Last Time:

- Attitude Regulation
- Eigen-Axis Slew

Today:

- Time-Varying LQR

TVLQR:

- Last time we talked about how to generate open-loop trajectories
- We still need a feedback controller to track these online
- The combination of offline trajectory planning with online LQR tracking is very common + effective.

* Problem Setup:

$$\min_{x,u} \sum_{n=1}^{N-1} x_n^T Q x_n + u_n^T R u_n + x_N^T Q_N x_N$$

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

* Dynamic Programming Solution:

- Define "cost-to-go" function $V_k(x)$
- $V_n(x)$ gives the minimum cost to drive the system from state x at time t_n to the goal at t_N
- Start at t_N and work backwards:

$$V_N(x) = x^T Q_N x = x^T S_N x$$

$$V_{N-1}(x) = \min_u \Big[\underbrace{x^T Q x + u^T R u + (A_{N-1}x + B_{N-1}u)^T S_N (A_{N-1}x + B_{N-1}u)}\Big]$$

$$\frac{\partial}{\partial u}(\ ) = 2u^T R + 2u^T B_{N-1}^T S_N B_{N-1} + 2x^T A_{N-1}^T S_N B_{N-1} = 0$$

$$\Rightarrow \quad u = -\underbrace{(R + B_{N-1}^T S_N B_{N-1})^{-1} B_{N-1}^T S_N A_{N-1}}_{K_{N-1}} x$$

- Now plug u back in to get $V_{N-1}(x)$:

$$V_{N-1}(x) = x^T Q x + x^T K^T R K x + x^T (A - BK)^T S_N (A - BK) x$$

$$\Rightarrow \boxed{S_{N-1} = Q + K_{N-1}^T R K_{N-1} + (A_{N-1} - B_{N-1}K_{N-1})^T S_N (A_{N-1} - B_{N-1}K_{N-1})}$$

- We keep doing this recursively until we get to $k = 1$

TVLQR Algorithm Summary:

1) Initialize $S_N = Q_N$
2) Compute $K_n = (R + B_n^T S_{n+1} B_n)^{-1} B_n^T S_{n+1} A_n$
3) Compute $S_n = Q + K_n^T R K_n + (A_n - B_n K_n)^T S_{n+1} (A_n - B_n K_n)$
4) While $k > 1$, go to 2)

Applying TVLQR to Trajectory Tracking:

- Assume we have a nonlinear system $x_{n+1} = f(x_n, u_n)$
- Assume we have a nominal trajectory $\bar{x}_{1:N}$, $\bar{u}_{1:N-1}$ that we want to track

- Define Q and R matrices to trade off tracking error vs. control effort. Typically these are diagonal and a good starting guess is "Bryson's Rule":

$$Q = \mathrm{diag}\left(\left(1/\text{max deviation}\right)^2, \dots\right)$$

$$R = \mathrm{diag}\left(\left(1/\text{max control}\right)^2, \dots\right)$$

- Perform a 1^{st} order Taylor expansion along the nominal trajectory:

$$\cancel{x_{k+1}} + \delta x_{n+1} \approx \cancel{f(x_n, u_n)} + \underbrace{\frac{\partial f}{\partial x}\Big|_{x_n,u_n}}_{A_n} \delta x_n + \underbrace{\frac{\partial f}{\partial u}\Big|_{x_n,u_n}}_{B_k} \delta u_n$$

$$\Rightarrow \quad \delta x_{n+1} = A_n \delta x_n + B_n \delta u_n$$

- Compute TVLQR gain matrices for this linear system

- Online control law:

$$\underbrace{u_n}_{\substack{\text{actual}\\ \text{command}}} = \underbrace{\bar{u}_n}_{\text{nominal}} - \underbrace{K_k \delta x_n}_{\substack{\text{feedback}\\ \text{term}}}$$

- For a system where $x \in \mathbb{R}^n$, $\delta x_n = \underbrace{x_n}_{\text{actual}} - \underbrace{\bar{x}_k}_{\text{planned}}$

- When x includes a quaternion (or is subject to any constraints) we have to do some more work.

TVLQR for Attitude Tracking:

* Basic Idea: Use axis-angle vector for attitude part of δx

- Continuous-Time Dynamics:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{\omega} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} q \begin{bmatrix} \omega \\ 0 \end{bmatrix} \\ -J^{-1}[\omega \times (J\omega + B_w r) + B_w u] \\ u \end{bmatrix}$$

rotors

- Linearization:

$$\delta \dot{x} = \begin{bmatrix} \dot{\phi} \\ \delta\dot{\omega} \\ \delta\dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} -\hat{\omega} & I & 0 \\ 0 & -J^{-1}[\hat{\omega}J - \widehat{(J\omega + B_w r)}] & -J^{-1}\hat{\omega}B_w \\ 0 & 0 & 0 \end{bmatrix}}_{A(t)} \begin{bmatrix} \phi \\ \delta\omega \\ \delta r \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -J^{-1}B_w \\ I \end{bmatrix}}_{B(t)} u$$

- Convert to discrete time with a zero-order hold

$$\delta x_{n+1} = \underbrace{e^{A(t_n)\delta t}}_{A_n} \delta x_n + \underbrace{(e^{A(t_n)\delta t} - I)B(t_n)A^{-1}(t_n)}_{B_n} \delta u_n$$

This can be problematic

- Can be computed with c2d in MATLAB (regardless of invertability of A)

- Often 1st order Taylor expansion is used if time steps are small:

$$A_n \approx I + A(t_n)\delta t \quad , \quad B_n \approx B(t_n)\delta t$$

- To Implement Online:

1) Calculate error vs. plan: $$\delta x_k = \begin{bmatrix} \log(\bar{q}_k^+ q_k) \\ \omega_k - \bar{\omega}_k \\ r_k - \bar{r}_k \end{bmatrix}$$

2) Apply control with feedback correction:

$$u_k = \bar{u}_k - K_k \delta x_k$$