Last Time:

- Control History
- Deterministic Optimal Control
- Pontryagin's Principle
- LQR Intro

Today:

- LQR via Shooting
- LQR as a QP
- Riccati Recursion

---

* The LQR Problem:

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

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

- Remember $Q \geq 0$, $R > 0$
- Can (locally) approximate many nonlinear problems
- Computationally tractable
- Many extensions e.g. infinite horizon, stochastic

* LQR with Indirect Shooting!

$$x_{n+1} = \nabla_\lambda H(x_n, u_n, \lambda_{n+1}) = A x_n + B u_n$$

$$\lambda_n = \nabla_x H(x_n, u_n, \lambda_{n+1}) = Q x_n + A^T \lambda_{n+1}$$

$$\lambda_N = Q_N x_N$$

$$\Delta u_n = \underset{\tilde{u}}{\operatorname{argmin}}\ H(x_n, \tilde{u}, \lambda_{n+1})$$

$$= -R^{-1} B^T \lambda_{n+1}$$

- Procedure:

1) Start with an initial guess trajectory
2) Simulate ("rollout") to get $x_{1:N}$
3) Backward pass to get λ and δu
4) Rollout with line search on δu
5) Go To 3 until convergence.

* 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; $\dot{q}$: velocity; q: position; u: force

- Think of this as a brick sliding on ice (frictionless)

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

time step

* LQR as a QP:

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

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

- Define:

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

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

- Define C and d:

$$\underbrace{\begin{bmatrix} B & (-I) & 0 & 0 & \cdots & & \\ 0 & A & B & (-I) & 0 & \cdots & \\ & & \ddots & \ddots & & & \\ & & & A_{N-1} & B_{N-1} & (-I) \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ \vdots \\ x_N \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} Ax_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{d}$$

- Now we can write the LQR problem as a standard QP:

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

- The Lagrangian of this QP is:

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

- KKT Conditions are:

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

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

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

* Example:
  - Double integrator
  - Compare shooting to QP

---

* A closer look at the LQR QP:

- The QP KKT system is very sparse (lots of zeros) and has a lot of structure

$$\begin{bmatrix} R & & & & B^T & & & \\ Q & R & & & -I & A^T & & \\ & Q & R & & & B^T & & \\ & & Q & R & & -I & A^T & \\ & & & Q_N & & & B^T & \\ & & & & & & -I & \\ B & -I & & & & & & \\ A & B & -I & & & & & \\ & A & B & -I & & & & \end{bmatrix} \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ -Ax_1 \\ 0 \\ 0 \end{bmatrix}$$

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

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

(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 feedback law

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

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

- Now we have a recursion for K and P:

$$P_N = Q_N$$

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

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

- This is called the Riccati equation / recursion
- We can solve the QP by doing a backward Riccati recursion followed by a forward rollout starting from x_1.
- QP has complexity $O(N^3 (n+m)^3)$
- Riccati solution is $O(N(n+m)^3)$. This carries over to the general nonlinear case

- Even more important: we now have a feedback policy instead of an open-loop trajectory

* Example:

  - Riccati solution exactly matches QP
  - Feedback policy lets us change x_0 and reject noise / disturbances.