Last Time:

- Convex Optimization Overview
- Convex MPC

Today:

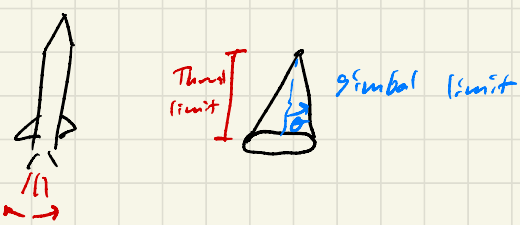
- Convex MPC Examples
- Nonlinear Trajectory Optimization
- Differential Dynamic Programming

---

* MPC Examples

- Rocket Landing

- Thrust vector constraints are conic:



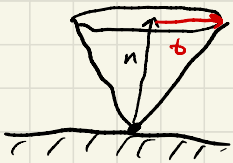
- SpaceX & JPL Mars landing use SOCP-based MPC with linearized dynamics

- Legged Robots

  - Contact forces must obey "friction cone" constraint

$$\|b\|_2 \leq \mu n$$

b: friction $\in \mathbb{R}^2$; μ: friction coefficient; n: normal force $\in \mathbb{R}$



  - Cone is often approximated as a pyramid so the constraint is linear ($Ax \leq b$)

- MIT Cheetah and other quadrupeds use QP-based MPC with linearized dynamics.

* What about Nonlinear Dynamics?

  - Linear stuff often works well, so use it if you can

  - Nonlinear dynamics make MPC problem non-convex
    ⇒ no convergence guarantees

  - Can work well in practice with effort

* Nonlinear Traj Opt Problem:

$$\min_{x_{1:N},\, u_{1:N-1}} \quad J = \sum_{n=1}^{N-1} \underbrace{l_n(x_n, u_n) + l_N(x_N)}_{\text{non-convex cost}}$$

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

$$\left.\begin{aligned} x_n &\in \mathcal{X}_n \\ u_n &\in \mathcal{U}_n \end{aligned}\right\} \text{non-convex constraints}$$

- Usually assume costs + constraints are C^2 (continuous 2nd derivatives)

---

* Differential Dynamic Programming (DDP)

- Nonlinear traj opt method based on approximate DP
- Use a 2nd-order Taylor expansion of cost-to-go in DP to compute Newton steps
- Very fast convergence is possible
- Can stop early for real-time applications

- Cost-to-go expansion:

$$V_n(x + \delta x) \approx V_n(x) + \underbrace{p_n^T \delta x}_{\text{gradient}} + \frac{1}{2} \delta x^T \underbrace{P_n}_{\text{Hessian}} \delta x$$

$$p_N = \nabla_x l_N(x) \quad , \quad P_N = \nabla_{xx}^2 l_N(x)$$

- Action-Value Function expansion:

$$S_n(x+\delta x, u+\delta u) \approx S_n(x,u) + \begin{bmatrix} g_x \\ g_u \end{bmatrix}^T \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} G_{xx} & G_{xu} \\ G_{ux} & G_{uu} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}$$

$$(G_{ux} = G_{xu}^T)$$

$$V_{n-1}(x) = \min_{\delta u} \Big[S_{n-1}(x,u) + g_n^T \delta x + g_u^T \delta u + \frac{1}{2} \delta x^T G_{xx} \delta x + \frac{1}{2} \delta u^T G_{uu} \delta u + \frac{1}{2} \delta x^T G_{xu} \delta u + \frac{1}{2} \delta u^T G_{ux} \delta x \Big]$$

$$\nabla_{\delta u}[\,\,] = g_u + G_{uu} \delta u + G_{ux} \delta x = 0$$

$$\Rightarrow \quad \delta u_{n-1} = \underbrace{-G_{uu}^{-1} g_u}_{d_{n-1}} \underbrace{- G_{uu}^{-1} G_{ux}}_{K_{n-1}} \delta x$$

$$= -d_{n-1} - K_{n-1} \delta x$$

"Feed-forward" term

"Feedback" term

- Plug back into S_{n-1} to get $V_{n-1}(x+\delta x)$

$$\Rightarrow V_{n-1}(x+\delta x) \approx V_{n-1}(x) + q_x^T \delta x + q_u^T(-d_{n-1} - K_{n-1}\delta x) + \frac{1}{2}\delta x^T G_{xx}\delta x + \frac{1}{2}(d_{n-1} + K_{n-1}\delta x)^T G_{uu}(d_{n-1} + K_{n-1}\delta x) - \frac{1}{2}\delta x^T G_{xu}(d_{n-1} + K_{n-1}\delta x) - \frac{1}{2}(d_{n-1} + K_{n-1}\delta x)^T G_{ux}\delta x$$

$$\Downarrow$$

$$P_{n-1} = G_{xx} + K_{n-1}^T G_{uu} K_{n-1} - G_{xu}K_{n-1} - K_{n-1}^T G_{ux}$$

$$p_{n-1} = q_x - K_{n-1}^T q_u + K_{n-1}^T G_{uu} d_{n-1} - G_{xu} d_{n-1}$$

- Need some more math to calculate q and G

---

* Matrix Calculus

- given $f(x): \mathbb{R}^n \to \mathbb{R}^m$, look at 2nd order Taylor expansion:

- if $m = 1$

$$f(x+\delta x) \approx f(x) + \underbrace{\frac{\partial f}{\partial x}}_{\mathbb{R}^{1\times n}}\delta x + \frac{1}{2}\delta x^T \underbrace{\frac{\partial^2 f}{\partial x^2}}_{\mathbb{R}^{n\times n}}\delta x$$

- if $m > 1$:

$$f(x+\delta x) \approx f(x) + \underbrace{\frac{\partial f}{\partial x}}_{\mathbb{R}^{m\times n}}\delta x + \underbrace{\frac{1}{2}\left(\frac{\partial}{\partial x}\left[\frac{\partial f}{\partial x}\delta x\right]\right)\delta x}_{\text{Ugly!}}$$

- for $m > 1$ $\frac{\partial^2 f}{\partial x^2}$ is a 3rd-rank tensor. Think of a "3D matrix". We need some tricks to work with these.

- Kronnecker Product:

$$\underbrace{A}_{l \times m} \otimes \underbrace{B}_{n \times p} = \underbrace{\begin{bmatrix} a_{11}B & a_{12}B & \cdots \\ a_{21}B & a_{22}B & \cdots \\ \vdots & & \ddots \end{bmatrix}}_{ln \times mp}$$

- Vectorization Operator

$$\underbrace{A}_{l \times m} = [A_1 \; A_2 \; A_3 \; \cdots \; A_m]$$

($A_1, A_2, \ldots$: column vectors)

$$\text{vec}(A) = \underbrace{\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix}}_{lm \times 1}$$

- The "vec trick"

$$\text{vec}(ABC) = (C^T \otimes A)\,\text{vec}(B)$$

$$\Rightarrow \text{vec}(AB) = (B^T \otimes I)\,\text{vec}(A) = (I \otimes A)\,\text{vec}(B)$$

- If we want to diff a matrix w.r.t. a vector, vectorize the matrix:

$$\frac{\partial A(x)}{\partial x} = \underbrace{\frac{\partial \text{vec}(A)}{\partial x}}_{lm \times n}$$

(employed whenever we diff a matrix)

- Back to Taylor expansion of $f(x)$:

$$f(x+\delta x) \approx f(x) + \underbrace{\frac{\partial f}{\partial x}}_{A}\delta x + \frac{1}{2}(\delta x^T \otimes I)\frac{\partial^2 f}{\partial x^2}\delta x$$

$$\hookrightarrow \frac{\partial}{\partial x}[\mathrm{vec}(I A_{(x)} \delta x)] \qquad \frac{\partial \mathrm{vec}(A)}{\partial x}$$

- Sometimes we want to diff through a transpose:

$$\frac{\partial}{\partial x}(A_{(x)}^T B) = (B^T \otimes I)T\frac{\partial A}{\partial x}$$

"commutator matrix"

$$T\,\mathrm{vec}(A) = \mathrm{vec}(A^T)$$

- Action-Value Function Derivatives:

$$S_n(x,u) = l_n(x,u) + V_{n+1}(f(x,u))$$

$$\Rightarrow \frac{\partial S}{\partial x} = \frac{\partial l}{\partial x} + \frac{\partial V}{\partial f}\underbrace{\frac{\partial f}{\partial x}}_{A} \Rightarrow \boxed{g_x = \nabla_x l + A_n^T p_{n+1}}$$

$$\frac{\partial S}{\partial u} = \frac{\partial l}{\partial u} + \frac{\partial V}{\partial f}\underbrace{\frac{\partial f}{\partial u}}_{B} \Rightarrow \boxed{g_u = \nabla_u l + B_n^T p_{n+1}}$$

$$G_{xx} = \frac{\partial g_x}{\partial x} = \nabla_{xx}^2 l(x,u) + A_n^T P_{n+1} A_n + (p_{n+1} \otimes I)^T T \frac{\partial A_n}{\partial x}$$

$$G_{uu} = \frac{\partial g_u}{\partial u} = \nabla_{uu}^2 l(x,u) + B_n^T P_{n+1} B_n + (p_{n+1} \otimes I)^T T \frac{\partial B_n}{\partial u}$$

$$G_{xu} = \frac{\partial g_x}{\partial u} = \nabla_{xu}^2 l(x,u) + A_n^T P_{n+1} B_n + (p_{n+1} \otimes I)^T T \frac{\partial A_n}{\partial x}$$

"Tensor Terms"