

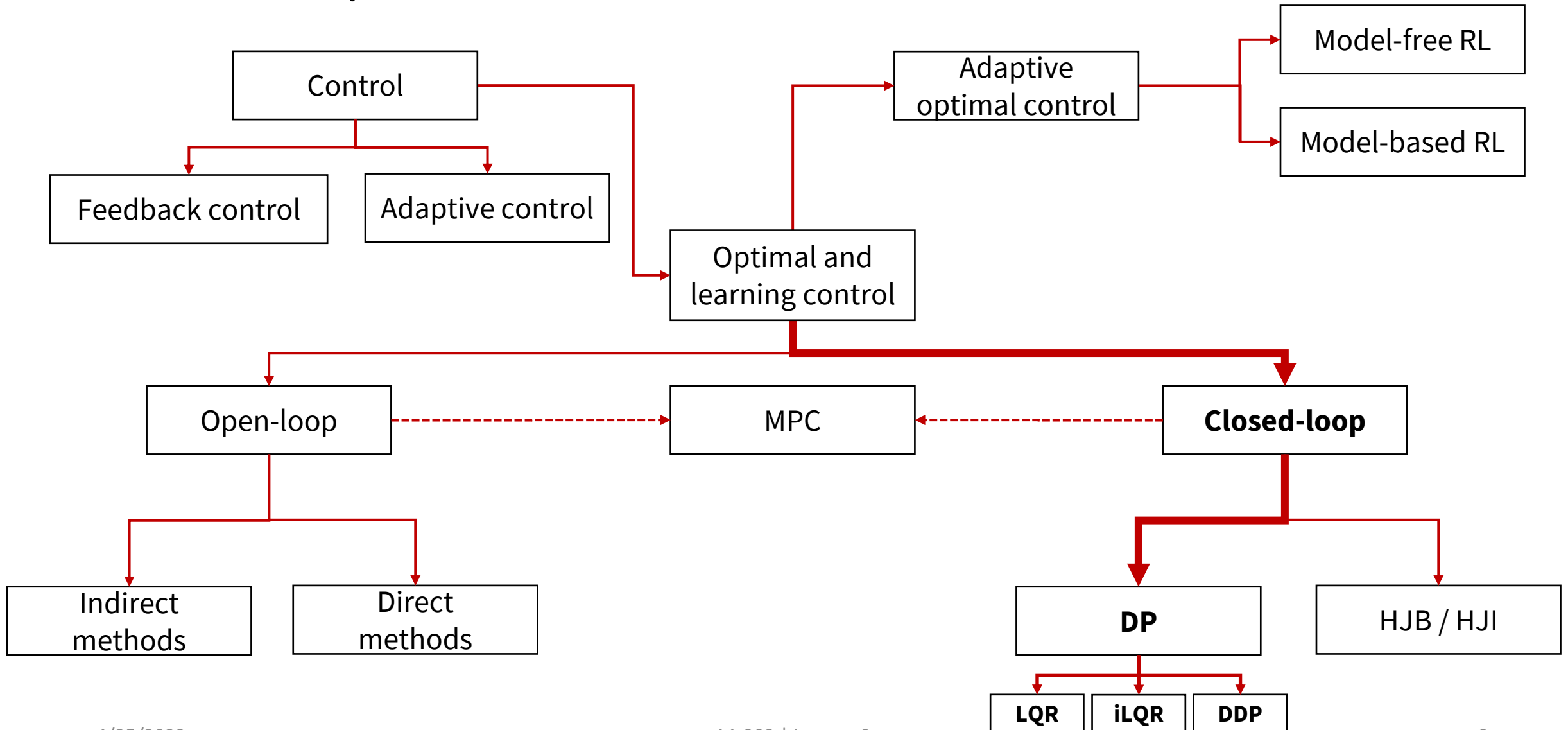
AA203

Optimal and Learning-based Control

Nonlinearity: tracking LQR, iterative LQR, differential dynamic programming



Roadmap



LQR-style algorithms for optimal control

- Linear tracking problems
- Nonlinear tracking problems
- Using LQR techniques to solve nonlinear optimal control problems
 - Iterative LQR
 - Differential dynamic programming
- Readings: [notes Section 3.1, 3.2](#) and references therein

