Introduction to Optimal Control

ORF523 CONVEX AND CONIC OPTIMIZATION

SUMEET SINGH, GOOGLE BRAIN ROBOTICS

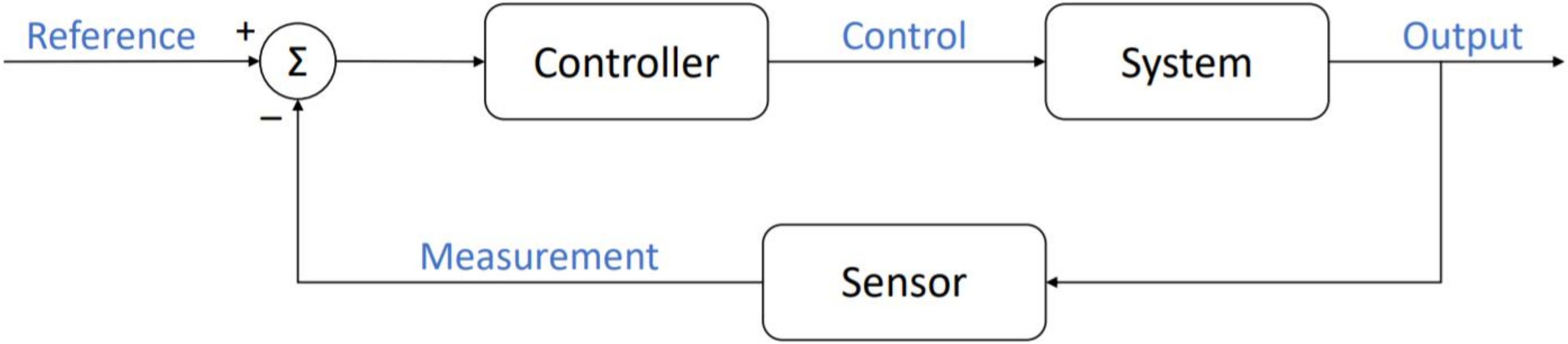
APRIL 22, 2021

Outline

- Optimal Control Problem
- Open- vs Closed-Loop Solutions
- Closed-Loop: Bellman’s Principle of Optimality & Dynamic Programming
  - Finite spaces
  - Continuous spaces – LQ control
- Open-Loop:
  - Gradient descent
  - Newton descent
  - DDP
- Model Predictive Control

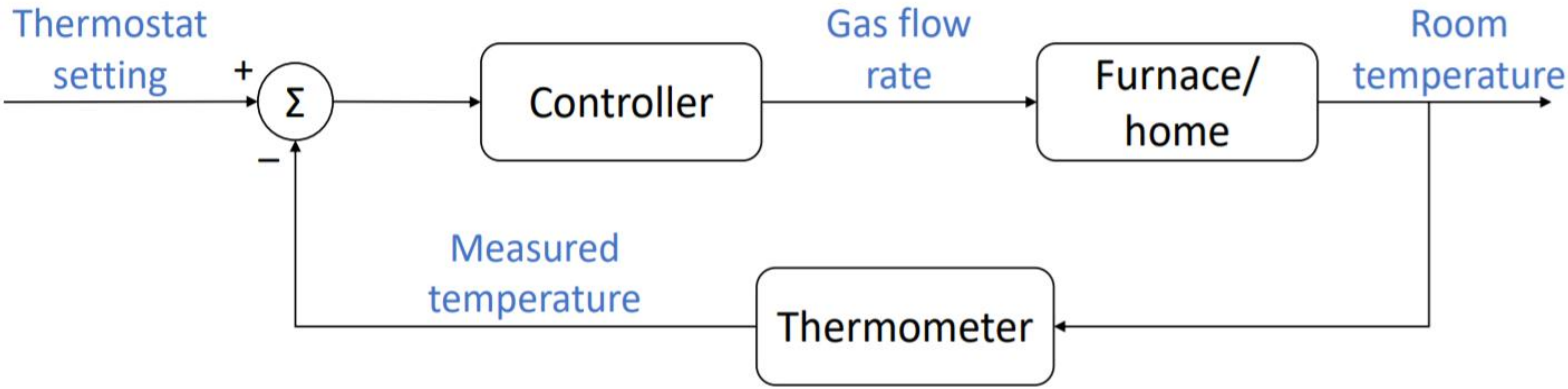
Feedback Control

- Consider block diagram for tracking some reference signal.



Feedback Control

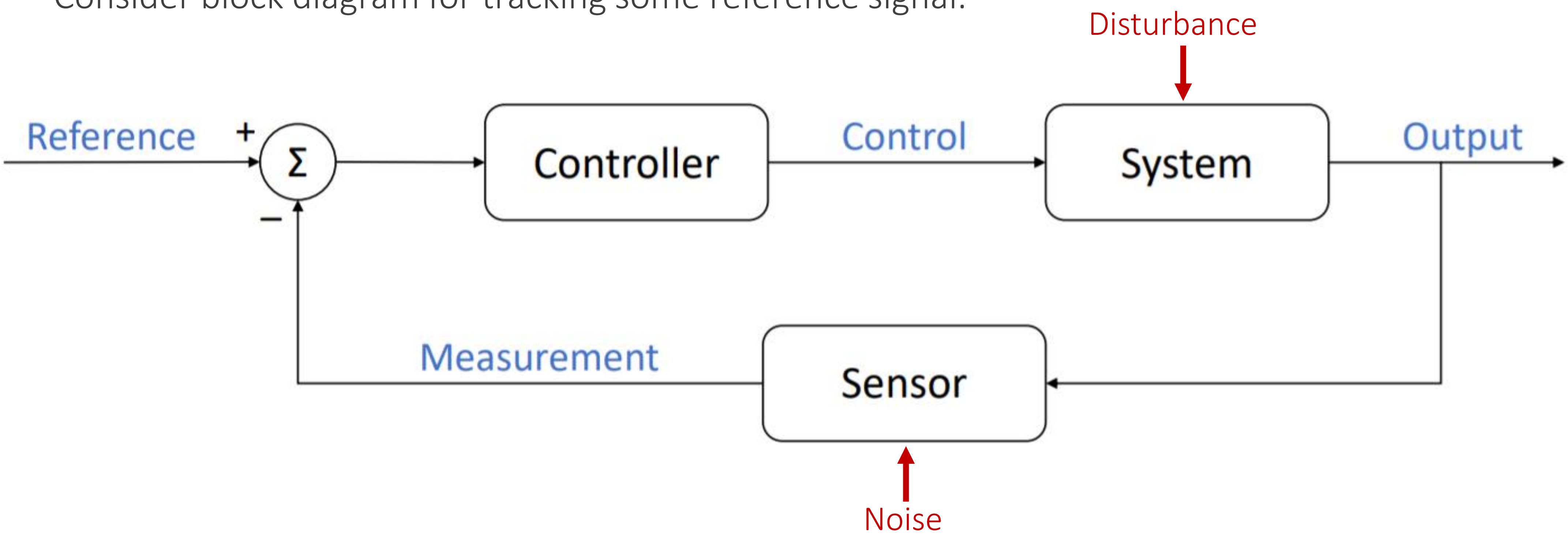
- Consider block diagram for tracking some reference signal.



Adapted from: Stanford AA203, Spring 2021

Feedback Control

- Consider block diagram for tracking some reference signal.



Adapted from: Stanford AA203, Spring 2021

Feedback Control Objectives

- *Stability*: various formulations; loosely, system output is "under control"
- *Tracking:* output should track reference "as close as possible"
- *Disturbance rejection:* output should be "as insensitive as possible" to disturbances/noise
- *Robustness:* controller should still perform well up to "some degree of model misspecification"

Adapted from: Stanford AA203, Spring 2021

What’s Missing?

- *Performance*: some mathematical quantification of all these objectives and control that realizes the tradeoffs

- *Planning:* providing an appropriate **reference trajectory** to track (can be highly non-trivial)

- *Learning:* adaptation to unknown properties of the system

Adapted from: Stanford AA203, Spring 2021

ht Statistics

Top Speed: $1.93\,\mathrm{m/s}$

Max Drag: $0.55\,\mathrm{m/s^2}$

What’s Missing?

