Lecture 12

Optimal Control and Reinforcement Learning

Libin Liu

School of Intelligence Science and Technology
Peking University

GAMES105 课程交流

VCL @ PKU

Outline

- Optimal Control
- Model-based Approaches vs. Model-free Approaches
- Sampling-based Optimization
- Reinforcement Learning

- Conclusion

Recap: Trajectory Optimization

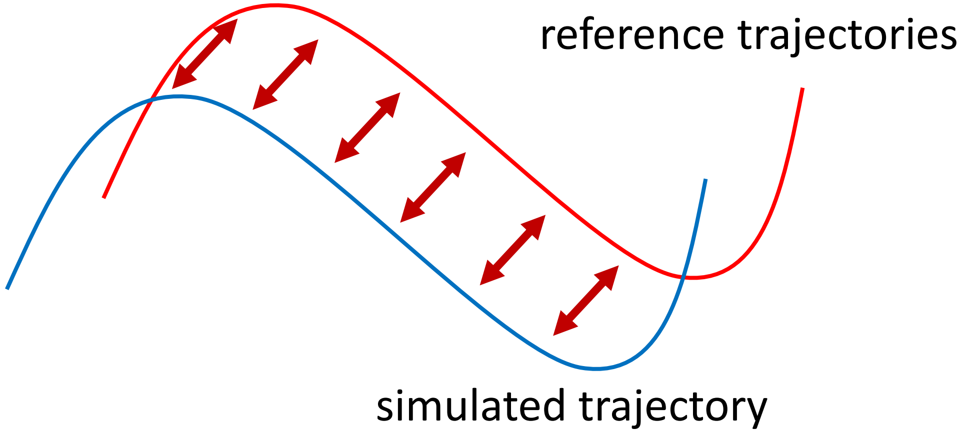
Find a control sequence $\{a_t\}$ that generates a state sequence $\{s_t\}$ start from s_0 minimizes

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

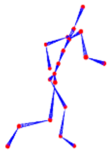
subject to

$$f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$$

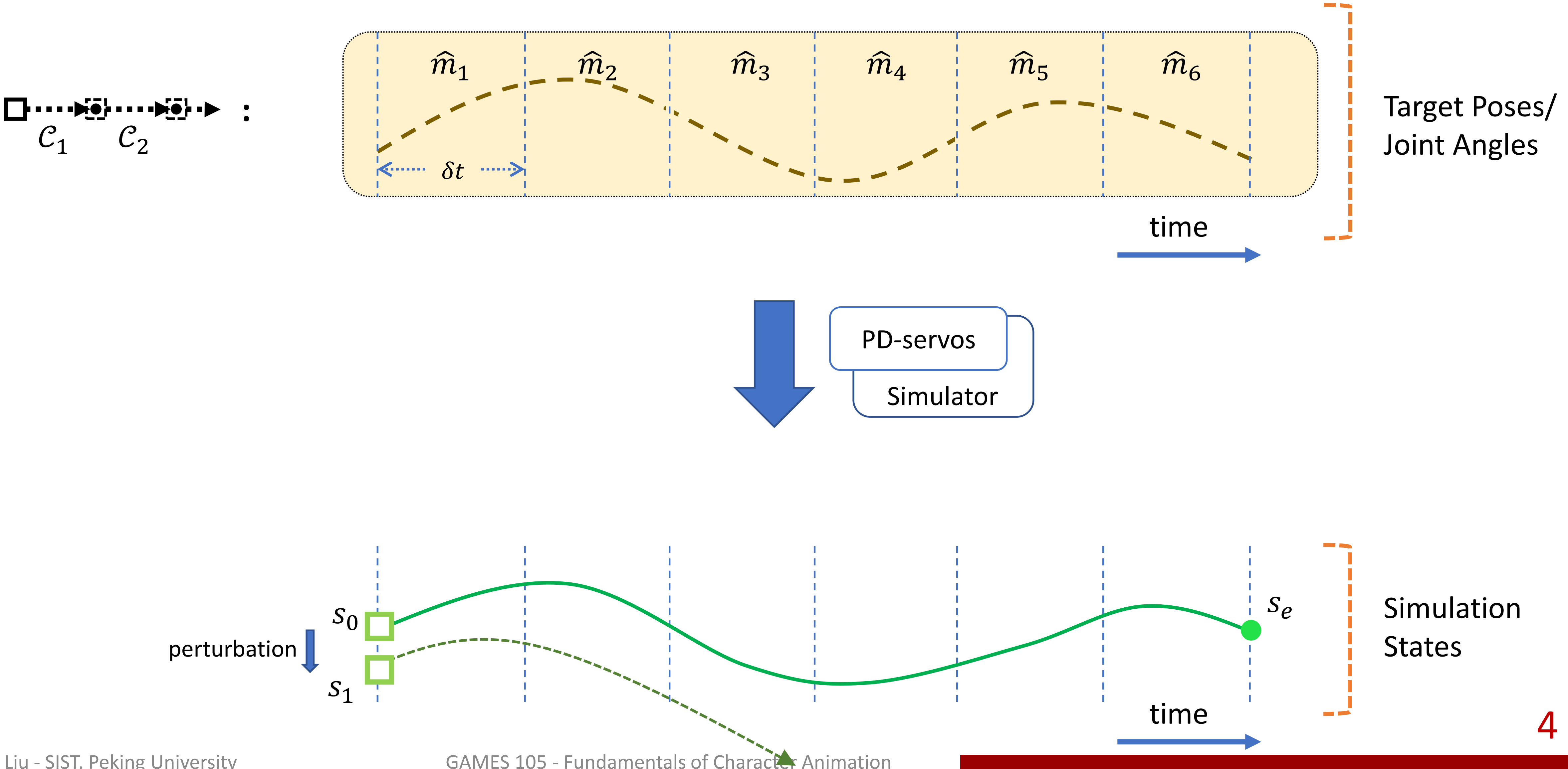
Equations of motion
$$M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda$$
$$g(\boldsymbol{x}, \boldsymbol{v}) \ge 0$$



