Lecture 16: Differential Dynamic Programming

Last Time: • Direct Collocation

Today: • A superficial primer on optimization algorithms

• Differential Dynamic Programming

- Combines Direct Shooting, Dynamic Programming, and indirect ideas
- Computational efficiency & Rapid Convergence

REVIEW:

Formulating Traj. Opt via Collocation:

$$\min_{u(\cdot)} \int \ell(t, x(t), u(t))\, dt$$

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

Suppose: N_e finite elements, N_c collocation points @ $\tau_1, \dots \tau_{N_c}$ in each finite element

Transcription Into a nonlinear programming (NLP) problem

$$\min_{\{u_{k,j},\, x_k,\, x_{k,j},\, \dot{x}_{k,j}\}} \sum_{k=1}^{N_e} h \sum_{j=1}^{N_c} w_j\, \ell(t_{k,j}, x_{k,j}, u_{k,j}) \qquad t_{k,j} = t_k + h\tau_j$$

s.t.

Collocation constraints

$$\dot{x}_{k,j} = f(t_{k,j}, x_{k,j}, u_{k,j})$$

$$x_{k,j} = x_k + \sum_{i=1}^{N_c} B_{j,i}\, \dot{x}_{k,i} \qquad \forall\, k \in \{0, N_e\},\ j \in \{1, N_c\}$$

$$x_{k+1} = x_k + h \sum_{i=1}^{N_c} w_i\, \dot{x}_{k,i} \qquad \forall\, k \in \{0, N_e - 1\}$$

A simpler setting...

Quadratic Programming (QP) Problem

$$\min_x \ \tfrac{1}{2} x^T Q x + c^T x \qquad Q \succeq 0$$

$$\text{s.t.} \quad Ax = b$$

$$\mathcal{L} = \tfrac{1}{2} x^T Q x + c^T x + \lambda^T (Ax - b)$$

First-order necessary condition for opt. (FONC)

$$\nabla_x \mathcal{L} = Qx^* + c + A^T \lambda = 0$$

$$\nabla_\lambda \mathcal{L} = Ax^* - b = 0$$

KKT system:

$$\underbrace{\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix}}_{:= K \ \text{KKT Matrix}} \begin{bmatrix} x^* \\ \lambda \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

$$\begin{bmatrix} x^* \\ \lambda \end{bmatrix} = K^{-1} \begin{bmatrix} -c \\ b \end{bmatrix}$$

- If $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$

  $K^{-1} \begin{bmatrix} -c \\ b \end{bmatrix}$ takes $O((n+m)^3)$ math operations

- If the KKT matrix is sparse you can solve this system much more quickly

When You Have a Hammer, Everything Looks like a nail: NLP via SQP

$\min_x f(x)$ s.t. $g(x)=0$ ⇐ Solve by approximating it as a QP (locally)

Given guess: x^k @ k-th iteration. Let $x = x^k + \delta x$

① Derive a local QP approx via taylor Series:

$$\min_{\delta x} f(x^k) + \nabla f(x^k)^T \delta x + \tfrac{1}{2}\delta x^T \nabla^2 f(x^k)\delta x + \text{h.o.t.}$$

$$\text{s.t. } g(x^k) + \nabla g(x^k)^T \delta x = 0 + \text{h.o.t.}$$

Goal: Find δx^* that improves cost & constraint violations

↩ Many many details here that are out of scope of this course.
See: Math Programming EE Nonlinear & Stoch. CBE optimization

② Move in direction From

$$\begin{bmatrix} \delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} \nabla^2 f & (\nabla g)^T \\ \nabla g & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\nabla f(x^k) \\ -g(x^k) \end{bmatrix}$$

Steps in this direction will be closer to satisfying the FONC of optimality

③ Update guess: $x^{k+1} = x^k + \delta x^*$

④ Repeat back to ①

Such methods are known as: Sequential Quadratic Programming (SQP)

Differential Dynamic Programming

- Dynamic programming doesn't scale ...
- Direct optimization is only open loop ...
- Rather than doing value iteration everywhere, why not just along a trajectory?