- *Performance*: some mathematical quantification of all these objectives and control that realizes the tradeoffs

- *Planning:* providing an appropriate **reference trajectory** to track (can be highly non-trivial)

- *Learning:* adaptation to unknown properties of the system

Optimal Control Problem

3 Key Ingredients:

- Mathematical description of the system to be controlled
- Specification of a performance criterion
- Specification of constraints

State-Space Models

$$\dot{x}_1(t) = f_1(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t)$$
$$\dot{x}_2(t) = f_2(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t)$$
$$\vdots \qquad\qquad \vdots$$
$$\dot{x}_n(t) = f_n(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t)$$

Where

- $x_1(t), x_2(t), \ldots, x_n(t)$ are the state variables
- $u_1(t), u_2(t), \ldots, u_m(t)$ are the control variables

Adapted from: Stanford AA203, Spring 2021

State-Space Models

In compact form:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$$

- A history of control input values during the interval [0, T] is called a *control history*, denoted by **u**
- A history of state values during the interval [0, T] is called a *state trajectory*, denoted by **x**

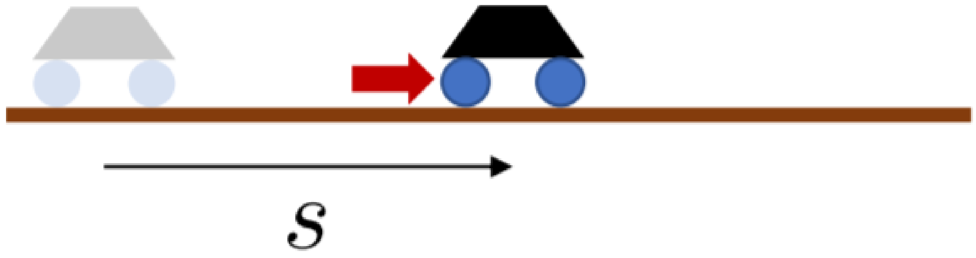
Adapted from: Stanford AA203, Spring 2021

Illustrative Example

- Double integrator: point mass under controlled acceleration

$$\ddot{s}(t) = a(t)$$

$$\begin{bmatrix} \dot{s} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ a \end{bmatrix}$$



Adapted from: Stanford AA203, Spring 2021

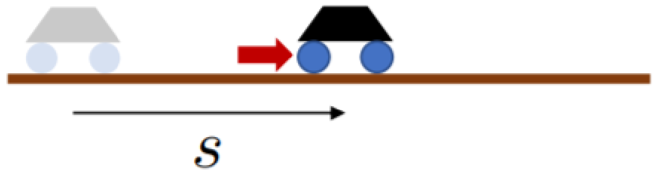
Illustrative Example

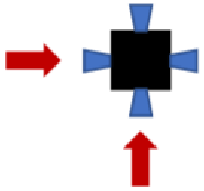
- Double integrator: point mass under controlled acceleration

$$\begin{bmatrix} \dot{s} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} a \end{bmatrix}$$

$$\dot{\mathbf{x}}(t) = A \quad \mathbf{x}(t) + B \quad \mathbf{u}(t)$$

$$\begin{bmatrix} \dot{\mathbf{s}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \begin{bmatrix} \mathbf{a} \end{bmatrix}$$





Adapted from: Stanford AA203, Spring 2021

Quantifying Performance

$$\min_{\mathbf{u}} \int_0^T \|x(t)\|^2 + \|u(t)\|^2 \, dt$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

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

Adapted from: Stanford AA203, Spring 2021

Quantifying Performance

$$\min_{\mathbf{u}} \int_0^T \|x(t)\|^2 + \|u(t)\|^2 \, dt$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

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

$$\mathbf{x}(T) = \mathbf{x}_f$$

Adapted from: Stanford AA203, Spring 2021

Quantifying Performance

$$
\begin{aligned}
\min_{\mathbf{u}} \quad & \int_0^T \mathbf{x}(t)^T Q \mathbf{x}(t) + \mathbf{u}(t)^T R \mathbf{u}(t) \; dt \\
\text{s.t.} \quad & \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \\
& \mathbf{x}(0) = \mathbf{x}_0 \\
& \mathbf{x}(T) = \mathbf{x}_f
\end{aligned}
$$

Adapted from: Stanford AA203, Spring 2021

Quantifying Performance

- More generally:

Instantaneous or stage-wise cost

$$J(\mathbf{u}, \mathbf{x}) = \int_0^T \underbrace{l(t, \mathbf{u}(t), \mathbf{x}(t))}_{\text{Instantaneous or stage-wise cost}} dt + \underbrace{\phi(\mathbf{x}(T))}_{\text{Terminal Cost}}$$

Terminal Cost

- l and ϕ are scalar functions, and T may be specified or “free”

Constraints

- Initial and final conditions (boundary conditions):

$$\mathbf{x}(0) = x_0 \qquad \mathbf{x}(T) = x_f$$

- Constraints on state trajectory:

$$\underline{X} \leq \mathbf{x}(t) \leq \overline{X}$$

- Control limits:

$$\underline{U} \leq \mathbf{u}(t) \leq \overline{U}$$

- A control history and state trajectory that satisfy the control & state constraints for the entire time interval are termed *admissible*

Adapted from: Stanford AA203, Spring 2021

The Optimal Control Problem

Definitions: State: $\mathbf{x} \in \mathbb{R}^n$, Control: $\mathbf{u} \in \mathbb{R}^m$

Continuous Time:

- Performance measure (minimize): $J(\mathbf{u}, \mathbf{x}) = \int_0^T l(t, \mathbf{u}(t), \mathbf{x}(t))dt + \phi(\mathbf{x}(T))$
- Dynamics: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$
- + other constraints, e.g., $\boldsymbol{u}(t) \in U,\ \boldsymbol{x}(t) \in X$
- $T < \infty$ or $T = \infty$

- Minimizer: $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ is an optimal solution pair.
- Existence & uniqueness not always guaranteed

The Optimal Control Problem

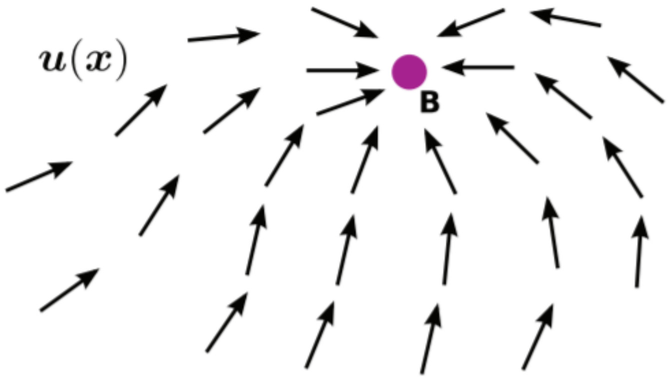
Discrete Time:

- Performance measure (minimize): $J(\mathbf{u}, \mathbf{x}) = \sum_{n=0}^{N-1} l(n, \mathbf{u}[n], \mathbf{x}[n]) + \phi(\mathbf{x}[N])$
- Dynamics: $\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n)$
- + other constraints
- $N < \infty$ or $N = \infty$

Solution Methods

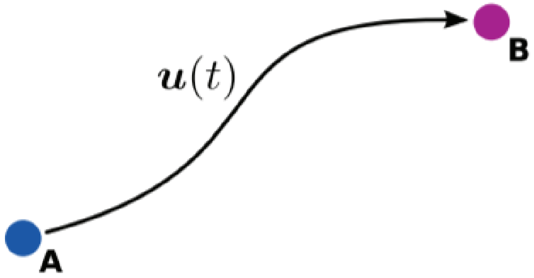
Dynamic Programming (Principle of Optimality)

- Compositionality of optimal paths
- **Closed-loop** solutions:
  find a solution for **all states at all times**



Calculus of Variations (Pontryagin Maximum/Minimum Principle)

- "Optimal curve should be such that neighboring curves don't lead to smaller costs" → "Derivative = 0"
- **Open-loop** solutions:
  find a solution for a **given initial state**



Figures: [Kelly, 2017]

The Optimal Control Problem

Continuous Time:

- Performance measure (minimize): $J(\mathbf{u}, \mathbf{x}) = \int_0^T l(t, \mathbf{u}(t), \mathbf{x}(t))dt + \phi(\mathbf{x}(T))$
- Dynamics: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$
- + other constraints, e.g., $\boldsymbol{u}(t) \in U$, $\boldsymbol{x}(t) \in X$

Closed-loop: find policy function $\pi^*(\boldsymbol{x}, t)$ s.t. $\boldsymbol{u}^*(t) = \pi^*(\boldsymbol{x}(t), t)$

Open-loop: given $\boldsymbol{x}(0) = \boldsymbol{x}_{\mathbf{0}}$, find optimal signals: $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, i.e., functions in $W^{1,\infty}[0, T]$ and $L^\infty[0, T]$