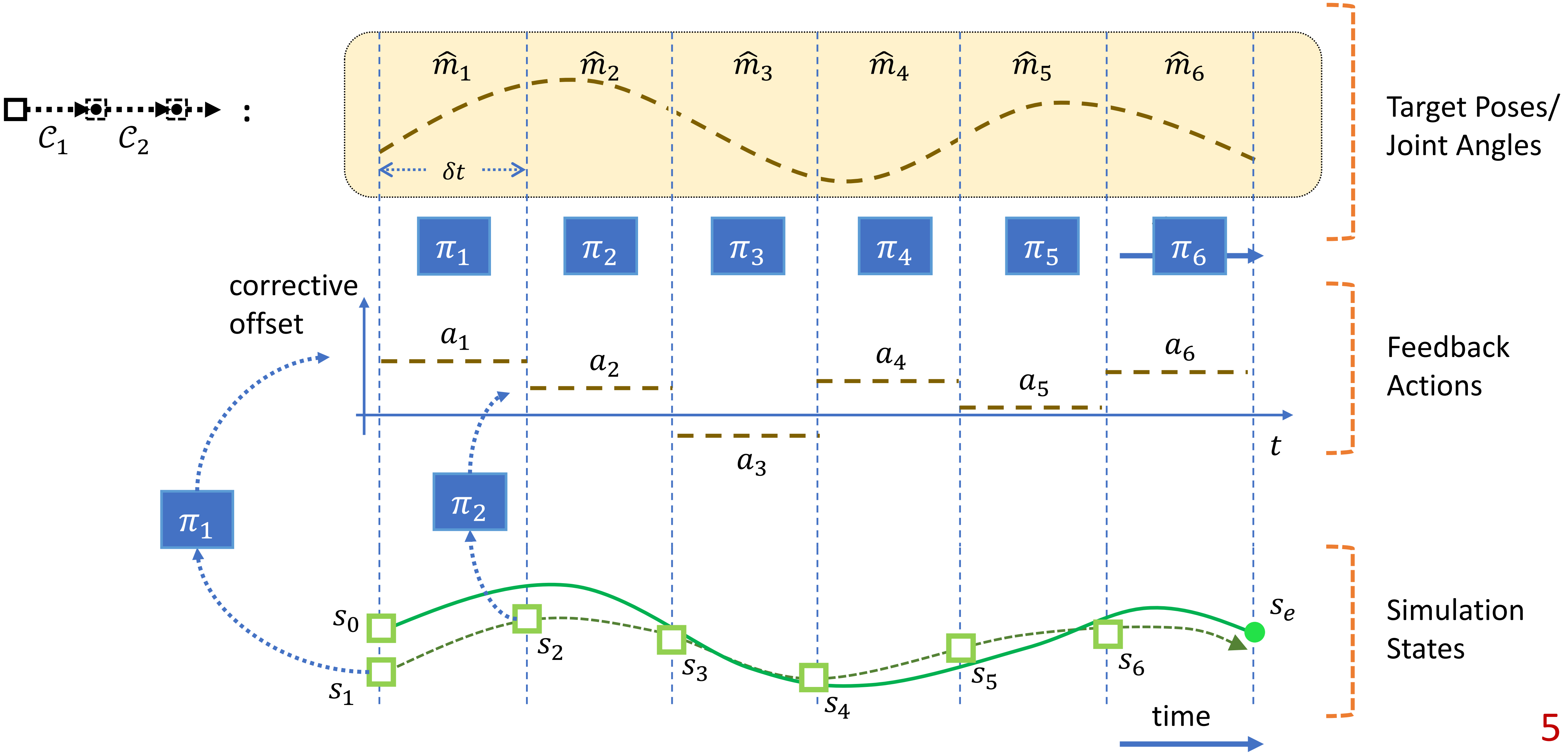
minimize accumulated tracking error:

$\sum_t \|$  $-$  $\| + \cdots$

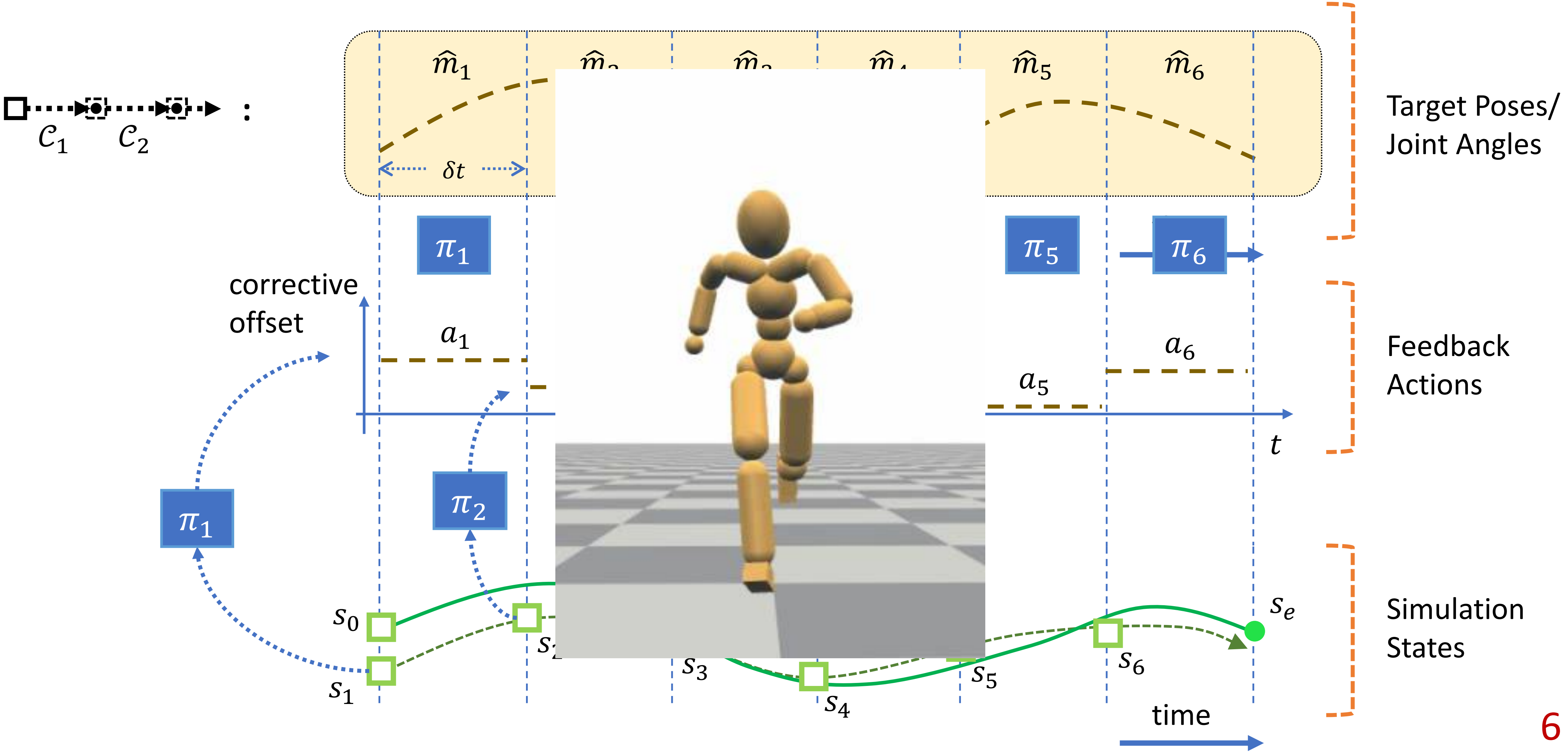
Recap: Feedforward Control



Recap: Feedback Control



Recap: Feedback Control



Recap: Feedback Control

Find a control policy $\pi(s_t, t)$ that computes actions a_t to generate a state sequence $\{s_t\}$ start from s_0 that minimizes

