Last Time:

- Deterministic Optimal Control
- Pontryagin
- Indirect Shooting

Today:

- LQR Problem
- LQR as a QP
- Riccati Recursion

---

* LQR Problem:

$$\min_{\substack{x_{1:N} \\ u_{1:N-1}}} \quad J = \sum_{n=1}^{N-1} \frac{1}{2} x_n^T Q_n x_n + \frac{1}{2} u_n^T R_n u_n + \frac{1}{2} x_N^T Q_N x_N$$

$$\text{s.t.} \quad x_{n+1} = A_n x_n + B_n u_n$$

$$Q_n \succeq 0 \;, \quad R_n > 0$$

* Example:

- "Double Integrator"

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

($\ddot{q}$: acceleration; q: position; $\dot{q}$: velocity; u: force)

- Think of this as a brick sliding on ice (no friction)

$$x_{n+1} = \underbrace{\begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q_n \\ \dot{q}_n \end{bmatrix}}_{x_n} + \underbrace{\begin{bmatrix} \frac{1}{2} h^2 \\ h \end{bmatrix}}_{B} u_n$$

(h: time step)

* LQR as a QP:

- Assume x_1 (initial state) is given (not a decision variable)

- Define $z = \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}$

- Define $H = \begin{bmatrix} R_1 & & & \\ & Q_2 & & 0 \\ & & R_2 & \\ & 0 & & \ddots \\ & & & Q_N \end{bmatrix}$

such that $J = \frac{1}{2} z^T H z$

- Define C and d:

$$\underbrace{\begin{bmatrix} B_1 & -I & 0 & \cdots & 0 \\ 0 & A_2 & B_2 & -I & 0 \\ & & \ddots & \ddots & \\ & & & A_{N-1} & B_{N-1} \quad -I \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ \vdots \\ x_N \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} -A_1 x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{d}$$

$$\Rightarrow Cz = d$$

- Now we can write LQR as a standard QP:

$$\min_z \ \frac{1}{2} z^T H z$$
$$\text{s.t. } Cz = d$$

- The Lagrangian of this QP is:

$$L(z, \lambda) = \frac{1}{2} z^T H z + \lambda^T [Cz - d]$$

- KKT Conditions:

$$\nabla_z L = Hz + C^T \lambda = 0$$
$$\nabla_\lambda L = Cz - d = 0$$

$$\Rightarrow \begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix}$$

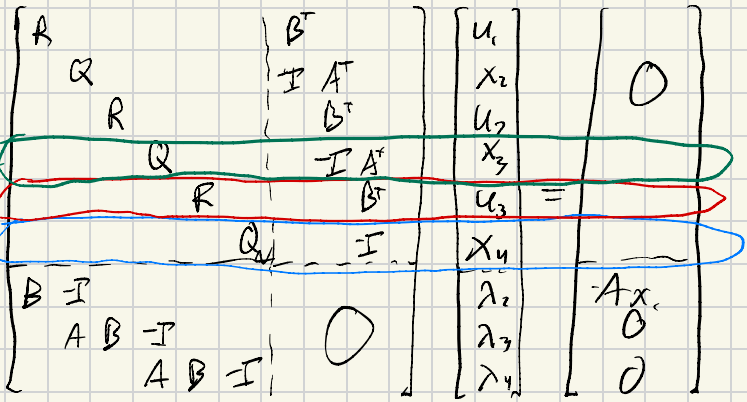
- We get the exact solution by solving one linear system!

* Example:

- Much better than shooting!

A closer look at the LQR QP:

- The KKT system for LQR is very sparse (lots of zeros) and has lots of structure:



$$Q_N x_4 - \lambda_4 = 0 \Rightarrow \lambda_4 = Q_N x_4$$

$$R u_3 + B^T \lambda_4 = R u_3 + B^T Q_N x_4 = 0$$

Plug in dynamics for x_4

$$\Rightarrow R u_3 + B^T Q_N (A x_3 + B u_3) = 0$$

$$\Rightarrow u_3 = -\underbrace{(R + B^T Q_N B)^{-1} B^T Q_N A}_{K_3} x_3$$

$$Q x_3 - \lambda_3 + A^T \lambda_4 = 0$$

Plug in λ_4

$$Q x_3 - \lambda_3 + A^T Q_N x_4 = 0$$

Plug in dynamics

$$Q x_3 - \lambda_3 + A^T Q_N (A x_3 + B u_3) = 0$$

Plug in $u_3 = -K_3 x_3$

$$\Rightarrow \lambda_3 = \underbrace{(Q + A^T Q_N (A - B K_3))}_{P_3} x_3$$

- Now we have a recursion for K and P:

$$P_N = Q_N$$

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

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

- This is called the Riccati equation / recursion

- We can solve the QP by doing a backward Riccati pass followed by a forward rollout to compute $x_{1:N}$ and $u_{1:N-1}$

- General (dense) QP has complexity $O([N(n+m)]^3)$ (horizon, state dim., control dim.)

- Riccati solution is $O(N(n+m)^3)$

* Example

  - Riccati exactly matches QP

  - Feedback policy lets us change x_0 and reject noise / disturbances

* Infinite-Horizon LQR

  - For time-invariant LQR converge to constants

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

  $\Rightarrow$ solve as a root-finding / fixed-point problem

- Julia / Matlab / Python "dare" function does this for you