  => This is the main idea of DDP (Mayne 1966)

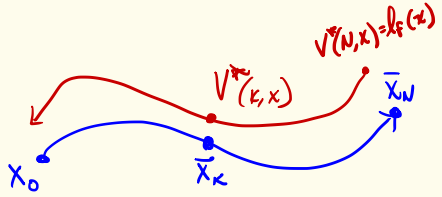
  => Gives an optimal open-loop control
  + locally optimal policy

  => Exploits structure of the traj. opt. problem

  ↳ Comuputes an SQP-like step w/o ever having to solve the KKT system

Discrete Time DDP : $x_{k+1} = f(x_k, u_k)$

$$\min_{U_{1:N}} \sum_{i=0}^{N-1} \ell(x_i, u_i) + \ell_f(x_N)$$



Iterative shooting method

- Guess controls: $\bar{u}_{1:N}$
- Simulate to get $\bar{x}_{1:N}$ (Forward Pass)
- Approximate $V^*(k, x)$ in the neighborhood of $\bar{x}_k$ up to second order (Backward Pass)
  - Derive a locally-optimal control policy $u_k^*(x_k)$ based on $V^*(k+1, x_{k+1})$
  - Use this policy to propagate the value function backward in time

$$V^*(k,x) = \ell(x, u_k^*(x)) + V^*(k+1, f(x, u_k^*(x)))$$

- use this policy to generate a new $\bar{x}_{1:N}$, $\bar{u}_{1:N}$ with lower cost
- Repeat until convergence

Approximating V^* locally to 2nd order

$$\min \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_f(x_N)$$
$$\text{s.t. } x_{k+1} = f(x_k, u_k)$$
$$x_0 \text{ fixed}$$

Suppose @ time $k+1$

$$V^*(k+1, \bar{x}_{k+1} + \delta x_{k+1}) = V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] + V_x[k+1]^T \delta x_{k+1} + \frac{1}{2} \delta x_{k+1}^T V_{xx}[k+1] \delta x_{k+1} + \text{h.o.t.}$$

- $V(k+1, \bar{x}_{k+1})$: Total cost from $k+1 \to N$ for current guess $\bar{u}_{k+1:N}$
- $\Delta V[k+1]$: Expected change in cost if $u_{k+1:N}$ is chosen optimally and $\delta x = 0$
- $V_x[k+1]$: Gradient of optimal cost-to-go w.r.t. x (@ $\bar{x}_{k+1}$)
- $V_{xx}[k+1]$: Hessian of optimal cost-to-go w.r.t. x (@ $\bar{x}_{k+1}$)

Define the Q-function:

$$Q(k, x, u) = \ell(x, u) + V^*(k+1, f(x, u))$$

Cost when you ① start @ state x @ time k
② Apply u @ time k
③ Apply optimal controls after that

Bellman Equation $\Rightarrow$

$$V^*(k, x) = \min_u Q(k, x, u)$$

To Second Order:

$$Q(k, \bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) = \ell(\bar{x}_k, \bar{u}_k) + V^*(k+1, \bar{x}_{k+1}) + \Delta V[k+1] + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} + \text{h.o.t.}$$

$$Q_x = \nabla_x H(\bar{x}_k, \bar{u}_k, V_x[k+1])$$

$$Q_u = \nabla_u H(\bar{x}_k, \bar{u}_k, V_x[k+1])$$

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

$$Q_{xx} = \nabla_{xx} H(\bar{x}_k, \bar{u}_k, V_x[k+1]) + \left[\frac{\partial f}{\partial x}\right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial x}\right]$$

$$Q_{uu} = \nabla_{uu} H(\bar{x}_k, \bar{u}_k, V_x[k+1]) + \left[\frac{\partial f}{\partial u}\right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial u}\right]$$

$$Q_{ux} = \nabla_{ux} H(\bar{x}_k, \bar{u}_k, V_x[k+1]) + \left[\frac{\partial f}{\partial u}\right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial x}\right], \quad Q_{xu} = Q_{ux}^T$$

