Last Time:

- Constrained Optimization
- Augmented Lagrangian
- QPs
- Regularization

Today:

- Merit functions + Line Searches
- Control History
- Deterministic Optimal Control

---

* Merit Functions

- How do we do a line search on a root-finding problem?

  find x^* such that $C(x^*) = 0$

- Define scalar "merit function" $P(x)$ that measures distance to solution

- Standard Choices:

  $P(x) = \frac{1}{2} C(x)^T C(x) = \frac{1}{2} \| C(x) \|_2^2$

  $P(x) = \| C(x) \|_1$ (any norm works)

- Now just do Armijo on $P(x)$:

  $\alpha = 1$

  while $P(x + \alpha \delta x) > P(x) + b\alpha \nabla P(x)^T \delta x$

  $\alpha \leftarrow \sigma \alpha$

  end

  (b: tolerance; α: step length)

$$x \leftarrow x + \alpha \Delta x$$

- How about constrained optimization?

$$\left.\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.t.} & c(x) \le 0 \\ & d(x) = 0 \end{array}\right\} \quad L(x, \lambda, \mu) = f(x) + \lambda^T c(x) + \mu^T d(x)$$

- Lot's of options for merit functions:

$$P(x, \lambda, \mu) = \frac{1}{2} \| r_{kkt}(x, \lambda, \mu) \|_2^2$$

r_{kkt} Residual $= \begin{bmatrix} \nabla_x L \\ \min(0, c(x)) \\ d(x) \end{bmatrix}$

$$P(x, \lambda, \mu) = f(x) + \rho \left\| \begin{bmatrix} \min(0, c(x)) \\ d(x) \end{bmatrix} \right\|_1$$

ρ: scalar weight; $\|\cdot\|_1$: any norm works

$$P(x, \lambda, \mu) = L_\rho(x, \lambda, \mu) \quad \text{(Augmented Lagrangian)}$$

* Example Take Aways:

- Excessive constraint penalties can cause problems
- All methods come with a merit function for free

* Deterministic Optimal Control

- Continuous time:

$$\min_{x(t),\,u(t)} \; J(x(t),u(t)) = \int_{t_0}^{t_f} \ell(x(t),u(t))\,dt + \ell_F(x(t_f))$$

$\min_{x(t),u(t)}$: state + input trajectories; $J(x(t),u(t))$: cost function; $\ell(x(t),u(t))$: "stage cost"; $\ell_F(x(t_f))$: "terminal cost"

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

(possibly other constraints)

- This is an "infinite dimensional" optimization problem
- 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 in this class

* Discrete Time:

$$\min_{X_{1:N},\, U_{1:N-1}} \; J(X_{1:N}, U_{1:N-1}) = \sum_{n=1}^{N-1} \ell(X_n, U_n) + \ell_F(X_N)$$

$$\text{s.t.} \quad X_{n+1} = f(X_n, U_n)$$

$$U_{min} \leq U \leq U_{max} \quad \leftarrow \text{"torque limits"}$$

$$C(X_n) \leq 0 \quad \leftarrow \text{obstacle/safety constraints}$$

- This is a finite dimension
- Samples X_n, U_n are often called "knot points"
- Continuous → discrete time using integration (e.g. Runge-Kutta)
- discrete → continuous using interpolation

* Pontryagin's Minimum Principle

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

- Given:

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

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

- We can form the Lagrangian

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

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

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

- Plug H into L:

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

Note change to indexing

- Take derivatives w.r.t. x and λ:

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

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

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

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

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

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

- In Summary:

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

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

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

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

- These can be stated in continuous time:

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

$$-\dot{\lambda} = \nabla_x H(x, u, \lambda)$$

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

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

* Some Notes:

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