

Last Time:

- Pontryagin's Minimum Principle
- LQR Problem
- Shooting Methods

Today:

- LQR as a QP
- Riccati Recursion

* The LQR Problem

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

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

- Remember: $Q \geq 0, R > 0$

* 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 $H = \begin{bmatrix} R_1 & & & & \\ Q_2 & 0 & & & \\ & R_2 & \ddots & & \\ & 0 & & \ddots & \\ & & & & Q_N \end{bmatrix}$ such that $J = \frac{1}{2} z^T H z$

- Define C and d

$$\underbrace{\begin{bmatrix} B & (-I) & & & & \\ 0 & A & B & -I & \cdots & 0 \\ & & & -I & \cdots & \\ & & & & \ddots & A_{n-1} \\ 0 & & & & & B_{n-1} & -I \end{bmatrix}}_C \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ \vdots \\ x_n \end{bmatrix} = \underbrace{\begin{bmatrix} -A_1 x_1 \\ \vdots \\ 0 \end{bmatrix}}_d$$

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

$$\min_z \frac{1}{2} z^T H z$$

$$\text{s.t. } Cz = d$$

- The Lagrangian of this QP is:

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

- The 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 / Sliding brick on ice
- Compare shooting to QP solution.

* A closer look at the LQR QP:

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

$$\begin{bmatrix} R & | & B^T & | & u_1 \\ Q & | & -I & A^T & | & \cdot \\ \hline A & | & B^T & | & x_2 \\ \hline Q & | & -I & A^T & | & u_2 \\ \hline R & | & B^T & | & x_3 \\ \hline Q_N & | & -I & | & u_3 = 0 \\ \hline B & -I & | & | & x_4 \\ \hline A & B & I & | & 0 \\ \hline A & B & -I & | & \lambda_1 \\ \hline & & O & | & \lambda_2 \\ \hline & & & | & \lambda_3 \\ \hline & & & | & \lambda_4 \\ \hline & & & | & -Ax_1 \\ \hline & & & | & 0 \\ \hline & & & | & 0 \end{bmatrix}$$

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

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

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

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

K₃

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

↓ λ₄

Plug in
dynamics

$$Qx_3 - \lambda_3 + A^T Q_n x_1 = 0 \quad \text{dynamics}$$

$$Qx_3 - \lambda_3 + A^T Q_n (Ax_3 + Bu_3) = 0 \quad \text{feedback law}$$

$$Qx_3 - \lambda_3 + A^T Q_n (A - BK) x_3 = 0$$

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

- 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 a Riccati equation/recursion
- We can solve the QP by doing a backward Riccati recursion followed by a forward rollout starting from x_1 .
- This has complexity $\mathcal{O}(N^{(nm)^3})$ instead of $\mathcal{O}(N^3(nm)^3)$ for the naive QP solution. This carries over to the general nonlinear case.
- Even more important: Now we have a feedback policy instead of an open-loop trajectory

Example:

- LQR w/ Riccati open-loop
- " closed-loop with noise

* Infinite Horizon:

- For time-invariant LQR K matrices converge to constant values
- For stabilization problems we almost always use the constant K
- Can solve for this explicitly with code in Julia / Matlab

* Controllability

- How do we know if LQR will work?
- For the time-invariant case there is a simple answer:

- For any initial state x_0 , x_n is given by:

$$x_n = Ax_{n-1} + Bu_{n-1}$$

$$= A(Ax_{n-2} + Bu_{n-2}) + Bu_{n-1}$$
$$\vdots$$

$$= A^n x_0 + A^{n-1}B u_0 + \dots + B u_{n-1}$$

$$x_n = \underbrace{[B \ AB \ A^2B \ \dots \ A^{n-1}B]}_{\text{"Controllability Matrix"} \ C} \begin{bmatrix} u_{n-1} \\ u_{n-2} \\ \vdots \\ u_0 \end{bmatrix} + A^n x_0$$

- In order for me to drive any x_0 to any desired x_n , the controllability matrix must have full row rank:

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

- This is equivalent to solving the following least-squares problem for $u_{0:n-1}$:

$$\begin{bmatrix} u_{n-1} \\ \vdots \\ u_0 \end{bmatrix} = \underbrace{\left[C^T \left(\overbrace{CC^T}^{\text{must be invertable}} \right)^{-1} \right]}_{\text{"Pseudo inverse"}} (x_n - A^n x_0)$$

- 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 smaller powers of A :

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

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