Derivation of Previous Slide:

$$V^*(k+1, \bar{x}_{k+1} + \delta x_{k+1}) = V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] + V_x[k+1]^T \delta x_{k+1} + \frac{1}{2} \delta x_{k+1}^T V_{xx}[k+1] \delta x_{k+1} + \text{h.o.t}$$

$$Q(k, \bar{x}_k + \delta x_k, \bar{u}_k + \delta u) = \ell(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) + V^*(k+1, f(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k))$$

$$= \ell(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) + V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] + V_x[k+1]^T \delta x_{k+1} + \frac{1}{2} \delta x_{k+1}^T V_{xx}[k+1] \delta x_{k+1}$$

Note that: $\delta x_{k+1} = f(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) - f(\bar{x}_k, \bar{u}_k) = \frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k + \text{h.o.t.}$

$$Q(k, \bar{x}_k + \delta x_k, \bar{u}_k + \delta u) = \ell(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) + V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] + V_x[k+1]^T \left(f(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) - f(\bar{x}_k, \bar{u}_k) \right)$$

$$+ \frac{1}{2} \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right] + \text{h.o.t.}$$

$$= H(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k, V_x[k+1]) + V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] - V_x[k+1]^T f(\bar{x}_k, \bar{u}_k)$$

$$+ \frac{1}{2} \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right] + \text{h.o.t.}$$

$$= \underbrace{\ell(\bar{x}_k, \bar{u}_k) + V_x[k+1]^T f(\bar{x}_k, \bar{u}_k)}_{H(\bar{x}_k, \bar{u}_k, V_x[k+1])} + H_x^T \delta x_k + H_u^T \delta u_k + \frac{1}{2} \delta x_k^T H_{xx} \delta x_k + \delta u_k^T H_{ux} \delta x_k + \frac{1}{2} \delta u_k^T H_{uu} \delta u$$

$$+ V(k+1, \bar{x}_{k+1}) + \Delta V[k+1] - V_x[k+1]^T f(\bar{x}_k, \bar{u}_k) + \frac{1}{2} \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right]^T V_{xx}[k+1] \left[\frac{\partial f}{\partial x} \delta x_k + \frac{\partial f}{\partial u} \delta u_k \right] + \text{h.o.t.}$$

Locally Optimal Feedback: Find δu^* to minimize $Q(k, \bar{x}_k + \delta x, \bar{u}_k + \delta u)$

From prev. slide:

$$Q(k, \bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) = \ell(\bar{x}_k, \bar{u}_k) + V^*(k+1, x_{k+1}) + \Delta V[k+1] + \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}$$

$$\delta u_k^* = \underset{\delta u_k}{\arg\min} \; \frac{1}{2} \delta u_k^T Q_{uu} \delta u_k + \delta x_k^T Q_{xu} \delta u_k + Q_J^T \delta u$$

$$\Rightarrow \delta u_k^* = -\varepsilon Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} \delta x_k$$

Feed forward change u_k — Feed back component

- Let $\varepsilon = 1$ for now...
- ε controls the change in the control trajectory from iteration to iteration

Plugging this control law back in: $V^*(k, \bar{x}_k + \delta x_k) = Q(k, \bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k^*)$

By matching terms for our approximation:

$$V^*(k, \bar{x}_k + \delta x_k) = V(k, \bar{x}_k) + \Delta V[k] + \delta x_k^T V_x[k] + \frac{1}{2} \delta x_k^T V_{xx}[k] \delta x_k$$

$$\Rightarrow$$

$$\Delta V[k] = \Delta V[k+1] - \varepsilon\left(1 - \frac{\varepsilon}{2}\right) Q_u^T Q_{uu}^{-1} Q_u$$

$$V_x[k] = Q_x - Q_{xu} Q_{uu}^{-1} Q_u$$

$$V_{xx}[k] = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux}$$

Derivation of Previous Slide:

$$Q(k, \bar{x}_k+\delta x, \bar{u}_k+\delta u) = \underbrace{\ell(\bar{x}_k,\bar{u}_k) + V(k+1, \bar{x}_{k+1})}_{V(k,\bar{x}_k)} + \Delta V[k+1] + Q_x^T \delta x + Q_u^T \delta u + \delta u^T Q_{ux} \delta x + \frac{1}{2}\delta x^T Q_{xx} \delta x + \frac{1}{2}\delta u^T Q_{uu} \delta u$$

