Last Time:

- Constrained Optimization
- Regularization w/ Constraints
- Merit Functions / Line searches

Today:

- Control History
- Deterministic Optimal Control
- Pontryagin's Minimum Principle
- Linear-Quadratic Regulators

---

* Deterministic Optimal Control

- Continuous time:

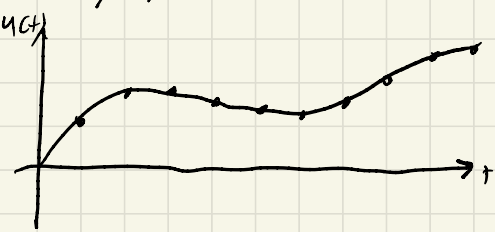
$$\min_{\substack{x(t)\\u(t)}} \underbrace{J(x(t), u(t))}_{\text{cost function}} = \int_{t_0}^{t_f} \underbrace{\ell(x(t), u(t))}_{\text{"stage cost"}}\, dt + \underbrace{\ell_F(x(t_f))}_{\text{"terminal cost"}}$$

$$\text{s.t. } \dot{x}(t) = f(x(t), u(t)) \quad \leftarrow \text{"dynamics constraint"}$$

(possibly other constraints)

state/input trajectories

- This is an "infinite dimensional" problem in the following sense:



$$U_{1:N} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} \Rightarrow u(t) = \lim_{N\to\infty} U_{1:N}$$

- Solutions are open-loop trajectories
- There are a handful of problems with analytic solutions but not many
- We will focus on the discrete-time setting which leads to tractable algorithms

* Discrete Time:

$$\min_{x_{1:N},\, u_{1:N-1}} \; J(x_{1:N}, u_{1:N-1}) = \sum_{n=1}^{N-1} l(x_n, u_n) + l_F(x_N)$$

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

$$u_{min} \le u_n \le u_{max} \quad \leftarrow \text{"torque limits"}$$

$$C(x_n) \le 0 \quad \leftarrow \text{obstacle/safety constraints}$$

- This is now a finite dimensional problem
- Samples x_n, u_n are often called "knot points"
- Convert continuous → discrete time using integration (e.g. Runge-Kutta etc.)
- Convert discrete → continuous using interpolation

* 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 special case of KKT conditions.
- Given:

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

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

- 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:

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

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

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

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

$$u_n = \underset{\tilde{u}}{\operatorname{argmin}} \; H(x_n, \tilde{u}, \lambda_{n+1}) \quad \text{s.t. } \tilde{u} \in \mathcal{U}$$

shorthand for "in feasible set"
e.g. $u_{min} \le \tilde{u} \le u_{max}$

- In Summary:

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

$$\lambda_n = \nabla_x H(x_n, u_n, \lambda_{n+1}) = \nabla_x \ell(x_n, u_n) + \left(\frac{\partial f}{\partial x_n}\right)^T \lambda_{n+1}$$

$$u_n = \underset{\tilde{u}}{\operatorname{argmin}} \; H(x_n, \tilde{u}, \lambda_{n+1}) \quad \text{s.t. } \tilde{u} \in \mathcal{U}$$

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

- These can be stated almost identically in continuous time;

$$\dot{x} = \nabla_\lambda H(x, u, \lambda) = f(x, u)$$

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

$$u = \underset{\tilde{u}}{\operatorname{argmin}} \; H(x, \tilde{u}, \lambda) \quad \text{s.t. } \tilde{u} \in \mathcal{U}$$

$$\lambda(t_f) = \frac{\partial \ell_F}{\partial x}$$

* Some Notes:

- Historically many algorithms were based on integrating the continuous ODEs 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 have improved.

* Linear - Quadratic Regulator (LQR)

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

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

- 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-variant" otherwise (TVLQR)
- Typically time-invariant LQR is for stabilizing fixed points and TVLQR is for tracking trajectories.