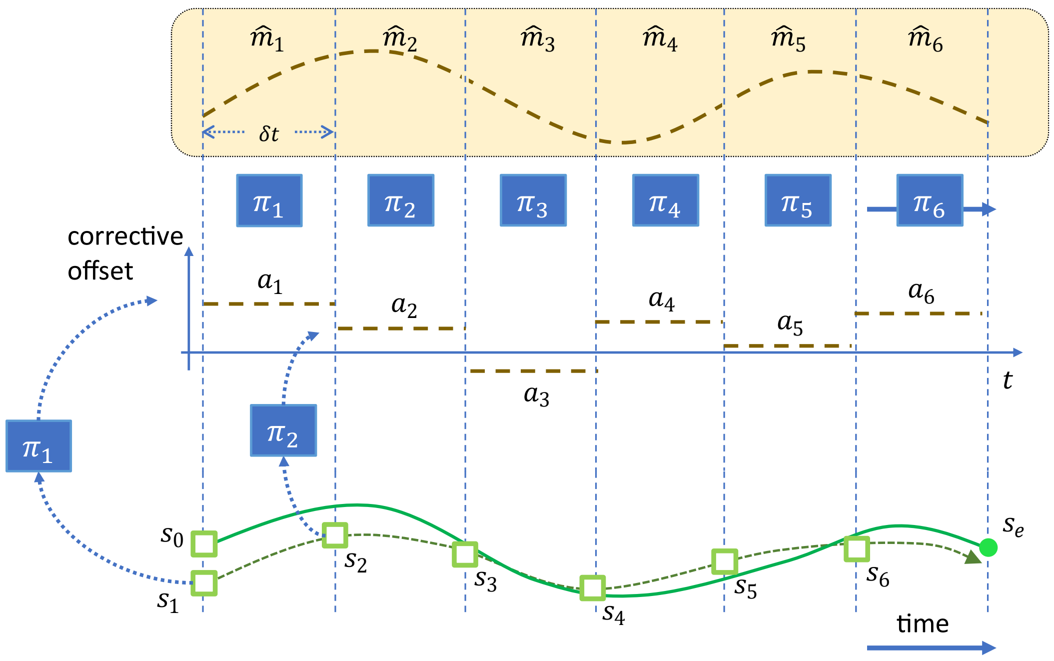
$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$$

Equations of motion $\quad M\dot{\boldsymbol{v}} + C(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f} + J^T\lambda$

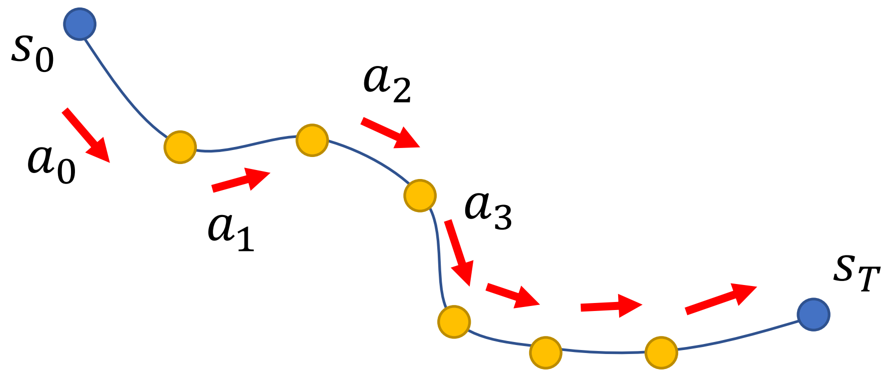
$\qquad g(\boldsymbol{x}, \boldsymbol{v}) \ge 0$



