Last Time:

- Regularization + Line Search w/ constraints
- Deterministic Optimal Control Problem

Today:

- Pontryagin's Minimum Principle
- Linear-Quadratic Regulator Intro

---

* Pontryagin's Minimum Principle

- Also called "maximum principle" if you maximize a reward
- First-order necessary conditions for deterministic optimal control problem
- In discrete time, just a special case of KKT
- Given:

$$\min_{\substack{x_{1:N} \\ u_{1:N-1}}} \sum_{n=1}^{N-1} \ell(x_n, u_n) + \ell_F(x_N)$$

$$\text{s.t. } x_{n+1} = f(x_n, u_n)$$

- We can form the Lagrangian:

$$L = \sum_{n=1}^{N-1} \ell(x_n, u_n) + \lambda_{n+1}^T (f(x_n, u_n) - x_{n+1}) + \ell_F(x_N)$$

- This result is usually stated in terms of the "Hamiltonian"

$$H(x,u,\lambda) = \ell(x,u) + \lambda^T f(x,u)$$

- Plug H into L:

$$L = H(x_1,u_1,\lambda_2) + \left[\sum_{n=2}^{N-1} H(x_n,u_n,\lambda_{n+1}) - \lambda_n^T x_n\right] + \ell_F(x_N) - \lambda_N^T x_N$$

↑ note change to indexing

- Take derivatives w.r.t. x and λ:

$$\frac{\partial L}{\partial \lambda_n} = \frac{\partial H}{\partial \lambda_n} - x_{n+1} = f(x_n,u_n) - x_{n+1} = 0$$

$$\frac{\partial L}{\partial x_n} = \frac{\partial H}{\partial x_n} - \lambda_n^T = \frac{\partial \ell}{\partial x_n} + \lambda_{n+1}^T \frac{\partial f}{\partial x_n} - \lambda_n^T = 0$$

$$\frac{\partial L}{\partial x_N} = \frac{\partial \ell_F}{\partial x_N} - \lambda_N^T = 0$$

- For u we write the min explicitly to handle torque limits:

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

$$\text{s.t. } \tilde{u} \in \mathcal{U}$$

shorthand for "in feasible set"

e.g. $u_{min} \le \tilde{u} \le u_{max}$

- In Summary:

$$x_{k+1} = \nabla_\lambda H(x_k, u_k, \lambda_{k+1}) = f(x_k, u_k)$$

$$\lambda_k = \nabla_x H(x_k, u_k, \lambda_{k+1}) = \nabla_x \ell(x_k, u_k) + \left(\frac{\partial f}{\partial x}\right)^T \lambda_{k+1}$$

$$u_k = \underset{\tilde{u}}{\operatorname{argmin}}\ H(x_k, \tilde{u}, \lambda_{k+1})$$

$$\text{s.t. } \tilde{u} \in \mathcal{U}$$

$$\lambda_N = \frac{\partial \ell_F}{\partial x_N}$$

- These can be stated almost identically in continuous time:

$$\dot{x} = \nabla_\lambda H(x, u, \lambda) = f(x, u)$$ ← continuous time dynamics

$$-\dot{\lambda} = \nabla_x H(x, u, \lambda) = \nabla_x \ell(x, u) + \left(\frac{\partial f}{\partial x}\right)^T \lambda$$

$$u = \underset{\tilde{u}}{\operatorname{argmin}}\ H(x, \tilde{u}, \lambda)$$

$$\text{s.t. } \tilde{u} \in \mathcal{U}$$

$$\lambda_N = \frac{\partial \ell_F}{\partial x_N}$$

✱ Some Notes:

- Historically, many algorithms were based on integrating the continuous ODEs for $x(t)$, $\lambda(t)$ forward/backward to do gradient descent on $u(t)$
- These are called "indirect" and/or "shooting" methods.
- In continuous time $\lambda(t)$ is called "co-state" trajectory
- These methods have largely fallen out of favor as computers + solvers have improved

✱ Linear-Quadratic Regulator (LQR)

$$\min_{\substack{x_{1:N} \\ u_{1:N-1}}} \sum_{n=1}^{N-1} \underbrace{\frac{1}{2} x_n^T Q_n x_n + \frac{1}{2} u_n^T R_n u_n}_{Q \succeq 0,\ R \succ 0} + \frac{1}{2} x_N^T Q_N x_N$$

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

- Goal is to drive the system to the origin
- Called "time-invariant" LQR if $A_n = A$, $B_n = B$ $Q_n = Q$, $R_n = R$ $\forall k$
- Called "time-varying" (TVLQR) otherwise.
- Typically time-invariant LQR is for stabilizing an equilibrium and TVLQR is for tracking trajectories.

- Can (locally) approximate many nonlinear problems
- Many extensions e.g. infinite-horizon, stochastic
- Has been called "crown jewel of control theory"

* LQR with Indirect Shooting

$$x_{n+1} = Ax_n + Bu_n$$

$$\lambda_n = Qx_n + A^T\lambda_{n+1}, \quad \lambda_N = Qx_N$$

$$u_n = -R^{-1}B^T\lambda_{n+1}$$ ← this gives the gradient of the cost w.r.t. u_n

- Procedure:

1) Start with an initial guess trajectory
2) Simulate ("rollout") to get $x(t)$
3) Backward pass to get $\lambda(t)$ and $\Delta u(t)$
4) Rollout with line search on Δu
5) GoTo 3 until convergence

* Example:

- "Double Integrator": $\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$ (velocity, position, acceleration)

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

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

* Bryson's Rule

- Cost-tuning heuristic
- Set diagonal elements of Q and R to $\left(1/\text{max value}\right)^2$
- Normalizes all cost terms to 1
- Good starting point for tuning.

- Scalar case:

$$\underbrace{\tfrac{1}{2}Qx^2}_{=\left(\frac{1}{x_{max}}\right)^2 x_{max}^2 = 1} + \underbrace{\tfrac{1}{2}Ru^2}_{\rightarrow = \left(\frac{1}{u_{max}}\right)^2 u_{max}^2 = 1}$$