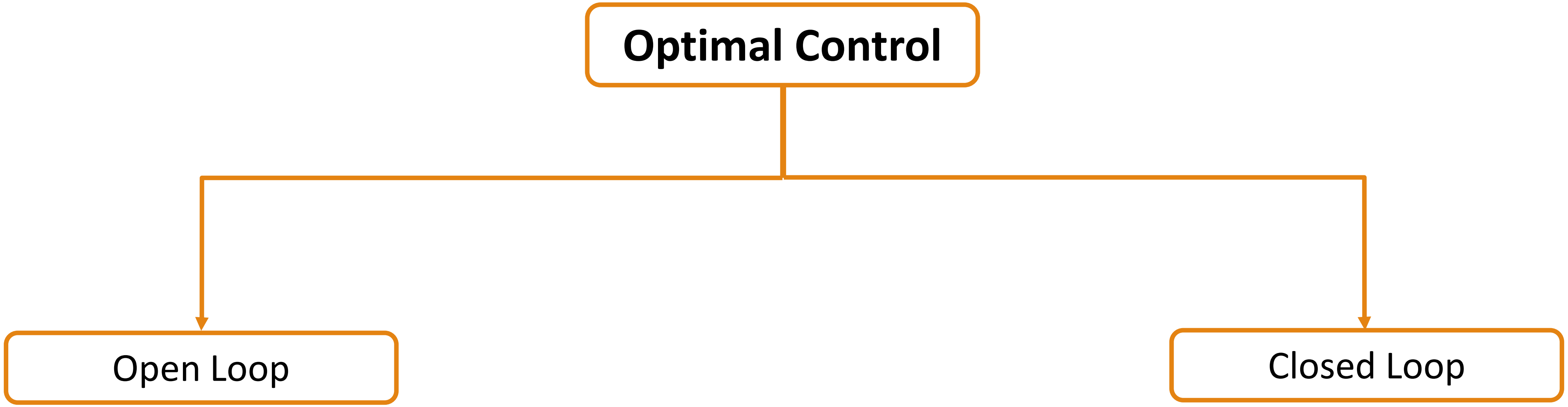
The Optimal Control Problem

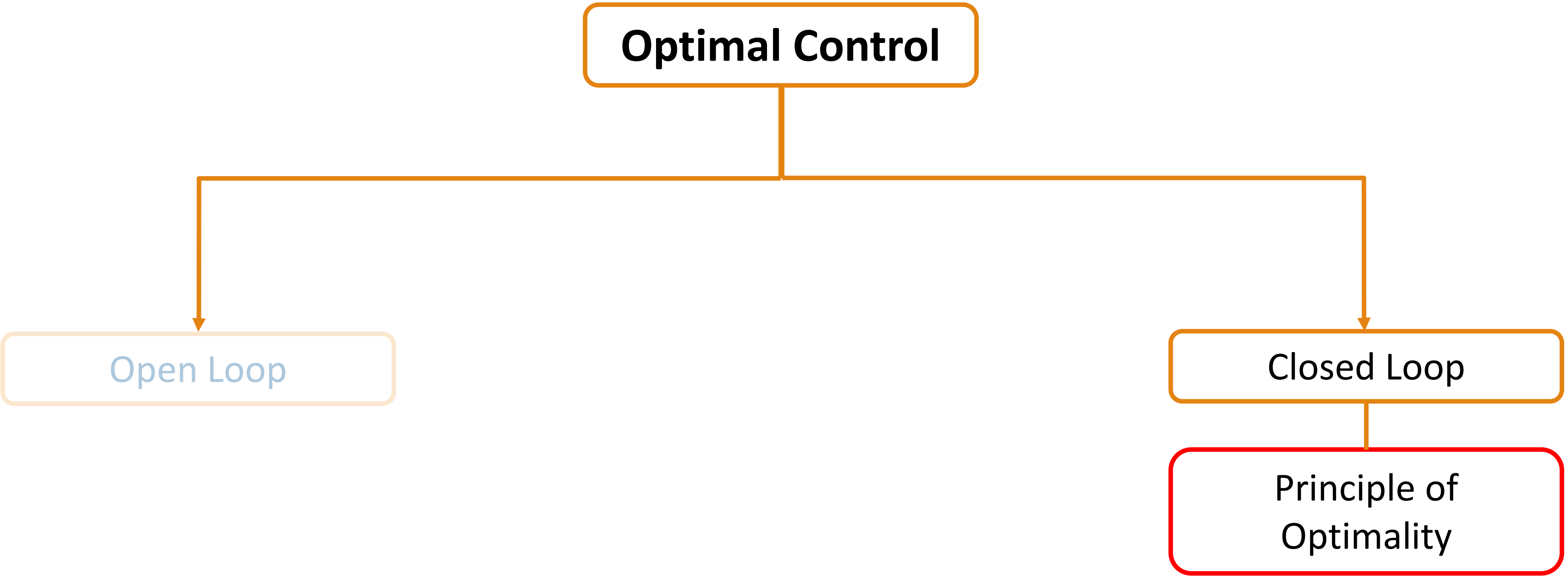
Discrete Time:

- Performance measure (minimize): $J(\mathbf{u}, \mathbf{x}) = \sum_{n=0}^{N-1} l(n, \mathbf{u}[n], \mathbf{x}[n]) + \phi(\mathbf{x}[N])$
- Dynamics: $\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n)$
- + other constraints

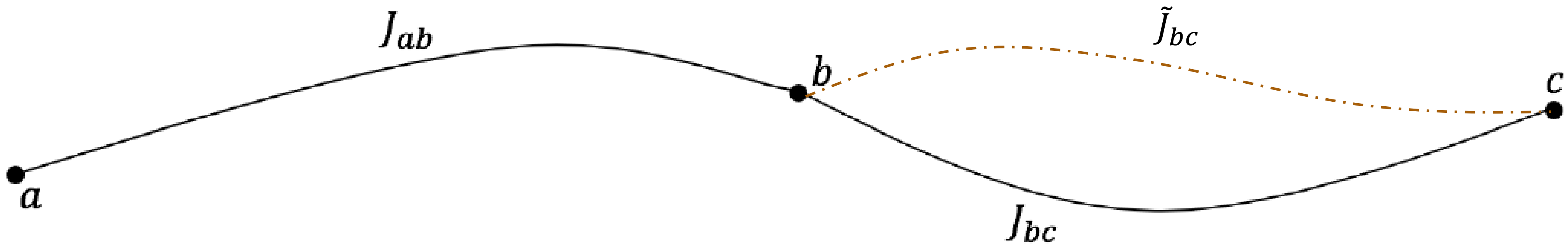
Closed-loop: find policy functions: $\{\pi_0^*, \dots, \pi_{N-1}^*\}$, s.t. $\boldsymbol{u}^*[n] = \pi_n^*(\boldsymbol{x}[n])$

Open-loop: Given $\boldsymbol{x}[0] = \boldsymbol{x}_0$, find optimal sequences: $(\boldsymbol{x}^*[\,], \boldsymbol{u}^*[\,])$.


Optimal Control
Open Loop
Closed Loop


Optimal Control
Open Loop
Closed Loop
Principle of Optimality

Principle of Optimality



Given trajectory from a→c, with cost $J_{ac} = J_{ab} + J_{bc}$ minimal, then J_{bc} minimal for path b→c.

Proof by contradiction:

- Assume there exists an alternative path b→c with lower cost $\tilde{J}_{bc} < J_{bc}$. Then, $\tilde{J}_{ac} = J_{ab} + \tilde{J}_{bc} < J_{ab} + J_{bc} = J_{ac}$, i.e., original path was not minimal.

[Bertsekas, 2017]

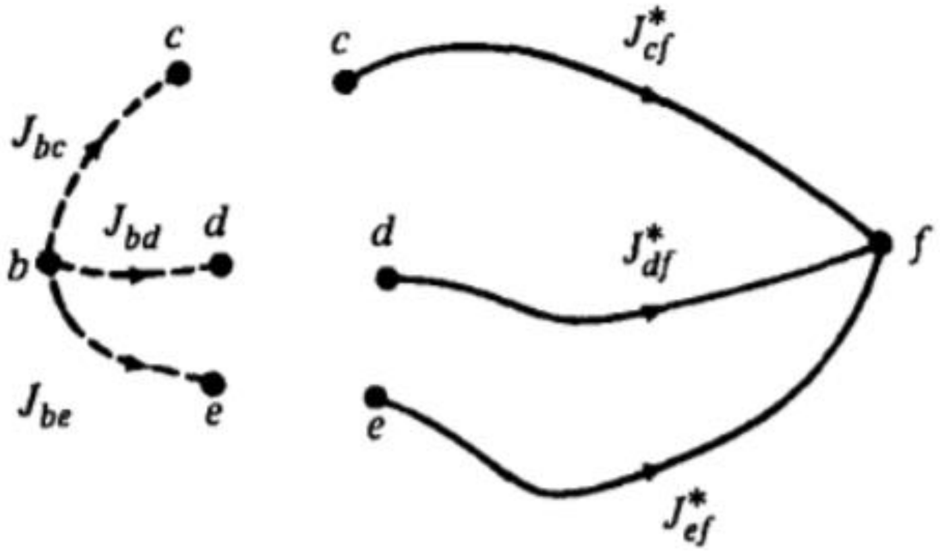
Principle of Optimality

Theorem (Discrete-time Principle of Optimality: Deterministic Case). *Let* $\pi^* = (\pi_0^*, \ldots, \pi_{N-1}^*)$ *be an optimal policy. Assume state* $\boldsymbol{x}_k$ *is reachable. Consider the subproblem whereby we are at* $\boldsymbol{x}_k$ *at time* k *and we wish to minimize the cost-to-go from time* k *to time* N*. Then the truncated policy* $(\pi_k^*, \ldots, \pi_{N-1}^*)$ *is optimal for the subproblem.*

Tail policies of an optimal policy are optimal for tail sub-problems.