Optimal Control

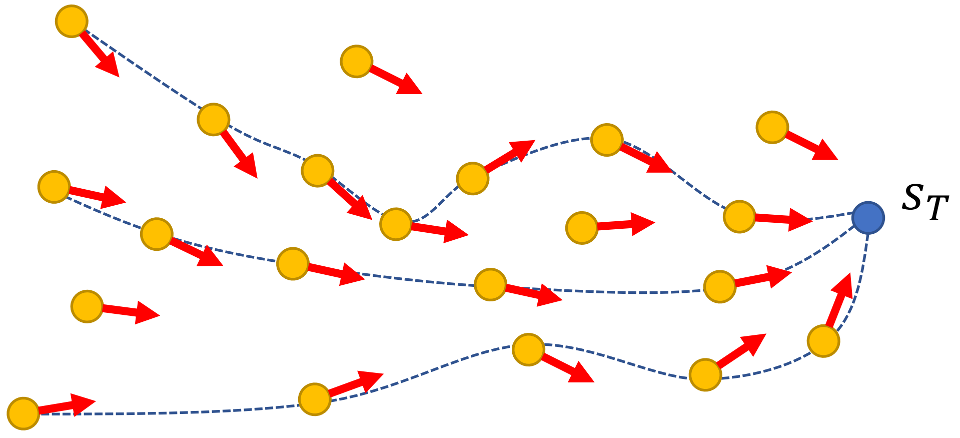
Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Feedback Control:

for any state s_t at time t, find the corresponding action $a_t = \pi(s_t, t)$ that eventually reach the goal



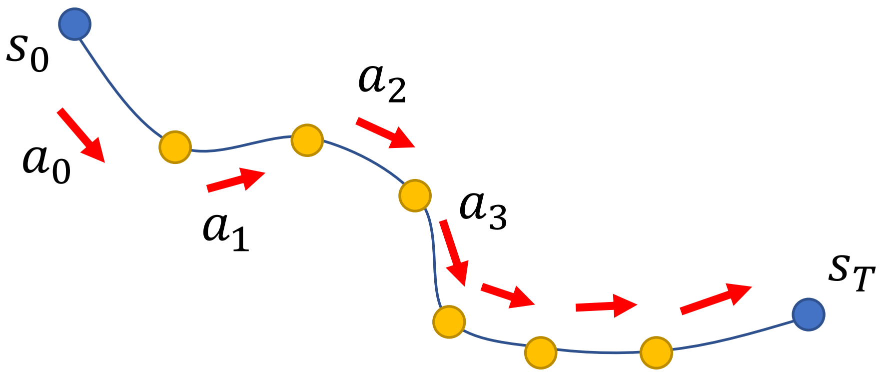
$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to $\quad f(s_t, a_t) - s_{t+1} = 0 \quad$ for $0 \le t < T$

Optimal Control

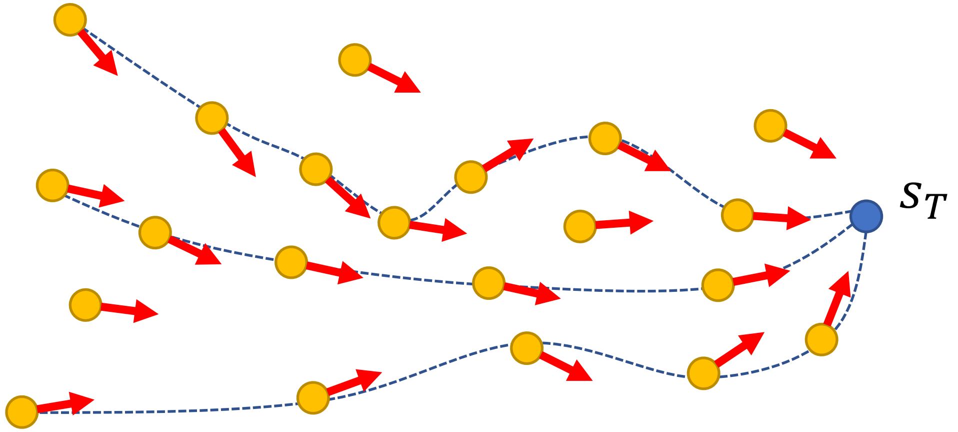
Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Feedback Control:

for any state s_t at time t, find the corresponding action $a_t = \pi(s_t, t)$ that eventually reach the goal



