Last Time:

- DDP w/ Constraints
- Min/free time problems

Today:

- Direct trajectory optimization
- Sequential quadratic programming
- Direct Collocation

---

* Direct Traj Opt

- Basic Strategy: discretize and "transcribe" continuous time optimal control problem into a nonlinear program (NLP):

objective (cost function)

$$\min_{x} \quad f(x)$$
$$\text{s.t.} \quad c(x) = 0$$
$$d(x) \leq 0$$

"Standard form NLP"

dynamics constraints

other constraints

- Several off-the-shelf large-scale NLP solvers can be used to solve the problem.

- Most common solvers: IPOPT (free), SNOPT (commercial)

- Common solution strategy: Sequential quadratic programming (SQP)

Sequential Quadratic Programming

- Strategy: Use 2nd order Taylor approximation of the Lagrangian and linearize $c(x)$, $d(x)$ to approximate the NLP as a QP:

$$\min_{\delta x} \frac{1}{2}\delta x^T H \delta x + g^T \delta x$$

$$\text{s.t. } c(x) + C\delta x = 0$$

$$d(x) + D\delta x \leq 0$$

where $H = \frac{\partial^2 L}{\partial x^2}$, $g = \frac{\partial L}{\partial x}$, $C = \frac{\partial c}{\partial x}$, $D = \frac{\partial d}{\partial x}$

$$L(x, \lambda, \mu) = f(x) + \lambda^T c(x) + \mu^T d(x)$$

- Solve QP to compute primal-dual search direction:

$$\delta z = \begin{bmatrix} \delta x \\ \delta \lambda \\ \delta \mu \end{bmatrix}$$

- Perform line search using merit function

- With only equality constraints, reduces to Newton method on the Lagrangian:

$$\underbrace{\begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = \begin{bmatrix} -\frac{\partial L}{\partial x} \\ -c(x) \end{bmatrix}}_{\text{"KKT System"}}$$

- Think of SQP as generalization of Newton to inequality constrained problems.

- Can use any QP solver to solve sub-problems, but good implementations typically use active-set strategies with warm starting tricks.

- For good performance on traj opt problems, taking advantage of sparsity in KKT system is crucial. SNOPT (Sparse Nonlinear OPTimizer) does this well.

- If inequalities are convex (e.g. cones), can generalize SQP to SCP (sequential convex programming) where inequalities are passed directly to the sub-problem solvers.

- SCP is still an active research area and not widely deployed yet.

---

* Direct Collocation

- So far we've used explicit RK methods to discretize dynamics:

$$\dot{x} = f(x,u) \rightarrow x_{n+1} = f(x_n, u_n)$$

- This makes sense if you're doing rollouts

- However, in a direct method where dynamics are enforced as equality constraints between knot points, this is not advantageous:

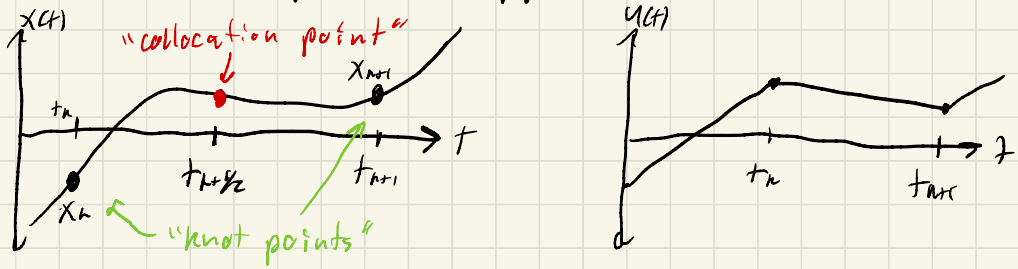
$$c(x_n, u_n, x_{n+1}, u_{n+1}) = 0$$

- Collocation methods represent trajectories with polynomial splines and enforce dynamics on the spline derivatives.

- Classic DIRCOL algorithm uses cubic splines for state trajectories and piecewise linear interpolation for $u(t)$

- Very high order polynomials are sometimes used (e.g. spacecraft trajectory design) but not common.

- DIRCOL Spline Approximations:



$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\downarrow$$

$$\dot{x}(t) = c_1 + 2c_2 t + 3c_3 t^2$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & h & h^2 & h^3 \\ 0 & 1 & 2h & 3h^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/h^2 & -2/h & 3/h^2 & -1/h \\ 2/h^3 & 1/h^2 & -2/h^3 & 1/h^2 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- Evaluate at $t_{n+1/2}$!

$$x_{n+1/2} = x(t_n + h/2) = \frac{1}{2}(x_n + x_{n+1}) + \frac{h}{8}(\dot{x}_n - \dot{x}_{n+1})$$

$$= \frac{1}{2}(x_n + x_{n+1}) + \frac{h}{8}(f(x_n, u_n) - f(x_{n+1}, u_{n+1}))$$

$$\dot{x}_{n+1/2} = \dot{x}(t_n + h/2) = -\frac{3}{2h}(x_n - x_{n+1}) - \frac{1}{4}(\dot{x}_n + \dot{x}_{n+1})$$

$$= -\frac{3}{2h}(x_n - x_{n+1}) - \frac{1}{4}(f(x_n, u_n) + f(x_{n+1}, u_{n+1}))$$

$$u_{n+1/2} = u(x_n + h/2) = \frac{1}{2}(u_n + u_{n+1})$$

- We can now enforce dynamics constraints!

$$c_i(x_n, u_n, x_{n+1}, u_{n+1}) =$$

$$f(x_{n+1/2}, u_{n+1/2}) - \left(-\frac{3}{2h}(x_n - x_{n+1}) - \frac{1}{4}(f(x_n, u_n) + f(x_{n+1}, u_{n+1})\right) = 0$$

continuous dynamics

- Note that only x_n, u_n are decision variables (not $x_{n+1/2}$, $u_{n+1/2}$)
- Called "Hermite-Simpson" integration
- Achieves 3rd order accuracy like RK3
- Requires fewer dynamics calls than explicit RK3!

RK3:

$$f_1 = f(x_n, u_n)$$
$$f_2 = f(x_n + \tfrac{1}{2}hf_1, u_n)$$
$$f_3 = f(x_n + 2hf_1 - hf_1, u_n)$$

$$x_{n+1} = x_n + \frac{h}{6}(f_1 + 4f_2 + f_3)$$

$\Rightarrow$ 3 dynamics evals per step
(no re-use across steps)

Hermite-Simpson:

$$f(x_{n+1/2}, u_{n+1/2}) + \tfrac{3}{2h}(x_n - x_{n+1}) - \tfrac{1}{4}\left(f(x_n, u_n) + f(x_{n+1}, u_{n+1})\right) = 0$$

these get re-used in adjacent steps!

$\Rightarrow$ Only 2 evals per step!

- Since dynamics calls often dominate computational cost, this can be a ~50% savings

* Example
  - Acrobot w/ DDP vs. DIRCOL