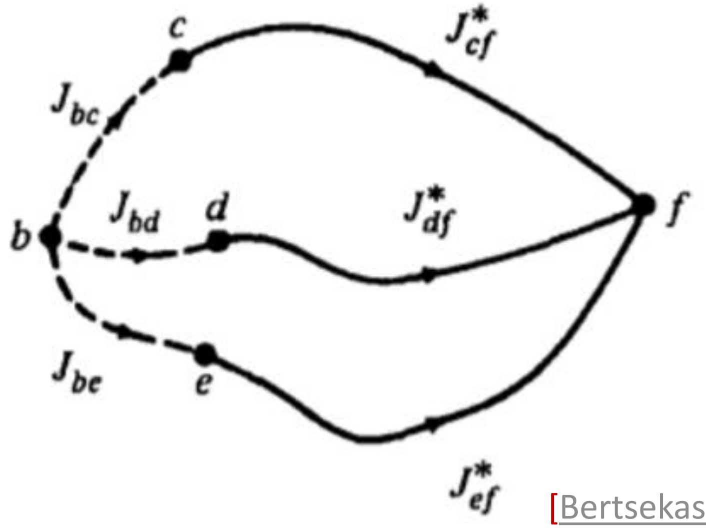
Applying Principle of Optimality

- Principle of Optimality: If b – c is the initial segment of the optimal path from b – f, then c – f is the terminal segment of this path.



- Thus, the optimal trajectory is found by comparing:

$$C_{bcf} = J_{bc} + J^*_{cf}$$
$$C_{bdf} = J_{bd} + J^*_{df}$$
$$C_{bef} = J_{be} + J^*_{ef}$$



[Bertsekas, 2017]

Applying Principle of Optimality

- Need only to compare **concatenation of immediate decisions** with **optimal** decisions
- In practice: carry out backwards in time.

Dynamic Programming (DP)

Performance measure: $$J(\mathbf{u}, \mathbf{x}) = \sum_{n=0}^{N-1} l(n, \mathbf{u}[n], \mathbf{x}[n]) + \phi(\mathbf{x}[N])$$

Dynamics: $$\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n)$$

Dynamic Programming recursion (Bellman Recursion) proceeds backwards:

$$J_N^*(\mathbf{x}[N]) = \phi(\mathbf{x}[N])$$

$$J_n^*(\mathbf{x}[n]) = \min_{\mathbf{u}}[l(n, \mathbf{u}, \mathbf{x}[n]) + J_{n+1}^*(f_d(\mathbf{x}[n], \mathbf{u}, n))] \quad n = N-1, \ldots, 0$$

Optimization over sequence → sequence of one-step optimizations

DP – Stochastic Case

Stochastic dynamics: $\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n, \omega[n])$

Markovian assumption: $\omega[n] = \omega[n](\boldsymbol{x}[n], \boldsymbol{u}[n])$

i.e., disturbance at time n is only a function of the state and control at time n

Implications:

1. Distribution of next state depends only on current state and control: $\boldsymbol{x}[n+1] \sim P(\cdot|\boldsymbol{x}[n], \boldsymbol{u}[n])$

2. Sufficient to look for optimal policy at time n as a function of $\boldsymbol{x}[n]$ (and not as a function of the entire history before time n)

DP – Stochastic Case

Stochastic dynamics with Markovian property: $\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n, \omega[n])$

Performance: $\mathbb{E}_{\omega_{0:N-1}} \left[\sum_{n=0}^{N-1} l(n, \pi_n(\mathbf{x}[n]), \mathbf{x}[n]) + \phi(\mathbf{x}[N]) \right]$

Applying principle of optimality and exploiting linearity of expectation:

$$J_N^*(\mathbf{x}[N]) = \phi(\mathbf{x}[N])$$

$$J_n^*(\mathbf{x}[n]) = \min_{\mathbf{u}} \mathbb{E}_{\omega[n]} \left[l(n, \mathbf{u}, \mathbf{x}[n]) + J_{n+1}^*(f_d(\mathbf{x}[n], \mathbf{u}, n, \omega[n])) \right]$$

$$n = N-1, \ldots, 0$$

DP – Inventory Control Example

- Stochastic DP
- Stock available $x[n] \in \boldsymbol{N}$, order $u[n] \in \boldsymbol{N}$, demand $w[n] \in \boldsymbol{N}$
- Dynamics: $x[n+1] = \max(0, x[n] + u[n] - w[n])$
- Constraints: $x[n] + u[n] \leq 2$
- Simple stationary demand model: $p(w[n] = 0) = 0.1, p(w[n] = 1) = 0.7, p(w[n] = 2) = 0.2$
- Objective:

$$\mathbb{E}\left[\underbrace{0}_{\text{No terminal cost}} + \sum_{n=0}^{2}\left(\underbrace{u[n]}_{\text{Cost to purchase}} + \underbrace{(x[n] + u[n] - w[n])^2}_{\text{Lost business/over-supply cost}}\right)\right]$$

DP – Inventory Control Example

DP Algorithm:

$$J_n^*(x[n]) = \min_{0 \le u[n] \le 2 - x[n]} \mathbb{E}_{w[n]} \Big[u[n] + (x[n] + u[n] - w[n])^2 + \\ + J_{n+1}^*(\max(0, x[n] + u[n] - w[n])) \Big]$$

As an example: $J_2^*(0) = \min_{u \in \{0,1,2\}} \mathbb{E}_{w[2]} \left[u + (u[2] - w[2])^2 \right]$

$$= \min_{u \in \{0,1,2\}} \left[u + 0.1u^2 + 0.7(u-1)^2 + 0.2(u-2)^2 \right]$$

Thus: $J_2^*(0) = 1.3, \pi_2^*(0) = 1$. Show: $J_0^*(0) = 3.7, J_0^*(1) = 2.7, J_0^*(2) = 2.818$

DP in Discrete Spaces

Notice: $$J_n^*(\mathbf{x}[n]) = \min_{\mathbf{u}}[l(n, \mathbf{u}, \mathbf{x}[n]) + \underbrace{J_{n+1}^*(f_d(\mathbf{x}[n], \mathbf{u}, n))}_{\text{Need to solve for all "successor" states first.}}]$$

Recursion needs solution for **all** possible next states.

- Doable for **finite/discrete** state-spaces (e.g., grids).
- Suffers from curse of dimensionality (e.g., consider quantizing a continuous state-space)

Value Iteration:

- Set up a recursion: $J_n(\boldsymbol{x}) \leftarrow \min_u (l(n, \boldsymbol{u}, \boldsymbol{x}) + J_{n+1}(f_d(\boldsymbol{x}, \boldsymbol{u}, n)))$ for all $\boldsymbol{x}$.
- Infinite horizon setting → drop the time dependence, and iterate until convergence.

Generalized Policy Iteration:

- Interleave policy evaluation (similar recursion with $\min$ replaced with policy), and policy improvement (argmin of Bellman formula with current value estimate)

DP in Continuous Spaces

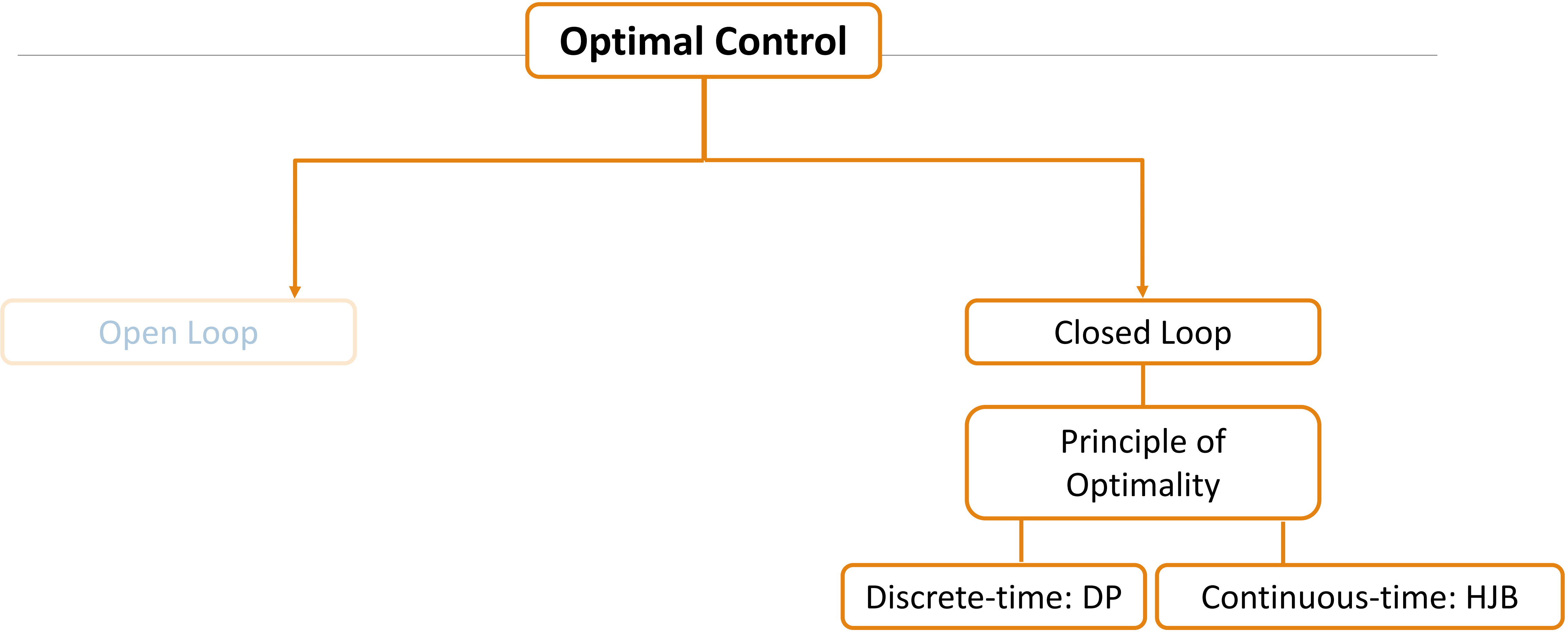
Rarely, we have exact solution in continuous spaces. Otherwise: need function approximation: *Approximate Dynamic Programming*