Trajectory Optimization

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

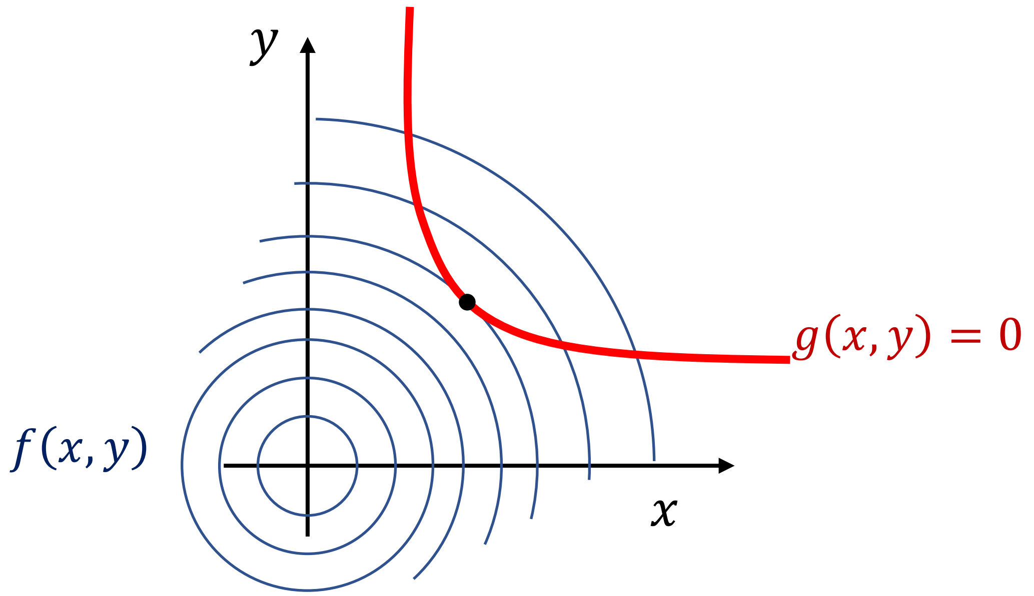
subject to $f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$

Dynamic Programming

Constrained Optimization

$$\min_x f(x)$$

$$s.t.\ \ g(x) = 0$$



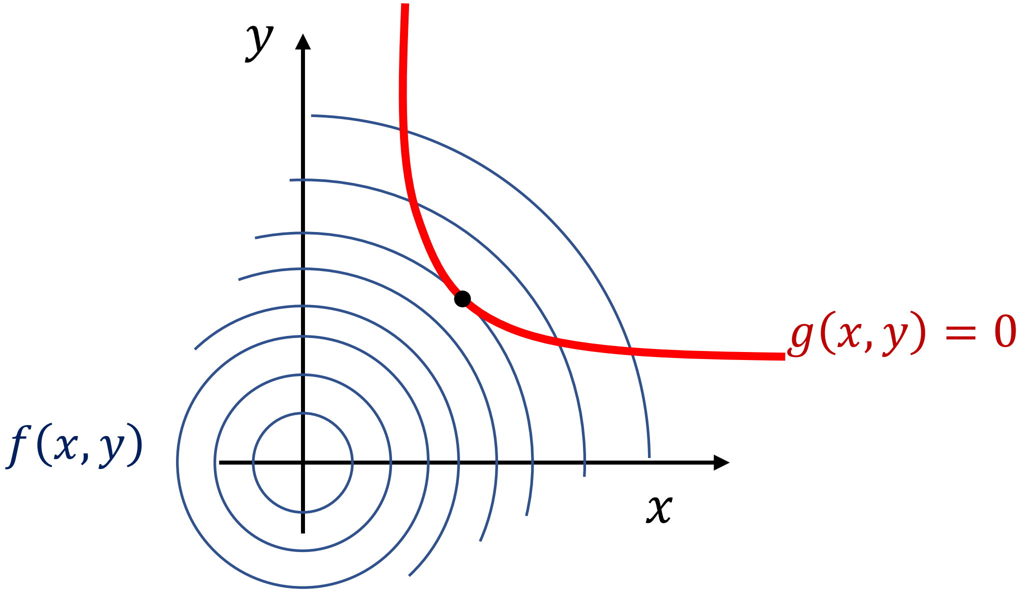
Constrained Optimization

$$\min_{x} f(x)$$

$$s.t. \quad g(x) = 0$$

Soft constraint?

