Last Time:

- LQR as a QP
- Riccati Recursion

Today:

- Dynamic Programming
- Convexity

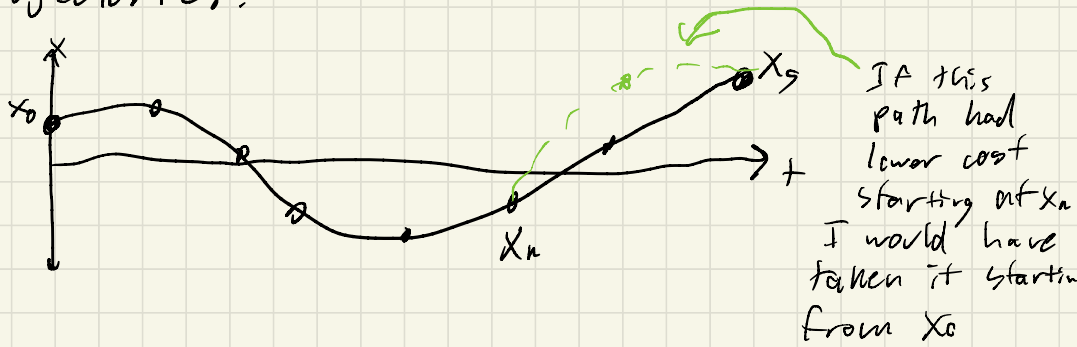
---

* Bellman's Principle

- Optimal control problems have an inherently sequential structure

- Past control inputs affect future states but future control inputs cannot affect past states

- Bellman's Principle "The Principle of Optimality" states the consequences of this for optimal trajectories:



- Sub-trajectories of optimal trajectories have to be optimal for the 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 in the co-state/multiplier equation from Pontryagin.
- Define "optimal cost-to-go" a.k.a "value function" $V_n(x)$
- Encodes cost incurred starting from state x at time k if we act optimally.
- For LQR:

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

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

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

$$= \min_u \frac{1}{2} u^T R_{N-1} u + \frac{1}{2}(A x_{N-1} + B_{N-1} u)^T Q_N (A_{N-1} x_{N-1} + B_{N-1} u)$$

$\nabla_u = 0$

$$\Rightarrow u^T R_{N-1} + (A_{N-1} x_{N-1} + B_{N-1} u)^T Q B_{N-1} = 0$$

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

Plug $u = -Kx$ back in to ✱:

$$V_{N-1}(x) = \frac{1}{2}x^T \underbrace{\left(Q_{N-1} + K_{N-1}^T R_{N-1} K + (A_{N-1} - B_{N-1}K_{N-1})^T Q_N (A_{N-1} - B_{N-1}K_{N-1})\right)}_{P_{N-1}} x$$

- Now we have a recursion in K and P that we can iterate until $k = 1$

- Just the Riccati equation again!

✱ Backward Dynamic Programming Algorithm

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

$k \leftarrow N$

while $k > 1$

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

$k \leftarrow k-1$

"Bellman Equation"

end

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

$$u_k(x_k) = \operatorname{argmin}_u \left[l(x_k, u) + V_{k-1}(f(x_k, u)) \right]$$

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

$$S_n(x,u) = l(x,u) + V_{n+1}(f(x,u))$$

$$\Rightarrow u_n(x_n) = \underset{u}{\operatorname{argmin}}\ S_n(x_n,u)$$

- These are usually denoted $Q(x,u)$ but we'll use $S(x,u)$ because Q is also used in LQR state cost.

* 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 down analytically for even simple nonlinear problems
- Even if we could, $\min S(x,u)$ will be non-convex and possibly hard to solve on its own.
- Cost of DP blows up with the 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 for a lot of modern RL
- DP generalizes to stochastic problems (just wrap everything in expectation operators). Pontryagin does not.

---

* Finally: What are those Lagrange Multipliers?

- Recall Riccati derivation from QP:

$$\lambda_n = P_n x_n, \quad P_n = Q + A^T P_{n+1}(A - BK)$$

$$= Q + K^T R K + (A-BK)^T P_{n+1}(A-BK)$$

Hessian of cost-to-go $V(x)$

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

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

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

---

* Example:

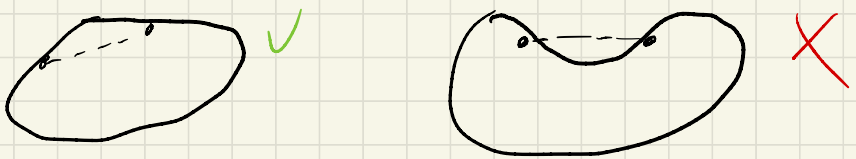
λ_n from QP matches $\nabla_x V_n(x)$ from DP

* Convex Model - Predictive Control

- LQR is very powerful, but often we need to explicitly reason about constraints.
- Often these are simple (e.g. torque limits) and can be encoded as a convex set.
- Constraints break the Riccati solution, but we can still solve the QP online.
- Convex MPC has gotten extremely popular as computers have gotten faster.

* Background: Convexity

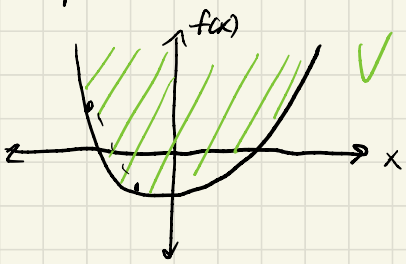
- Convex Set: a line connecting any two points in the set is contained in the set.



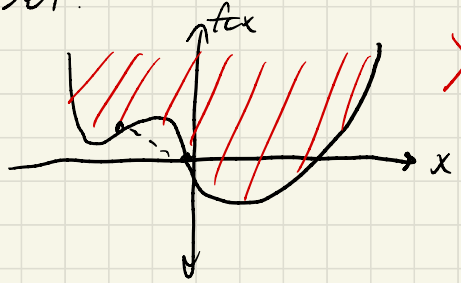
- Standard Examples:
  - Linear Subspaces ($Ax = b$)
  - Half-spaces / boxes / polytopes ($Ax \leq b$)
  - Ellipsoids ($x^T P x \leq 1$)
  - Cones ($\|x_{2:n}\|_2 \leq x_1$)

This is called the "second-order cone." It's the standard "ice cream cone" you're used to.

- Convex functions: a function $f(x): \mathbb{R}^n \to \mathbb{R}$ who's epigraph is a convex set.



✓



✗

- Standard examples:
  - Linear $\quad f(x) = c^T x$
  - Quadratic $\quad f(x) = \frac{1}{2} x^T Q x + q^T x \quad , \quad Q \succeq 0$
  - Norms $\quad f(x) = \|x\|$ (any norm)

- Convex optimization problem: minimize a convex function over a convex set.
- Standard examples:

  Linear Program (LP): linear f(x), linear C(x)

  Quadratic Program (QP): quadratic f(x), linear C(x)

  Quadratically "constrained" (QCQP): " , ellipsoid C(x)

  Second-order cone program (SOCP): linear f(x), cone C(x)

- Convex problems don't have spurious local optima that satisfy KKT

  ⇒ if you find a local KKT solution, you have the global optimum.

- Practically, Newton's method converges really fast and reliably (5~10 iterations max).

  ⇒ Can bound solution time for real-time control.