Last Time:

- Controllability
- Dynamic Programming

Today:

- Convexity Background
- Convex MPC

---

* Convex Model-Predictive Control

- LQR is very powerful but we often need to reason about constraints

- Often these are simple (e.g. actuator limits)

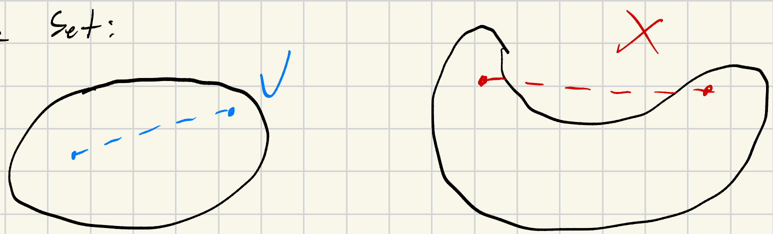
- Constraints break the Riccati solution, but we can still solve the QP online

- Convex MPC has gotten popular as computers have gotten faster

---

* Background: Convexity

- Convex Set:



- A line connecting any 2 points in the set is fully contained in the set

- Standard Examples

  - Linear subspaces $(Ax = b)$
  - Half space/box/polytope $(Ax \leq b)$
  - Ellipsoids $(x^T P x \leq 1,\ P \succ 0)$
  - Cones $\underbrace{(x_1 \geq \|x_{2:n}\|_2}_{\text{"second-order" cone (ice cream cone)}})$

---

* Convex Function

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

- Standard Examples:

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

- Convex optimization problem: minimize a convex function over a convex set

- Standard Examples:

  - Linear Program (LP): $f(x)$, $c(x)$ both linear
  - Quadratic Program (QP): Quadratic $f(x)$, linear $c(x)$
  - Quadratically-Constrained QP (QCQP): " , ellipsoid $c(x)$
  - Second-order Cone Program (SOCP): linear $f(x)$, cone $c(x)$

* Convex optimization problems don't have any spurious local optima that satisfy KKT

  $\Rightarrow$ If you find a local KKT solution, you have <u>The Solution</u>.

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

  $\Rightarrow$ Can bound solution time for real-time control

---

* Convex MPC

  - Think "constrained LQR"

  - Remember from DP, if we have a cost-to-go function $V(x)$, we can get u by solving a one-step problem:

$$u_n = \underset{u}{\operatorname{argmin}} \; l(x, u) + V_{n+1}(f(x, u))$$

$$= \underset{u}{\operatorname{argmin}} \; \frac{1}{2} u^T R u + (Ax + Bu)^T P_{n+1} (Ax + Bu)$$

  - We can add constraints on u to this one-step problem but this will perform poorly because $V(x)$ was computed without constraints.

- There's no reason I can't add more steps to the one-step problem:

$$\min_{\substack{x_{1:H} \\ u_{1:H-1}}} \sum_{n=1}^{H-1} \frac{1}{2} x_n^T Q x_n + \frac{1}{2} u_n^T R u_n + \underbrace{x_H^T P_H x_H}_{\text{LQR cost-to-go}}$$

- $H \ll N$ is called "Horizon"

- With no additional constraints, MPC ("receeding horizon") exactly matches LQR for any H

- Intuition: explicit constrained optimization over first H steps gets the state close enough to the reference that constraints are no longer active and LQR cost-to-go is valid farther into the future.

- In General:

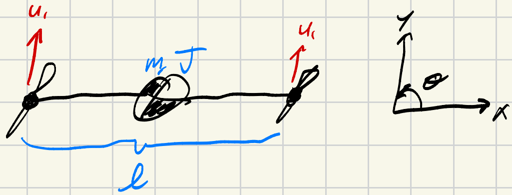
  - A good approximation of $V(x)$ is important for good performance

  - Better $V(x) \Rightarrow$ shorter horizon

  - Longer $H \Rightarrow$ less reliance on $V(x)$

* Example:

- Planar Quadrotor



$$
\begin{aligned}
m\ddot{x} &= -(u_1+u_2)\sin(\theta) \\
m\ddot{y} &= (u_1+u_2)\cos(\theta) - mg \\
J\ddot{\theta} &= \tfrac{1}{2} l (u_2 - u_1)
\end{aligned}
$$

- Linearize about hover:

$$\Rightarrow \quad u_1 = u_2 = \tfrac{1}{2} mg$$

$$
\Rightarrow \begin{cases}
\Delta\ddot{x} \approx -g\Delta\theta \\
\Delta\ddot{y} \approx \frac{1}{m}(\Delta u_1 + \Delta u_2) \\
\Delta\ddot{\theta} \approx \frac{1}{J}\frac{l}{2}(\Delta u_2 - \Delta u_1)
\end{cases}
$$

$$
\underbrace{\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{y} \\ \Delta\dot{\theta} \\ \Delta\ddot{x} \\ \Delta\ddot{y} \\ \Delta\ddot{\theta} \end{bmatrix}}_{\dot{x}}
=
\underbrace{\left[\begin{array}{ccc|c} & 0 & & I \\ \hline 0 & 0 & -g & \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \end{array}\right]}_{A}
\underbrace{\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \\ \Delta\dot{x} \\ \Delta\dot{y} \\ \Delta\dot{\theta} \end{bmatrix}}_{x}
+
\underbrace{\left[\begin{array}{cc} \multicolumn{2}{c}{0} \\ \hline 0 & 0 \\ 1/m & 1/m \\ -\frac{l}{2J} & \frac{l}{2J} \end{array}\right]}_{B}
\underbrace{\begin{bmatrix} \Delta u_1 \\ \Delta u_2 \end{bmatrix}}_{u}
$$

- MPC Cost Function:

$$
J = \sum_{n=1}^{H-1} \tfrac{1}{2}(x_n - x_{ref})^T Q (x_n - x_{ref}) + \tfrac{1}{2}\Delta u_n^T R \Delta u_n
$$
$$
+ \tfrac{1}{2}(x_H - x_{ref})^T P_H (x_n - x_{ref})
$$