$$\min_{x} f(x) + wg(x)$$



Constrained Optimization

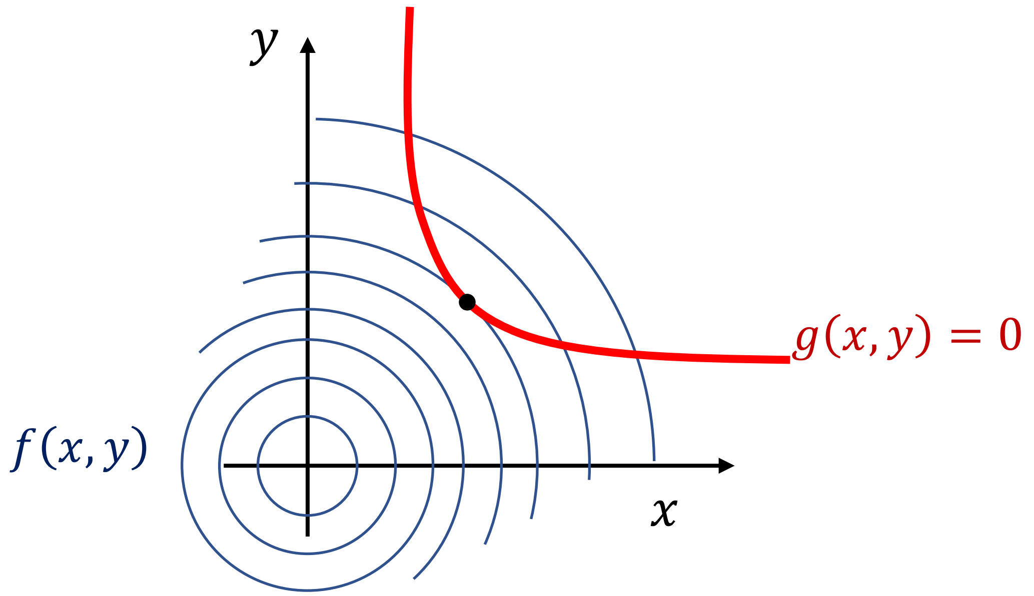
$$\min_x f(x)$$

$$s.t.\ \ g(x) = 0$$

Soft constraint?

$$\min_x f(x) + wg(x)$$

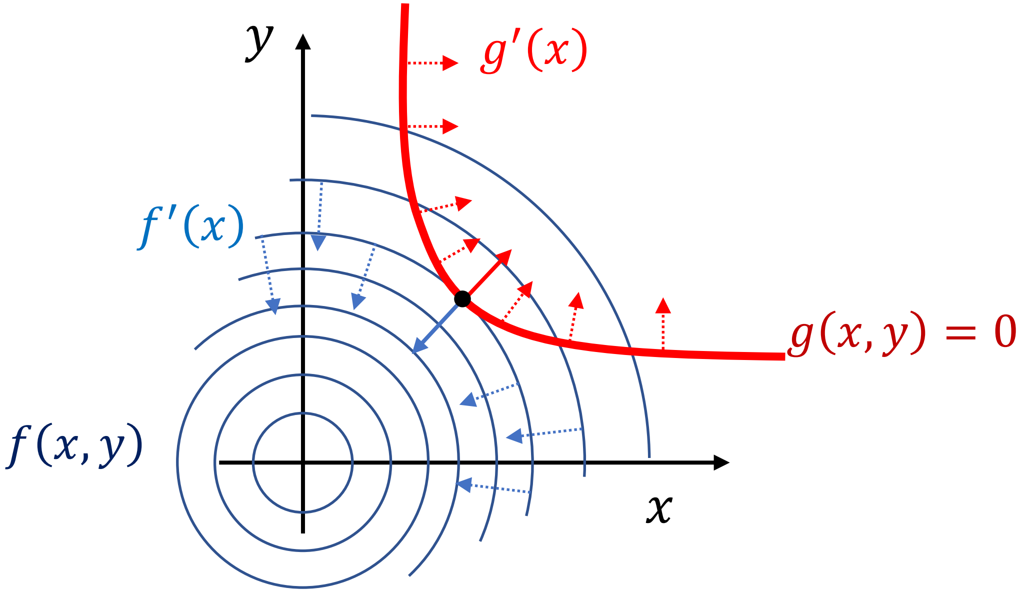
* The solution x^* may not satisfy the constraint