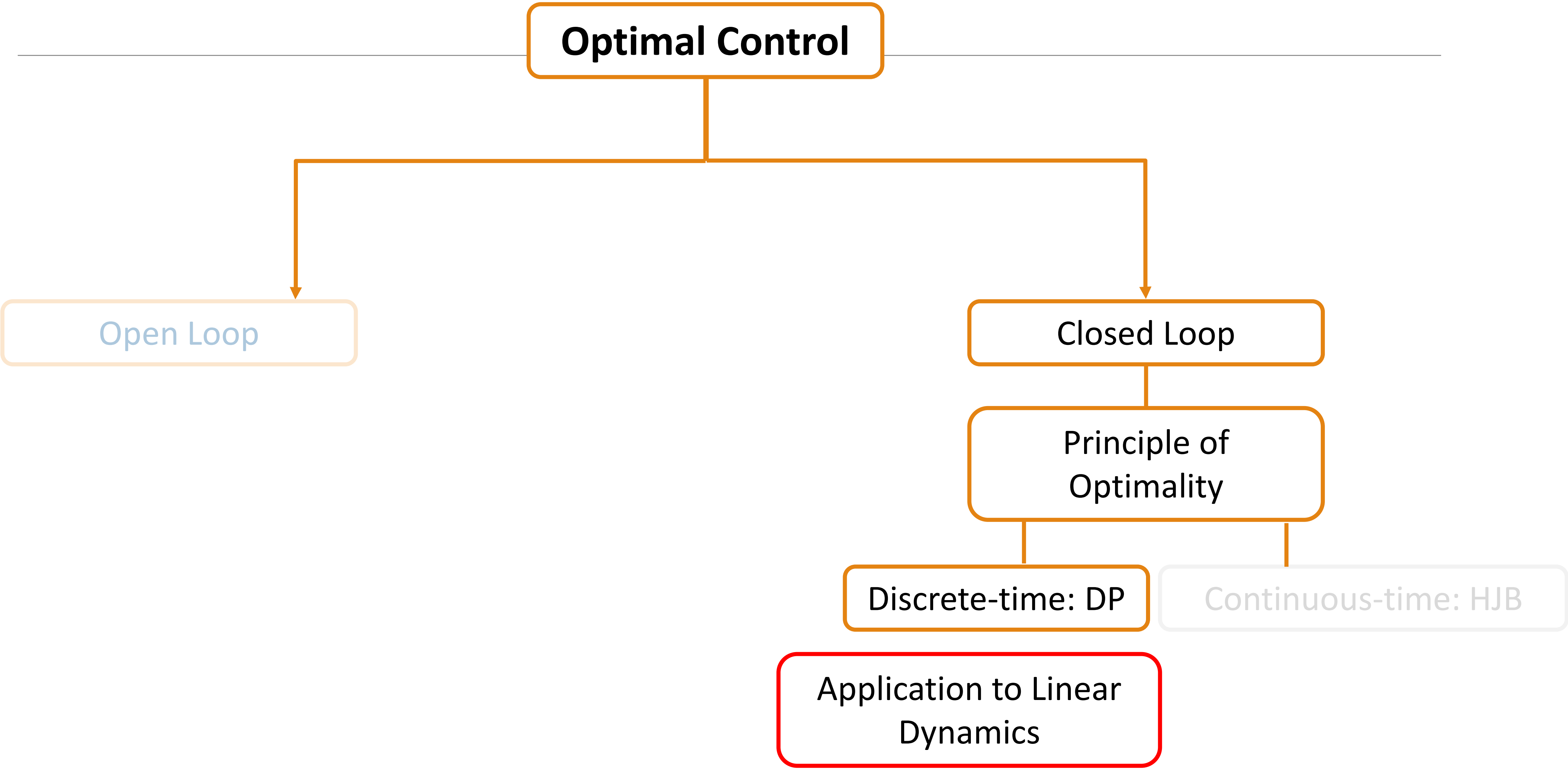
Examples:

- *Fitted Value Iteration*: bootstrap off current/delayed estimate of value function to compute “targets” and regress.
- *Meshes*: perform iteration on a discrete mesh and use interpolation

Dynamics unknown: Reinforcement Learning: find optimal policy and value function using samples of experience $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}', c)$.

- Algorithms resemble stochastic approximations of recursion formulas (+tricks)


Optimal Control
Open Loop
Closed Loop
Principle of Optimality
Discrete-time: DP
Continuous-time: HJB



Optimal Control
Open Loop
Closed Loop
Principle of Optimality
Discrete-time: DP
Continuous-time: HJB
Application to Linear Dynamics

DP For LQ Control

Linear time-varying dynamics: $\boldsymbol{x}[n+1] = A_n \boldsymbol{x}[n] + B_n \boldsymbol{u}[n]$

Quadratic time-varying cost:

$$J(\boldsymbol{u}, \boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}[N]^T Q_N \boldsymbol{x}[N] + \frac{1}{2}\sum_{0}^{N-1}\left(\boldsymbol{x}[n]^T Q_n \boldsymbol{x}[n] + \boldsymbol{u}[n]^T R_n \boldsymbol{u}[n] + 2\boldsymbol{x}[n]^T S_n \boldsymbol{u}[n]\right)$$

$Q_n \succcurlyeq 0, R_n \succ 0$

Can treat as one big (convex) QP:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^T W \mathbf{z} \\ \text{s.t.} \quad & C\mathbf{z} + \mathbf{d} = \mathbf{0} \end{aligned}$$

Instead, let’s apply DP.

DP for LQ Control

Initialize Bellman recursion: $J_N^*(\boldsymbol{x}[N]) = \frac{1}{2}\boldsymbol{x}[N]^T Q_N \boldsymbol{x}[N] := \frac{1}{2}\boldsymbol{x}[N]^T V_N \boldsymbol{x}[N]$

Apply recursion:

$$J_{N-1}^*(\boldsymbol{x}[N-1]) = \frac{1}{2}\min_{\boldsymbol{u}} \underbrace{\left[\begin{bmatrix} \boldsymbol{x}[N-1] \\ \boldsymbol{u} \end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1} \\ S_{N-1}^T & R_{N-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}[N-1] \\ \boldsymbol{u} \end{bmatrix} + \|\boldsymbol{x}[N]\|_{V_N}^2 \right]}_{:=Q_{N-1}(\boldsymbol{x}[N-1],\boldsymbol{u})}$$

Plug in for dynamics, optimize w.r.t. $\boldsymbol{u}$ (set gradients to zero*) and solve:

$$\mathbf{u}^*[N-1] = L_{N-1}\mathbf{x}[N-1]$$

*Confirm: $\nabla_{\boldsymbol{u}}^2 Q_{N-1}(\boldsymbol{x},\boldsymbol{u}) = R_{N-1} + B_{N-1}^T V_N B_{N-1} \succ 0$

Plug optimal control law back into J_{N-1}^* to get

$$J_{N-1}^* = \boldsymbol{x}[N-1]^T V_{N-1} \boldsymbol{x}[N-1]$$

Optimal cost-to-go is quadratic
Optimal policy is time-varying linear

DP for LQ Control

Full backward recursion (Riccati difference recursion):

$$V_N = Q_N$$

$$L_n = -(R_n + B_n^T V_{n+1} B_n)^{-1} \left(B_n^T V_{n+1} A_n + S_n^T \right)$$

$$V_n = Q_n + A_n^T V_{n+1} A_n - (A_n^T V_{n+1} B_n + S_n)(R_n + B_n^T V_{n+1} B_n)^{-1}(B_n^T V_{n+1} A_n + S_n^T)$$

$$\pi_n^*(\boldsymbol{x}) = L_n \boldsymbol{x}$$

$$J_n^*(\boldsymbol{x}[n]) = \frac{1}{2}\boldsymbol{x}[n]^T V_n \boldsymbol{x}[n]$$

For $N = \infty$, $(A_n, B_n, Q_n, R_n, S_n) = (A, B, Q, R, S)$ with (A, B) controllable, $V_n, L_n \rightarrow$ constant matrices (thereby obtaining infinite horizon/stationary policy)

DP for LQ Control

If cost has linear terms $q_n^T \boldsymbol{x}[n] + r_n^T \boldsymbol{u}[n]$ and/or the dynamics has a drift term:

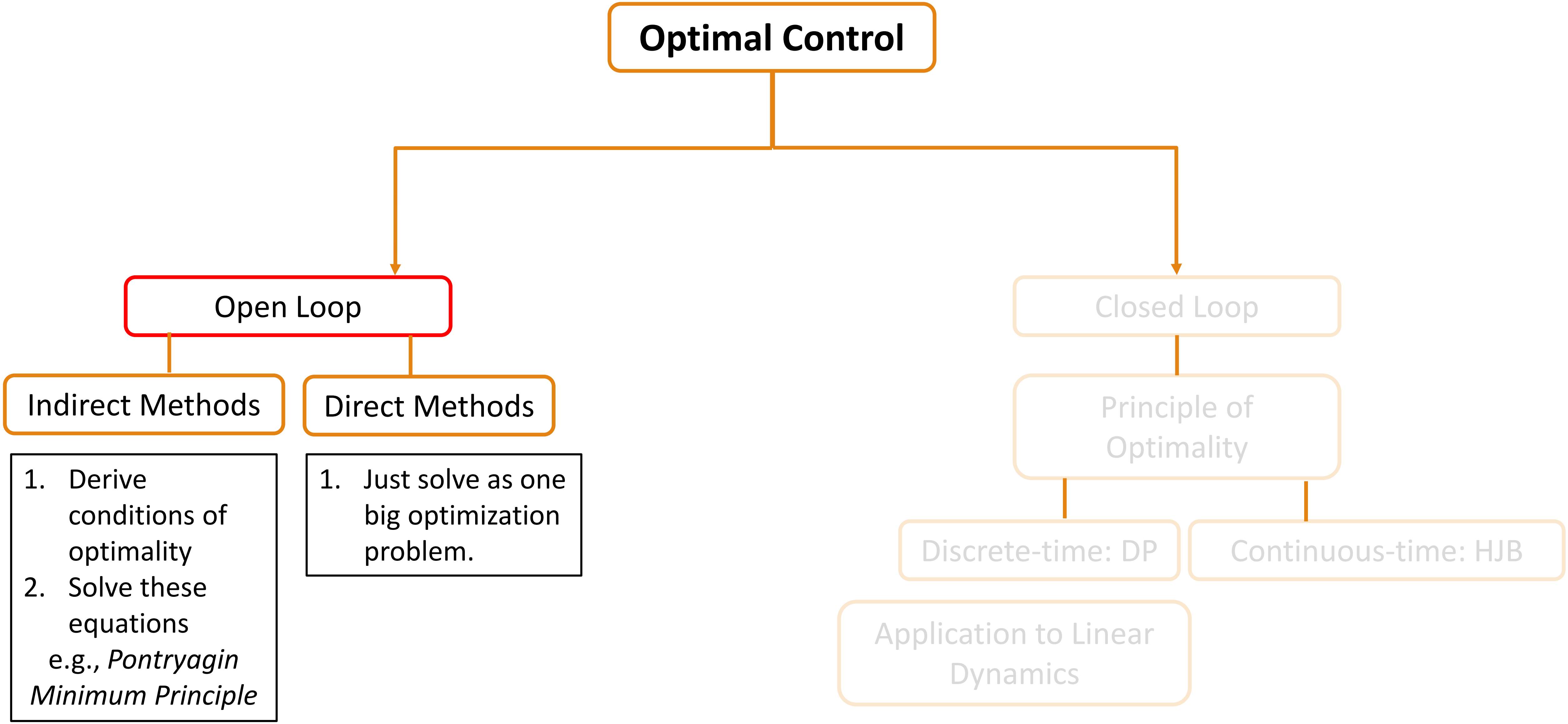
$$\boldsymbol{x}[n+1] = A_n \boldsymbol{x}[n] + B_n \boldsymbol{u}[n] + c_n$$

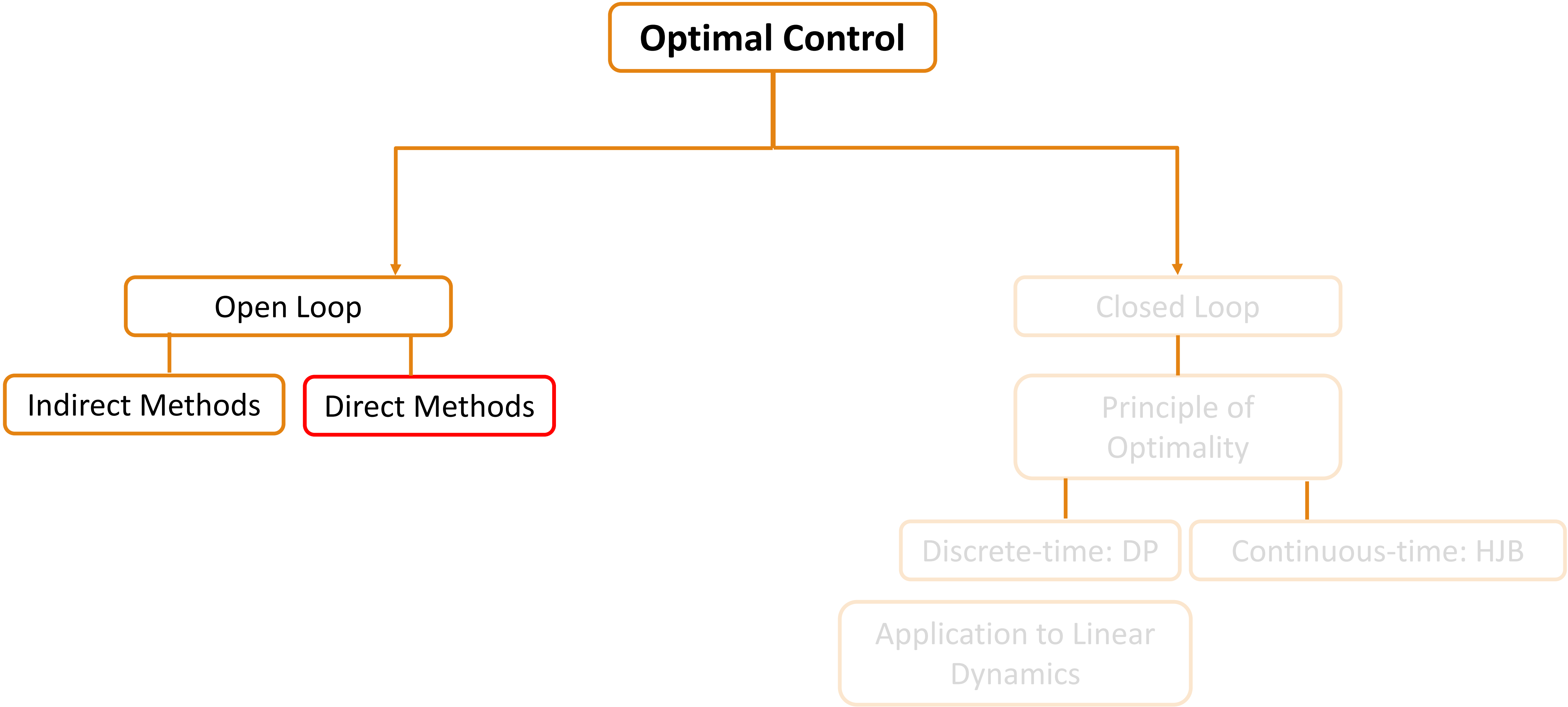
Then, re-write using the composite state $\boldsymbol{y} = (\boldsymbol{x}, 1)^T$:

$$\boldsymbol{y}[n+1] = \begin{bmatrix} \boldsymbol{x}[n+1] \\ 1 \end{bmatrix} = \begin{bmatrix} A_n & c_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}[n] \\ 1 \end{bmatrix} + \begin{bmatrix} B_n \\ 0 \end{bmatrix} \boldsymbol{u}[n] := \tilde{A}_n \boldsymbol{y}[n] + \tilde{B}_n \boldsymbol{u}[n]$$

Implications:

- Optimal cost-to-go is a general quadratic: $J_n^*(\boldsymbol{x}[n]) = \frac{1}{2} \boldsymbol{x}[n]^T V_n \boldsymbol{x}[n] + v_n^T \boldsymbol{x}[n] + p_n$
- Optimal policy is time-varying affine: $\pi_n^*(\boldsymbol{x}) = L_n \boldsymbol{x} + k_n$


Optimal Control
Open Loop
Closed Loop
Indirect Methods
Direct Methods
1. Derive conditions of optimality
2. Solve these equations
e.g., Pontryagin Minimum Principle
1. Just solve as one big optimization problem.
Principle of Optimality
Discrete-time: DP
Continuous-time: HJB
Application to Linear Dynamics


Optimal Control
Open Loop
Closed Loop
Indirect Methods
Direct Methods
Principle of Optimality
Discrete-time: DP
Continuous-time: HJB
Application to Linear Dynamics

Open-Loop Optimal Control

Discrete Time:

- Performance measure (minimize): $J(\mathbf{u}, \mathbf{x}) = \sum_{n=0}^{N-1} l(n, \mathbf{u}[n], \mathbf{x}[n]) + \phi(\mathbf{x}[N])$
- Dynamics: $\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}[n], n)$

Open-loop: Given $\boldsymbol{x}[0] = \boldsymbol{x}_{\mathbf{0}}$, find optimal sequences: $(\boldsymbol{x}^*[\,], \boldsymbol{u}^*[\,])$.

- If objective convex and dynamics linear → convex problem.

