AA203
Optimal and Learning-based Control

Nonlinearity: tracking LQR, iterative LQR, differential dynamic programming
Intro to direct methods for optimal control

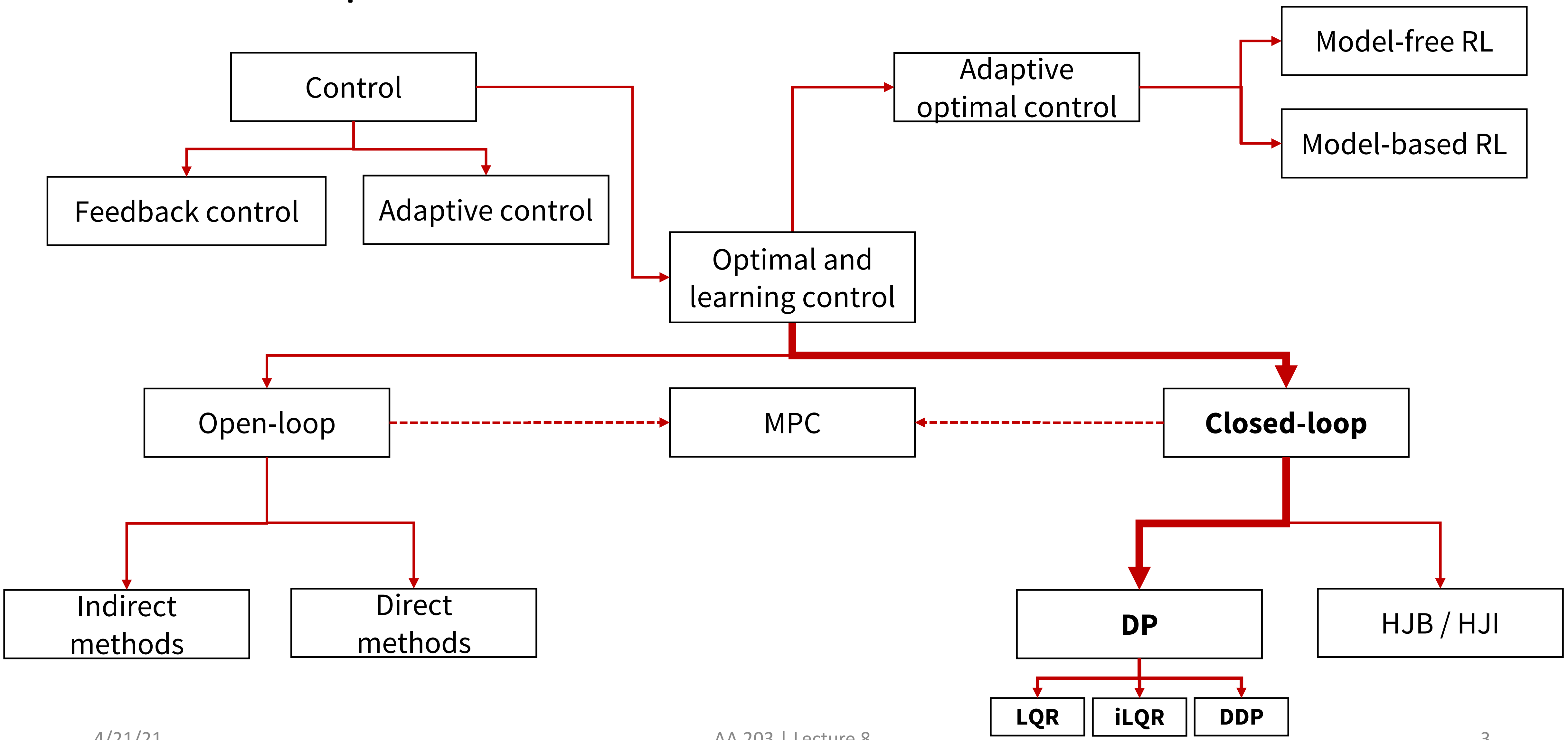
Stanford University

Autonomous Systems Lab

Logistics

- Recitation 3 (regression models) tomorrow 10:30—11:50AM
- HW2 out, due Monday 5/3
  - A bit less involved than HW1, but relevant to many projects
- Project feedback will be released tonight
  - Midterm report due Friday 5/7, but nail down ASAP:
    "A precise statement of the project setting you are considering in your project."
- 1/3-quarter feedback form is open until Sunday 4/25