Recapping LQR

- Minimize

$$J_0(\mathbf{x}_0) = \frac{1}{2} \mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2} \sum_{k=0}^{N-1} (\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2 \mathbf{x}_k^T H_k \mathbf{u}_k)$$

$$\text{s.t. } \mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \quad k \in \{0, 1, \dots, N-1\}$$

- Solved efficiently using dynamic programming by computing value function:

$$J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} Q_k & H_k \\ H_k^T & R_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} + \right. \\ \left. (A_k \mathbf{x}_k + B_k \mathbf{u}_k)^T P_{k+1} (A_k \mathbf{x}_k + B_k \mathbf{u}_k) \right)$$

- Result:

$$\pi_k^*(\mathbf{x}_k) = L_k \mathbf{x}_k$$

$$J_k^*(\mathbf{x}_k) = \frac{1}{2} \mathbf{x}_k^T P_k \mathbf{x}_k$$

Recapping LQR

- Minimize

$$J_0(\mathbf{x}_0) = \frac{1}{2} \mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2} \sum_{k=0}^{N-1} (\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2 \mathbf{x}_k^T H_k \mathbf{u}_k)$$

$$\text{s.t. } \mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k, \quad k \in \{0, 1, \dots, N-1\}$$

- Solved efficiently using dynamic programming by computing value function:

$$J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k} \frac{1}{2} \left(\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} Q_k & H_k \\ H_k^T & R_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} + \underbrace{(A_k \mathbf{x}_k + B_k \mathbf{u}_k)^T P_{k+1} (A_k \mathbf{x}_k + B_k \mathbf{u}_k)} \right)$$

- Result:

$$\pi_k^*(\mathbf{x}_k) = L_k \mathbf{x}_k$$

$$J_k^*(\mathbf{x}_k) = \frac{1}{2} \mathbf{x}_k^T P_k \mathbf{x}_k$$

Q function (state-action value function)

Recapping LQR

- Can also generalize cost (adding linear/constant terms), and dynamics (adding affine term)

Minimize

$$J_0(\mathbf{x}_0) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_N \\ 1 \end{bmatrix}^T \begin{bmatrix} Q_N & \mathbf{q}_N \\ \mathbf{q}_N^T & 2c_N \end{bmatrix} \begin{bmatrix} \mathbf{x}_N \\ 1 \end{bmatrix} + \frac{1}{2} \sum_{k=0}^{N-1} \left(\begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}^T \begin{bmatrix} Q_k & \mathbf{q}_k \\ \mathbf{q}_k^T & 2c_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} + \mathbf{u}_k^T R_k \mathbf{u}_k + 2 \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}^T \begin{bmatrix} H_k \\ \mathbf{r}_k^T \end{bmatrix} \mathbf{u}_k \right)$$

subject to dynamics

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A_k & \mathbf{d}_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} + \begin{bmatrix} B_k \\ 0 \end{bmatrix} \mathbf{u}_k$$

$$\rightarrow \pi_k^*(\mathbf{x}_k) = \begin{bmatrix} L_k & \ell_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$

$$J_k^*(\mathbf{x}_k) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}^T \begin{bmatrix} P_k & \mathbf{p}_k \\ \mathbf{p}_k^T & 2p_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}$$

Linear tracking problems

- Imagine you are given a *nominal trajectory*

$$(\bar{\mathbf{x}}_0, \dots, \bar{\mathbf{x}}_N), (\bar{\mathbf{u}}_0, \dots, \bar{\mathbf{u}}_{N-1})$$

- Assume nominal trajectory satisfies linear dynamics
- Linear tracking problem: find policy to minimize cost

$$\frac{1}{2}(\mathbf{x}_N - \bar{\mathbf{x}}_N)^T Q_N(\mathbf{x}_N - \bar{\mathbf{x}}_N) + \frac{1}{2} \sum_{k=0}^{N-1} [(\mathbf{x}_k - \bar{\mathbf{x}}_k)^T Q(\mathbf{x}_k - \bar{\mathbf{x}}_k) + (\mathbf{u}_k - \bar{\mathbf{u}}_k)^T R(\mathbf{u}_k - \bar{\mathbf{u}}_k)]$$