Gradient Descent

- More generally, define the stage-wise **Hamiltonian**:

$$H_n(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = l(n, \boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T f_d(\boldsymbol{x}, \boldsymbol{u}, n)$$

- Then, with $J_R(\boldsymbol{u}) \coloneqq J(\boldsymbol{u}, \boldsymbol{x}[\boldsymbol{u}])$, we have:

$$\langle \nabla_{\boldsymbol{u}} J_R, \boldsymbol{\delta u} \rangle = \sum_{n=0}^{N-1} \nabla_{\boldsymbol{u}[n]} H_n(\boldsymbol{x}[n], \boldsymbol{u}[n], \boldsymbol{\lambda}[n+1])^T \boldsymbol{\delta u}[n]$$

- Where $\boldsymbol{\lambda}$ (co-state/adjoint) satisfies a backward recursion:

$$\boldsymbol{\lambda}[n] = \frac{\partial l(n, \boldsymbol{x}[n], \boldsymbol{u}[n])}{\partial \boldsymbol{x}} + \left(\frac{\partial f_d(\boldsymbol{x}[n], \boldsymbol{u}[n], n)}{\partial \boldsymbol{x}} \right)^T \boldsymbol{\lambda}[n+1] \qquad \boldsymbol{\lambda}[N] = \frac{\partial \phi(\boldsymbol{x}[N])}{\partial \boldsymbol{x}}$$

Newton Descent

Moreover, we also have:

Newton Direction ↔ Solve an LQ problem

For completeness:

$$\langle \nabla_{\boldsymbol{u}} J_R, \boldsymbol{\delta u} \rangle + \frac{1}{2}\langle \boldsymbol{\delta u}, \nabla^2_{\boldsymbol{u}} J_R \boldsymbol{\delta u} \rangle = \frac{1}{2}\boldsymbol{\delta x}[N]^T \nabla^2_{xx}\phi(\boldsymbol{x}[N])\boldsymbol{\delta x}[N] + $$

$$+ \sum_{n=0}^{N-1} \left(q_n^T \boldsymbol{\delta x}[n] + r_n^T \boldsymbol{\delta u}[n] + \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta x}[n] \\ \boldsymbol{\delta u}[n] \end{bmatrix}^T \begin{bmatrix} \nabla^2_{xx} H_n & \nabla^2_{xu} H_n \\ \nabla^2_{ux} H_n & \nabla^2_{uu} H_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta x}[n] \\ \boldsymbol{\delta u}[n] \end{bmatrix} \right)$$

Where

$$\boldsymbol{\delta x}[n+1] = A_n \boldsymbol{\delta x}[n] + B_n \boldsymbol{\delta u}[n], \quad \boldsymbol{\delta x}[0] = 0$$

$$(A_n, B_n) = (\nabla_x f_d(n, \boldsymbol{x}[n], \boldsymbol{u}[n]), \nabla_u f_d(n, \boldsymbol{x}[n], \boldsymbol{u}[n]))$$

Dunn, 1989

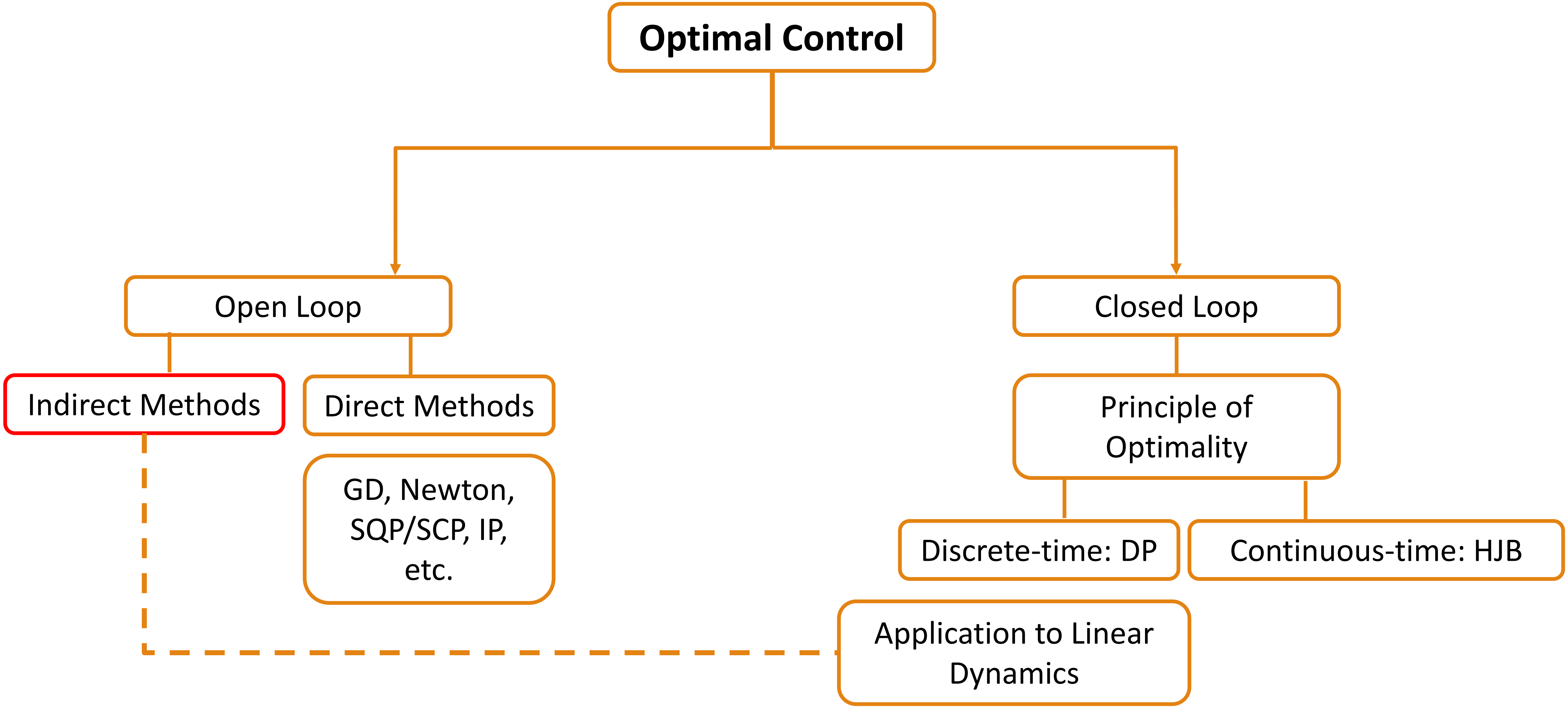
Newton Descent

Moreover, we also have:

Newton Direction ↔ Solve an LQ problem

Can we do better?

Dunn, 1989



Optimal Control
Open Loop
Closed Loop
Indirect Methods
Direct Methods
GD, Newton, SQP/SCP, IP, etc.
Principle of Optimality
Discrete-time: DP
Continuous-time: HJB
Application to Linear Dynamics

Differential Dynamic Programming (DDP)

Consider the DP recursion:

$$J_n(\boldsymbol{x}[n]) = \min_{\boldsymbol{u}}[l(\boldsymbol{x}[n], \boldsymbol{u}) + J_{n+1}(f_d(\boldsymbol{x}[n], \boldsymbol{u}))]$$

Fix sequence of controls $\boldsymbol{u}$, with corresponding state sequence $\boldsymbol{x}$, and for $n' = N - 1$, consider:

$$J_{n'}(\boldsymbol{x}[n'] + \boldsymbol{\delta x}) = \min_{\boldsymbol{\delta u}}[\underbrace{l(\boldsymbol{x}[n'] + \boldsymbol{\delta x}, \boldsymbol{u}[n'] + \boldsymbol{\delta u}) + \phi(f_d(\boldsymbol{x}[n'] + \boldsymbol{\delta x}, \boldsymbol{u}[n'] + \boldsymbol{\delta u}))}_{:= Q_{n'}(\boldsymbol{\delta x}, \boldsymbol{\delta u})}]$$