Roadmap


Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI
LQR
iLQR
DDP

LQR-style algorithms for optimal control

- Linear tracking problems
- Non-linear tracking problems
- Using LQR techniques to solve non-linear optimal control problems
  - Iterative LQR
  - Differential dynamic programming

- Readings: notes sections 3.1, 3.2 and references therein

Recapping LQR

- Minimize

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

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

- Solved efficiently using dynamic programming by computing value function:

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

- 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

- 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

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

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

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

Linear tracking problems

- Imagine you are given a *nominal trajectory*

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

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

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

- Then define dev*iation variables*

$$\delta\boldsymbol{x}_k \coloneqq \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k \text{ and } \delta\boldsymbol{u}_k \coloneqq \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k$$

and solve standard LQR with respect to deviation variables

Nonlinear tracking problems

- Imagine you are given a *feasible nominal trajectory*

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

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

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

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

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

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

Nonlinear optimal control problem

- Consider now nonlinear optimal control problem

$$\min_{\mathbf{u}} \sum_{k=0}^{N-1} c(\mathbf{x}_k, \mathbf{u}_k)$$
$$\text{subject to } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$$

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

Iterative LQR

- Imagine you are given a *feasible nominal trajectory*

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

- Linearize the dynamics around feasible trajectory

$$\boldsymbol{x}_{k+1} \approx f(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + \frac{\partial f}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + \frac{\partial f}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k)$$

- And Taylor expand cost function around feasible trajectory

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

Iterative LQR

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

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

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

Iterative LQR

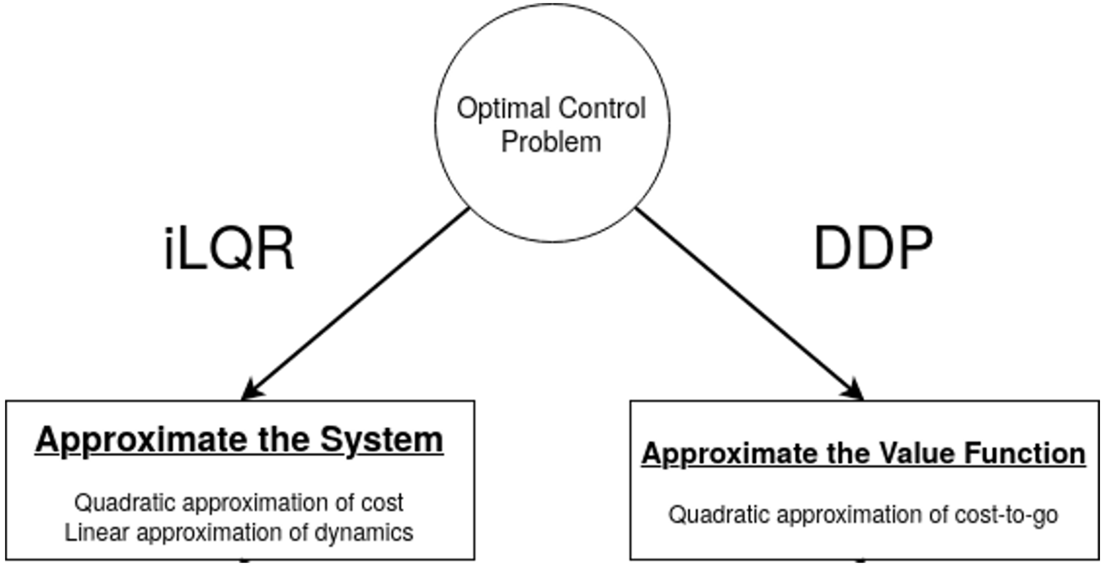
- Backward pass ($k = N$ to 0):
  - Compute locally linear dynamics, locally quadratic cost around nominal trajectory
  - Solve local approximation of DP recursion to compute control law
  - Compute cost-to-go
- Forward pass ($k = 0$ to N):
  - Use control law to update nominal trajectory
- Iterate until 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 $(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k)$