- Then define *deviation variables*

$$\delta \mathbf{x}_k := \mathbf{x}_k - \bar{\mathbf{x}}_k \text{ and } \delta \mathbf{u}_k := \mathbf{u}_k - \bar{\mathbf{u}}_k$$

and solve standard LQR with respect to deviation variables

Nonlinear tracking problems

- Imagine you are given a *feasible nominal trajectory*

$$(\bar{\mathbf{x}}_0, \dots, \bar{\mathbf{x}}_N), (\bar{\mathbf{u}}_0, \dots, \bar{\mathbf{u}}_{N-1})$$

- The tracking cost is still quadratic, but the dynamics are now nonlinear

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$$

- To apply LQR, we can linearize around the nominal trajectory

$$\mathbf{x}_{k+1} \approx f(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k) + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}_{A_k} \overbrace{(\mathbf{x}_k - \bar{\mathbf{x}}_k)}^{\delta \mathbf{x}_k} + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}_{B_k} \overbrace{(\mathbf{u}_k - \bar{\mathbf{u}}_k)}^{\delta \mathbf{u}_k}$$

- And apply LQR to the deviation variables (with dynamics $\delta \mathbf{x}_{k+1} = A_k \delta \mathbf{x}_k + B_k \delta \mathbf{u}_k$)

Nonlinear optimal control problem

- Consider now nonlinear optimal control problem

$$\min_{\mathbf{u}} \sum_{k=0}^{N-1} c(\mathbf{x}_k, \mathbf{u}_k)$$

subject to $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$

- Can we apply LQR-techniques to approximately solve it?

Iterative LQR

- Imagine you are given a *feasible nominal trajectory*

$$(\bar{\mathbf{x}}_0, \dots, \bar{\mathbf{x}}_N), (\bar{\mathbf{u}}_0, \dots, \bar{\mathbf{u}}_{N-1})$$

- Linearize the dynamics around feasible trajectory

$$\mathbf{x}_{k+1} \approx \underbrace{f(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}_{\bar{\mathbf{x}}_{k+1}} + \underbrace{f_{\mathbf{x}}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}_{A_k} \delta \mathbf{x}_k + \underbrace{f_{\mathbf{u}}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}_{B_k} \delta \mathbf{u}_k$$

- And Taylor expand cost function around feasible trajectory

$$c(\delta \mathbf{x}_k, \delta \mathbf{u}_k) \approx c_k + \underbrace{c_{\mathbf{x},k}^T}_{\mathbf{q}_k} \delta \mathbf{x}_k + \underbrace{c_{\mathbf{u},k}^T}_{\mathbf{r}_k} \delta \mathbf{u}_k + \frac{1}{2} \delta \mathbf{x}_k^T \underbrace{c_{\mathbf{xx},k}}_{Q_k} \delta \mathbf{x}_k + \frac{1}{2} \delta \mathbf{u}_k^T \underbrace{c_{\mathbf{uu},k}}_{R_k} \delta \mathbf{u}_k + \delta \mathbf{x}_k^T \underbrace{c_{\mathbf{xu},k}}_{H_k} \delta \mathbf{u}_k$$

Iterative LQR

- By optimizing over deviation variables (using results for LQR with cross-quadratic cost & affine dynamics), we obtain new solution:

$$\{\bar{\mathbf{x}}_k + \delta \mathbf{x}_k^*\} \text{ and } \{\bar{\mathbf{u}}_k + \delta \mathbf{u}_k^*\}$$

- We can then re-linearize and Taylor expand around this new trajectory, and iterate!