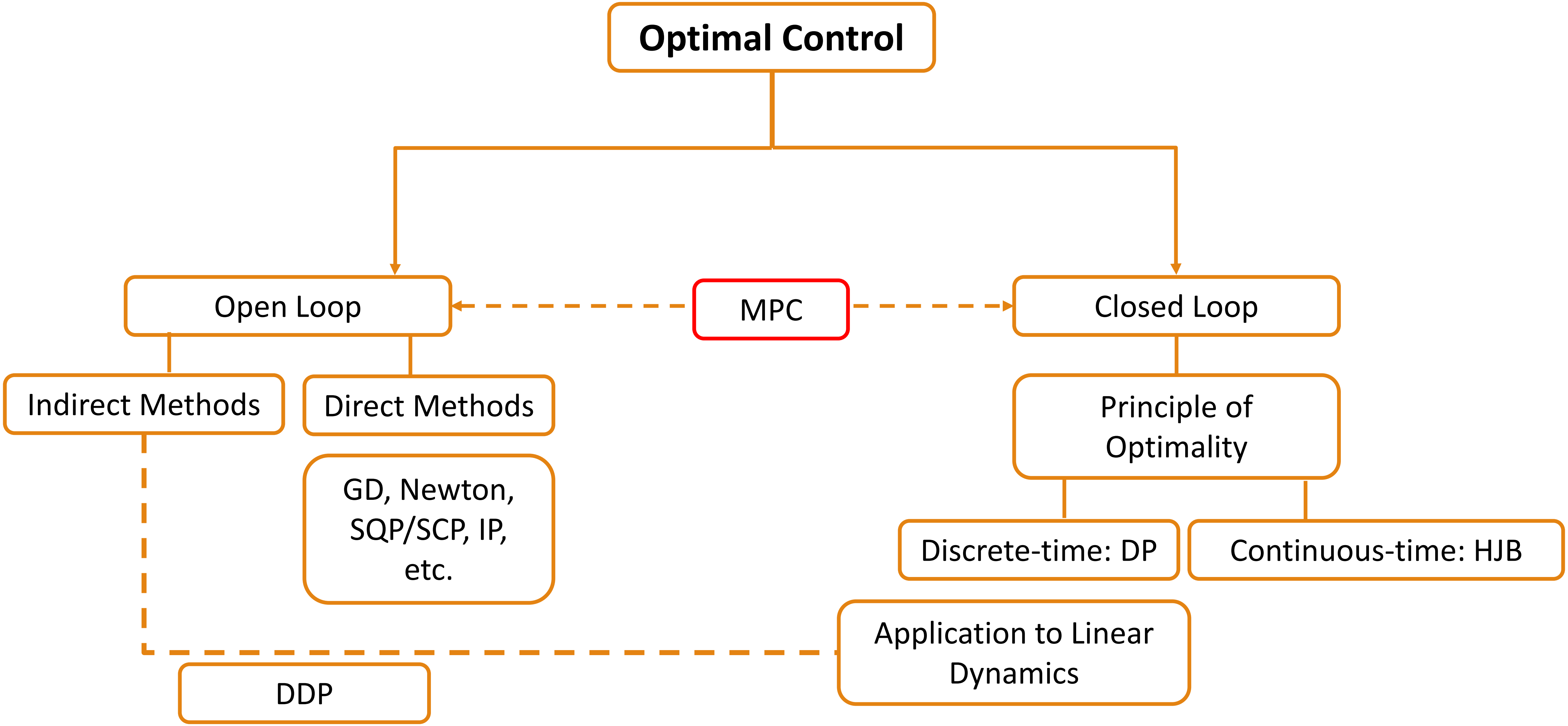
Taylor expand $Q_{n'}$ about $(\boldsymbol{x}[n'], \boldsymbol{u}[n'])$ to 2nd order and minimize w.r.t. $\boldsymbol{\delta u}$

- Yields affine control law: $\boldsymbol{\delta u}^*[n'] = L_n \boldsymbol{\delta x}[n'] + \boldsymbol{\delta u}_{n'}$ "feedforward correction"
- Substitute back into $Q_{n'}$, yielding quadratic approximation for $\widehat{J_{n'}}$ about $\boldsymbol{x}[n']$
- Continue recursion going backwards, with
$$Q_n(\boldsymbol{\delta x}, \boldsymbol{\delta u}) = l(\boldsymbol{x}[n] + \boldsymbol{\delta x}, \boldsymbol{u}[n] + \boldsymbol{\delta u}) + \widehat{J_{n+1}}(f_d(\boldsymbol{x}[n] + \boldsymbol{\delta x}, \boldsymbol{u}[n] + \boldsymbol{\delta u}))$$
- *Almost* the Riccati backward recursion for the Newton LQ problem

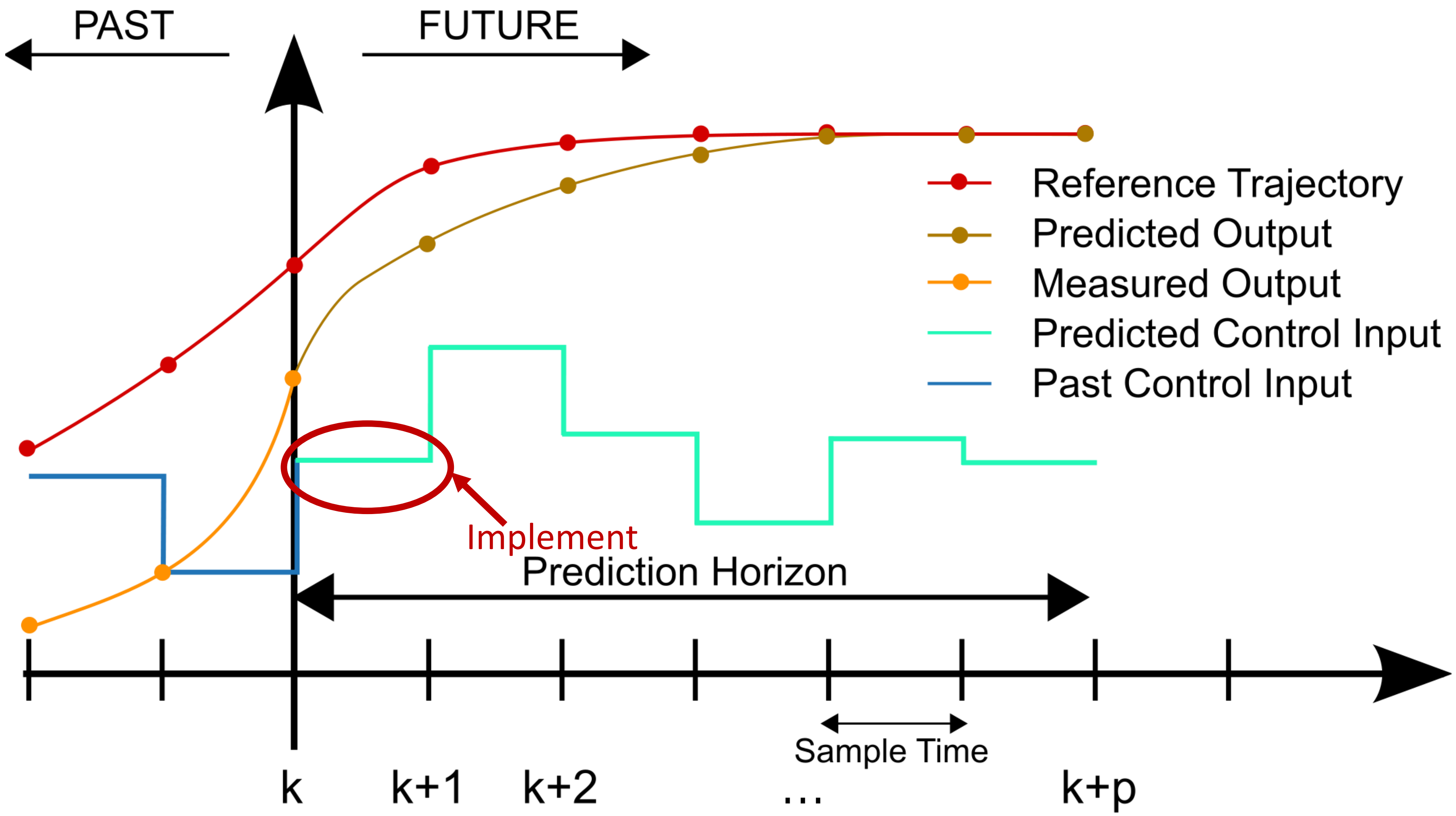
Forward pass $\boldsymbol{u} + \boldsymbol{\delta u}$ through f_d using affine control law: DDP Algorithm: "Quasi-Newton" Descent

- Newton would forward pass through linearized dynamics

Near optimal, behaves like Newton. Far from optimal, much more efficient.


Optimal Control
Open Loop
MPC
Closed Loop
Indirect Methods
Direct Methods
Principle of Optimality
GD, Newton, SQP/SCP, IP, etc.
Discrete-time: DP
Continuous-time: HJB
Application to Linear Dynamics
DDP

Model Predictive Control (MPC)



Tradeoffs on Open vs Closed

Open-Loop Control

- Rarely used blindly online due to model errors
- Expensive to compute online (need to use MPC and/or simplifications)
- Can compute offline for a set of initial conditions (e.g., a trajectory library) and adapt online via fine-tuning (e.g., Boston Dynamics Atlas Robot)

Closed-Loop Control

- Needed if there is **any** source of unmodelled effects (dynamics, disturbances, other agents, sensor noise), i.e., **always**
- Difficult to compute **true** optimal in discrete-time (functional recursion) or continuous-time (PDE)
  - Difficult to **certify** optimality
  - Instead, we look for certifying **correctness** and **safety** (e.g., via Lyapunov)
- MPC: bridge of open/closed-loop via **online re-planning**
- Feedback tracking: bridge of open/closed-loop: track open-loop **plans** with **feedback controllers**

Other Topics

Hybrid systems (e.g., locomotion)

Adaptive Control

Reinforcement Learning

Stochastic dynamics

Approximate Dynamic Programming

Partial observability

High-dimensional observations (vision)

Feedback controller design

Task & Motion Planning

Some References

The (updated) classic: Optimal Control & Dynamic Programming:

- Bertsekas Volumes 1 & 2

Introductory text – a must have:

- Kirk

Applied Optimal control – more advanced, generally assumes knowledge of the basics:

- Bryson and Ho

Model Predictive control – from a more modern perspective:

- Kouvaritakis & Cannon

Applied Nonlinear control – a comprehensive intro to nonlinear control:

- Slotine & Li