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

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

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

Expanding the approximation, one gets

$$Q_k(\delta \boldsymbol{x}_k, \delta \boldsymbol{u}_k) \approx Q_k(0,0) + \underbrace{Q_{x,k}^\top \delta \boldsymbol{x}_k + Q_{u,k}^\top \delta \boldsymbol{u}_k}_{\text{first order terms}}$$

$$+ \underbrace{\frac{1}{2}\delta \boldsymbol{x}_k^\top Q_{xx,k} \delta \boldsymbol{x}_k + \frac{1}{2}\delta \boldsymbol{u}_k^\top Q_{uu,k} \delta \boldsymbol{u}_k + \delta \boldsymbol{x}_k^\top Q_{xu,k} \delta \boldsymbol{u}_k}_{\text{second order terms}}$$

Differential Dynamic Programming (DDP)

Apply conditions for optimality (gradient equal to zero):

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

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

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

$$\delta \boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \delta \boldsymbol{x}_k$$

Algorithm proceeds via same forward/backward passes as iLQR

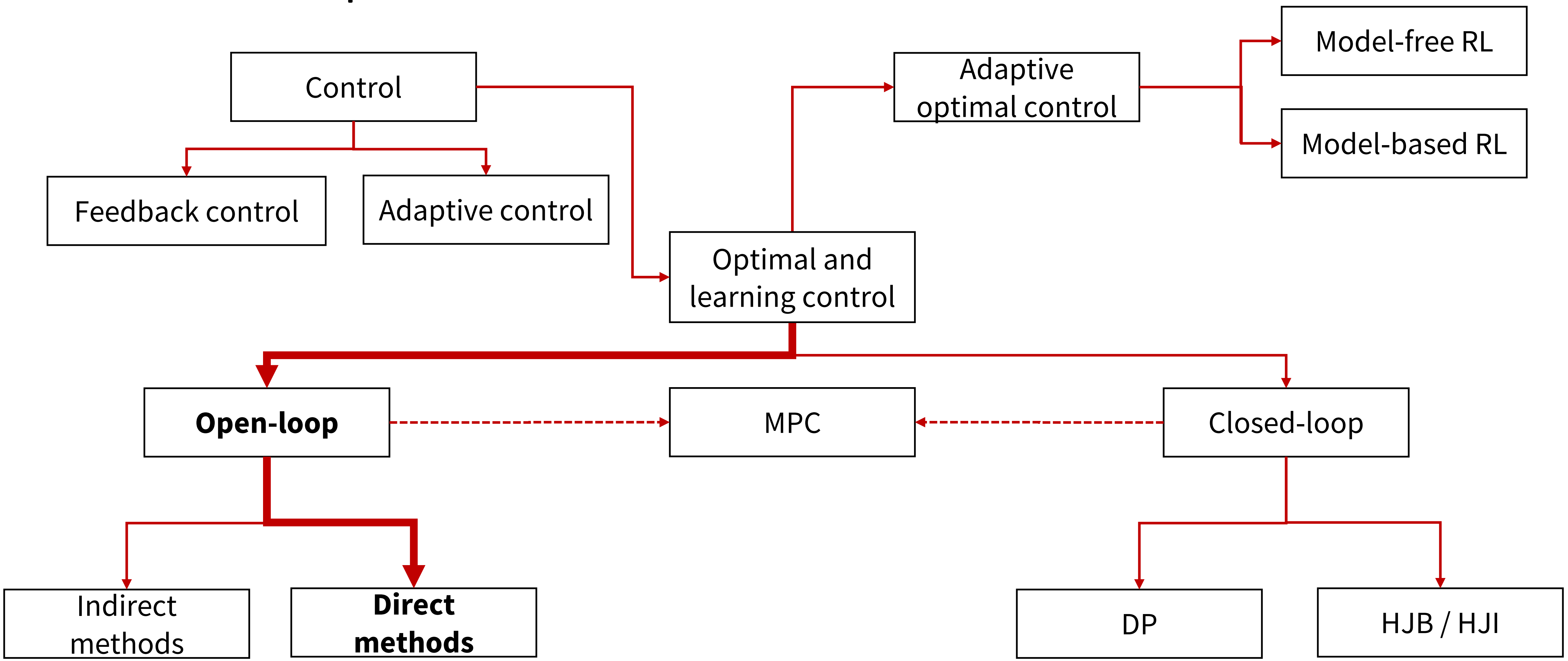
iLQR vs. DDP

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

$$
\begin{aligned}
Q_k &= c_k + v_{k+1} \\
Q_{\mathbf{x},k} &= c_{\mathbf{x},k} + f_{\mathbf{x},k}^T \boldsymbol{v}_{k+1} \\
Q_{\mathbf{u},k} &= c_{\mathbf{u},k} + f_{\mathbf{u},k}^T \boldsymbol{v}_{k+1} \\
Q_{\mathbf{xx},k} &= c_{\mathbf{xx},k} + f_{\mathbf{x},k}^T V_{k+1} f_{\mathbf{x},k} + \boldsymbol{v}_{k+1} \cdot f_{\mathbf{xx},k} \\
Q_{\mathbf{uu},k} &= c_{\mathbf{uu},k} + f_{\mathbf{u},k}^T V_{k+1} f_{\mathbf{u},k} + \boldsymbol{v}_{k+1} \cdot f_{\mathbf{uu},k} \\
Q_{\mathbf{ux},k} &= c_{\mathbf{ux},k} + f_{\mathbf{u},k}^T V_{k+1} f_{\mathbf{x},k} + \boldsymbol{v}_{k+1} \cdot f_{\mathbf{ux},k}
\end{aligned}
$$

DDP contains second-order dynamics derivatives compared to iLQR

Roadmap


Control
Feedback control
Adaptive control
Optimal and learning control
Adaptive optimal control