Iterative LQR

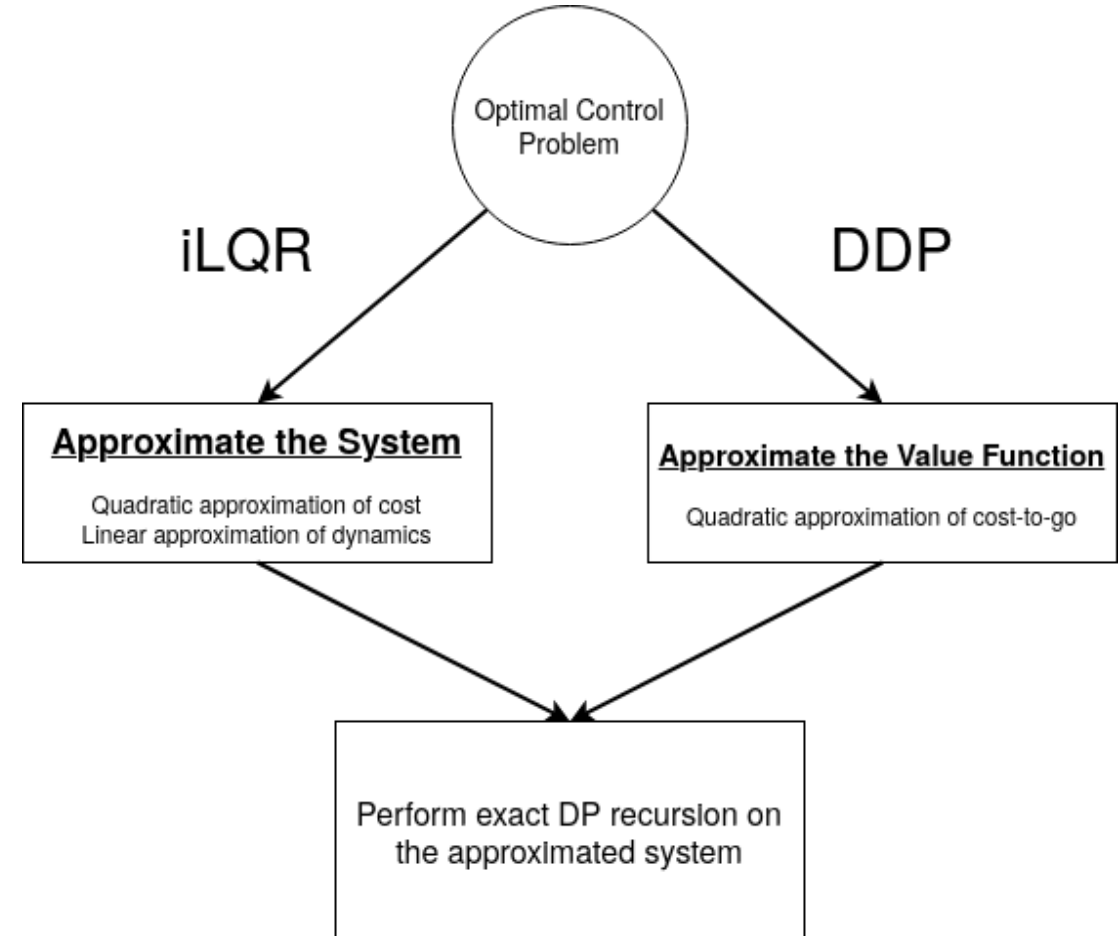
- Backward pass ($k = N$ to 0):
 - Compute locally linear dynamics, locally quadratic cost around nominal trajectory
 - Solve local approximation of DP recursion to compute control law
 - Compute cost-to-go
- Forward pass ($k = 0$ to N):
 - Use optimal control policy to update nominal trajectory
 - Propagate full nonlinear dynamics f , not the linearized approximate dynamics!
- Iterate until convergence

Algorithmic details

- Need to make sure that the new state / control stay close to the linearization point
 - Add extra penalty on deviations
 - Apply a line search on policy rollout
- Need to decide on termination criterion
 - For example, one can stop when cost improvement is “small”
- Method can get stuck in local minima → “good” initialization is often critical
- Cost matrices may not be positive definite
 - Regularize them until they are
- Great collection of tips/tricks: [Yuval Tassa’s thesis](#) (Section 2.2.3)

