- LQR as a QP
- Riccati Recursion

Today:

- Infinite-Horizon LQR
- Controllability
- Dynamic Programming

---

* Infinite Horizon LQR:

- For time-invariant LQR, K matrices converge to constant values
- For stabilization problems we usually use constant $K$
- Backward recursion for P:

$$K_n = (R + B^T P_{n+1} B)^{-1} B^T P_{n+1} A$$

$$P_n = Q + A^T P_{n+1} (A - B K_n)$$

- Infinite horizon limit $\Rightarrow P_n = P_{n+1} = P_{inf}$

$\Rightarrow$ Solve as a root-finding / fixed point problem for $P_{inf}$

- Julia/MATLAB dare function does this for you

## ★ 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:

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

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

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

must be invertable

- For $CC^T$ to be invertible:

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

- I get to stop at n time steps because the Cayley-Hamilton theorem says that $A^n$ can be written as a linear combination of lower powers of A:

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

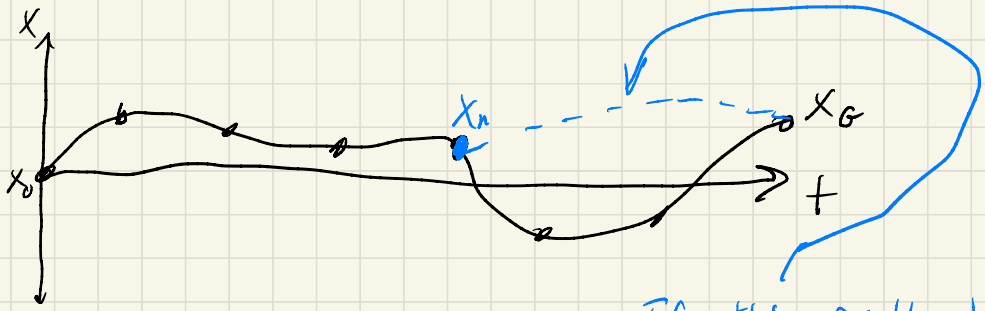
- Therefore adding more time steps / columns to C can't increase the rank

$$\Rightarrow \quad C = [B \quad AB \quad \cdots \quad A^{n-1}B]$$

---

* Bellman's Principle

- Optimal control problems have an inherently sequential structure.
- Past control inputs affect future states, but future control inputs can not affect past states
- Bellman's Principle "The Principle of Optimality" states the consequence of this for optimal trajectories.




If this path had lower cost starting at $X_n$, I would have taken it starting from $X_0$.

- Sub-trajectories of optimal trajectories must be optimal for appropriately defined sub problem.

* Dynamic Programming
  - Bellman's Principle suggests starting from the end of the trajectory and working backwards.
  - We've already seen hints of this in the Riccati equation and Pontryagin.
  - Define "optimal cost-to-go" a.k.a. "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 $V_{N-1}(x)$:

$$V_{N-1} = \min_u \underbrace{\frac{1}{2} x_{N-1}^T Q x_{N-1} + \frac{1}{2} u^T R u + V_N(A_{N-1} x_{N-1} + B_{N-1} u)}_{*}$$

$$= \min_u \frac{1}{2} u^T R u + \frac{1}{2} (A x_{N-1} + B u)^T P_N (A x_{N-1} + B u)$$

$\nabla_u = 0$

$$\Rightarrow R u + B^T P_N (A x_{N-1} + B u) = 0$$

$$\Rightarrow u_{N-1} = \underbrace{-(R_{N-1} + B_{N-1} P_N B_{N-1})^{-1} B_{N-1} P_N A_{N-1}}_{K_{N-1}} x_{N-1}$$

- Plug $u = -Kx$ back into $*$

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

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

- Now we have a backward recursion for $K$ and $P$ that we can iterate until $k = 0$
- Just Riccati recursion again!

## Backward Dynamic Programming Algorithm

$V_N(x) \leftarrow \ell_N(x)$

$k \leftarrow N$.

while $k > 1$

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

$k \leftarrow k-1$

end

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

$$u_k(x) = \arg\min_u \left[ \ell(x,u) + V_{k+1}(f(x,u)) \right]$$

- DP equations written equivalently in terms of "action-value" or "Q" function:

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

$$\Rightarrow u_k(x) = \arg\min_u S_k(x,u)$$

- Usually denoted $Q(x,u)$ but we'll use $S(x,u)$ to avoid confusion with LQR
- Avoids need for dynamics model.

* The Curse

- DP is sufficient for a global optimum
- Only tractable for simple problems (LQR or low-dimensional)
- $V(x)$ stays quadratic for LQR, but becomes impossible to write analytically for even 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 difficulty of representing $V(x)$

* Then Why DO We Care?

- Approximate DP where $V(x)$ or $S(x,u)$ is represented with a function approximator can work well.
- Forms the basis of modern RL
- DP generalizes to stochastic problems (just wrap everything in expectation operators). Pontryagin does not.

* Finally: What are the Lagrange Multipliers?

- Recall Riccati derivation from QP:

$$\lambda_n = P_n x_n$$

- From DP:

$$V_n(x) = \frac{1}{2} x^T P_n x$$

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

- Dynamics multipliers are cost-to-go gradients!
- Carries over to the general non-linear setting (not just LQR).

* Example:

$\lambda_n$ from QP matches $\nabla_x V_n(x)$ from DP