Model-free RL
Model-based RL
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

Optimal control problem

(**OCP**)

$$\min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$$

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t),\ t \in [0, t_f]$$

$$\mathbf{x}(0) = \mathbf{x}_0$$
$$\mathbf{x}(t_f) \in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\}$$

$$\mathbf{u}(t) \in U \subseteq \mathbb{R}^m, \quad t \in [0, t_f]$$

For simplicity:

- We assume the terminal cost h is equal to 0
- We assume $t_0 = 0$

- Direct Methods:
  1. Transcribe (**OCP**) into a nonlinear, constrained optimization problem
  2. Solve the optimization problem via nonlinear programming

- Indirect Methods:
  1. Apply necessary conditions for optimality to (**OCP**)
  2. Solve a two-point boundary value problem

Transcription into nonlinear programming

$$\min \int_0^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$$

(OCP)

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t),\ \ t \in [0, t_f]$$

$$\mathbf{x}(0) = \ \mathbf{x}_0$$

$$\mathbf{x}(t_f) \in M_f = \{\mathbf{x} \in \mathbb{R}^n : F(\mathbf{x}) = 0\}$$

$$\mathbf{u}(t) \in U \subseteq \mathbb{R}^m,\ \ \ t \in [0, t_f]$$

Forward Euler time discretization

1. Select a discretization $0 = t_0 < t_1 < \cdots < t_N = t_f$ for the interval $[0, t_f]$ and, for every $i = 0, \ldots, N-1$, define $\mathbf{x}_i \sim \mathbf{x}(t)$, $\mathbf{u}_i \sim \mathbf{u}(t)$, $t \in [t_i, t_{i+1})$ and $\mathbf{x}_0 \sim \mathbf{x}(0)$
2. By denoting $h_i = t_{i+1} - t_i$, (**OCP**) is transcribed into the following nonlinear, constrained optimization problem

$$\min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i)$$

(NLOP)

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, t_i), \qquad i = 0, \ldots, N-1$$

$$\mathbf{u}_i \in U\ , i = 0, \ldots, N-1\ , \qquad F(\mathbf{x}_N) = 0$$

Next time

- Examples of direct transcription
- Direct collocation
- Sequential convex programming