Differential Dynamic Programming (DDP)

- iLQR first approximates dynamics and cost, then performs exact DP recursion
- DDP instead approximates DP recursion directly



Differential Dynamic Programming (DDP)

In detail, consider the change in cost to go at timestep k under a perturbation $(\delta \mathbf{x}_k, \delta \mathbf{u}_k)$

$$Q_k(\delta \mathbf{x}_k, \delta \mathbf{u}_k) := c(\bar{\mathbf{x}}_k + \delta \mathbf{x}_k, \bar{\mathbf{u}}_k + \delta \mathbf{u}_k) + J_{k+1}(f(\bar{\mathbf{x}}_k + \delta \mathbf{x}_k, \bar{\mathbf{u}}_k + \delta \mathbf{u}_k))$$

Using a 2nd order Taylor Expansion,

$$Q_k(\delta \mathbf{x}_k, \delta \mathbf{u}_k) \approx Q_k(0, 0) + \nabla Q_k^T \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix} \nabla^2 Q_k \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix}$$

Differential Dynamic Programming (DDP)

The optimal control perturbation is

$$\delta \mathbf{u}_k^* = \operatorname{argmin}_{\delta \mathbf{u}} Q(\delta \mathbf{x}_k, \delta \mathbf{u})$$

Expanding the approximation, one gets

$$\begin{aligned} Q_k(\delta \mathbf{x}_k, \delta \mathbf{u}_k) \approx & Q_k(0, 0) + \underbrace{Q_{x,k}^\top \delta \mathbf{x}_k + Q_{u,k}^\top \delta \mathbf{u}_k}_{\text{first order terms}} \\ & + \underbrace{\frac{1}{2} \delta \mathbf{x}_k^\top Q_{xx,k} \delta \mathbf{x}_k + \frac{1}{2} \delta \mathbf{u}_k^\top Q_{uu,k} \delta \mathbf{u}_k + \delta \mathbf{x}_k^\top Q_{xu,k} \delta \mathbf{u}_k}_{\text{second order terms}} \end{aligned}$$

Differential Dynamic Programming (DDP)

Apply conditions for optimality (gradient equal to zero):

$$Q_{u,k} + Q_{ux,k}\delta\mathbf{x}_k + Q_{uu,k}\delta\mathbf{u}_k = 0$$

$$\implies \delta\mathbf{u}_k^* = -Q_{uu,k}^{-1}Q_{u,k} - Q_{uu,k}^{-1}Q_{ux,k}\delta\mathbf{x}_k$$

As was the case with LQR, the optimal control has the form

$$\delta\mathbf{u}_k^* = \mathbf{l}_k + \mathbf{L}_k\delta\mathbf{x}_k$$

Algorithm proceeds via same forward/backward passes as iLQR

iLQR vs. DDP

Quadratic approximations for the state-action value function (Q function):

$$Q_k = c_k + v_{k+1}$$

$$Q_{\mathbf{x},k} = c_{\mathbf{x},k} + f_{\mathbf{x},k}^T v_{k+1}$$

$$Q_{\mathbf{u},k} = c_{\mathbf{u},k} + f_{\mathbf{u},k}^T v_{k+1}$$

$$Q_{\mathbf{xx},k} = c_{\mathbf{xx},k} + f_{\mathbf{x},k}^T V_{k+1} f_{\mathbf{x},k} + \mathbf{v}_{k+1} \cdot f_{\mathbf{xx},k}$$

$$Q_{\mathbf{uu},k} = c_{\mathbf{uu},k} + f_{\mathbf{u},k}^T V_{k+1} f_{\mathbf{u},k} + \mathbf{v}_{k+1} \cdot f_{\mathbf{uu},k}$$

$$Q_{\mathbf{ux},k} = c_{\mathbf{ux},k} + f_{\mathbf{u},k}^T V_{k+1} f_{\mathbf{x},k} + \mathbf{v}_{k+1} \cdot f_{\mathbf{ux},k}$$

DDP contains second-order dynamics derivatives compared to iLQR

Next time

- Direct methods for trajectory optimization
- Sequential convex programming