$Q_{ux} = Q_{xu}^T$

- Plug in $\delta u^* = -\varepsilon Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} \delta x$

$$V^*(k, \bar{x}_k+\delta x) = Q(k, \bar{x}_k+\delta x, \bar{u}_k+\delta u^*) = V(k,\bar{x}_k) + \Delta V[k+1] + Q_x^T \delta x - Q_u^T\left[\varepsilon Q_{uu}^{-1} Q_u + Q_{uu}^{-1} Q_{ux}\delta x\right]$$
$$- \delta x^T Q_{xu} Q_{uu}^{-1} Q_{ux} \delta x - \varepsilon Q_u^T Q_{uu}^{-1} Q_{ux}\delta x + \frac{1}{2}\delta x^T Q_{xx}\delta x$$
$$+ \frac{1}{2}\delta x^T Q_{xu} Q_{uu}^{-1} Q_{ux}\delta x + \frac{1}{2}\varepsilon^2 Q_u^T Q_{uu}^{-1} Q_u + \varepsilon Q_u^T Q_{uu}^{-1} Q_{ux}\delta x$$

$$= V(k,\bar{x}_k) + \underbrace{\Delta V[k+1] - \varepsilon\left(1-\frac{\varepsilon}{2}\right)\left[Q_u^T Q_{uu}^{-1} Q_u\right]}_{\Delta V[k]} + \underbrace{\left(Q_x - Q_{xu}Q_{uu}^{-1}Q_u\right)^T \delta x}_{V_x[k]^T\delta x} + \underbrace{\frac{1}{2}\delta x^T\left[Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{ux}\right]\delta x}_{\frac{1}{2}\delta x^T V_{xx}[k]\delta x}$$

DDP algorithm: Input guess controller $\bar{u}(k,x)$

① Simulate $\bar{x}_{1:N}$ via $\bar{u}$, store nominal cost $\bar{L} = \sum \ell + \ell_f(\bar{x}_N)$

② Backward Pass

Seed $V_x[N] = \nabla \ell_f|_{\bar{x}_N}$, $V_{xx}[N] = \nabla^2_{xx} \ell_f|_{\bar{x}_N}$

for $k = N-1:0$

Form $Q_x, Q_u, Q_{xx}, Q_{ux}, Q_{uu}, Q_{ux}$ from $\bar{x}_k, \bar{u}_k, V_x[k+1]$ $V_{xx}[k+1]$

$\delta u^*_k = -\varepsilon Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} \delta x_k$ (Store control law)

Update $V_x[k]$, $V_{xx}[k]$ via previous slide

③ Forward Pass [Algorithm parameters: $0<\gamma<1$, $0<\beta<1$ (You pick)]

ⓐ Set $\varepsilon = 1$

ⓑ Simulate a $x_{1:N}$ via the policy $u_k = \bar{u}_k - \varepsilon Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} (x_k - \bar{x}_k)$ (δx_k)

ⓒ Record $L = \sum \ell(x_k, u_k) + \ell_f(x_N)$, $\Delta V = \sum_{i=1}^{N} -\varepsilon(1-\frac{\varepsilon}{2}) Q_u^T Q_{uu}^{-1} Q_u$

ⓓ If $L < \bar{L} + \gamma \Delta V$ then go to ⓔ else $\varepsilon = \beta\varepsilon$ and go to ⓑ

ⓔ $\bar{x}_{1:N} \leftarrow x_{1:N}$, $\bar{u}_{1:N} \leftarrow u_{1:N}$

Until convergence

DDP Pros & Cons

$L(U_{1:N}) = \sum_{k=1}^{N} l(x_k, u_k) + l_f(x_N) \qquad x \in \mathbb{R}^n$

Pros:

- DDP takes $\approx$ Newton Steps on L w/ ever computing $\nabla^2 L$
  - Efficient: $O(n^3 N)$ operations per iteration
  - Rapid convergence: Converges Quadratically when $\bar{U}_{1:N}$ "near" optimal
- DDP gives local control for Free!

$$v_k = \bar{u}_k - Q_{uu}^{-1} Q_{uk} (x_k - \bar{x}_k)$$

locally optimal too!

Cons:

- Only local feedback
- Control contstaints ... possible
- State constraints - active research
- Sensitive
  - MS - DDP recent development