Constrained Optimization

$$\min_x f(x)$$

$$s.t.\ \ g(x) = 0$$



Constrained Optimization

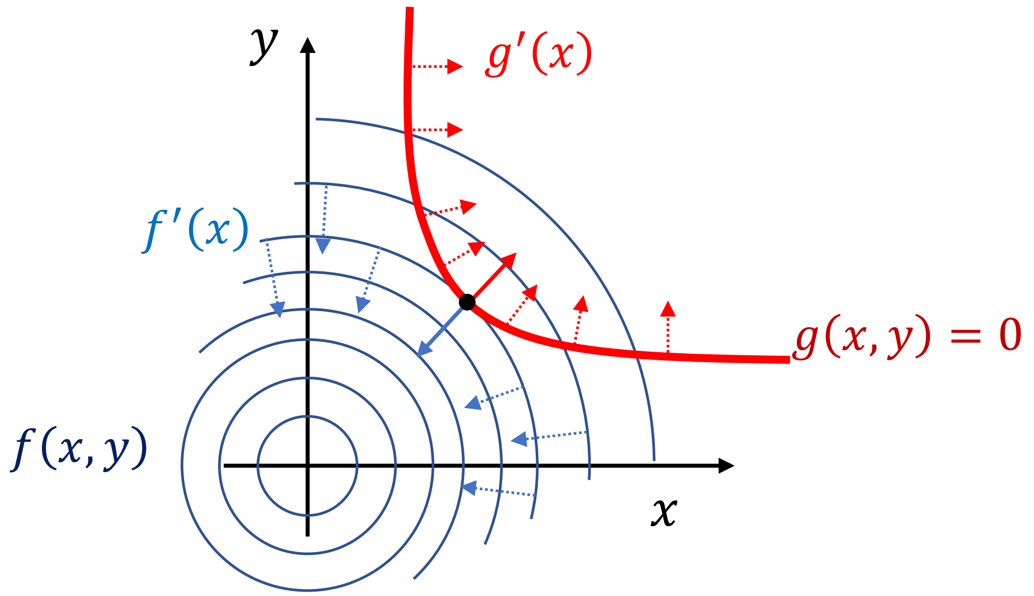
$$\min_{x} f(x)$$

$$s.t.\ \ g(x) = 0$$

x is optimal

$\Downarrow$

$f'(x)$ is parallel to $g'(x)$



Constrained Optimization

$$\min_x f(x)$$

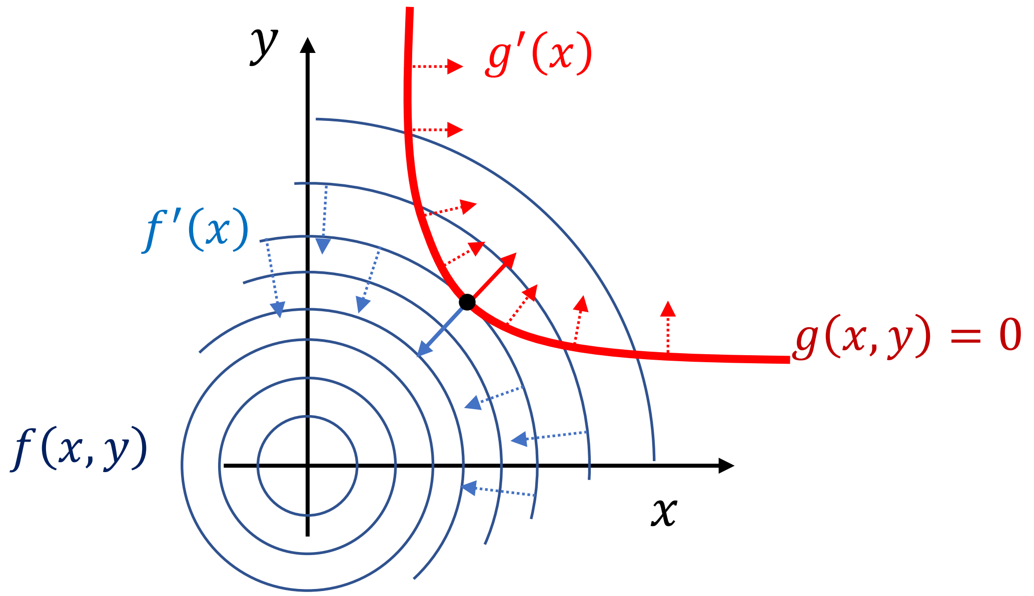
$$s.t.\ \ g(x) = 0$$

x is optimal

$\Downarrow$

$f'(x)$ is parallel to $g'(x)$

$$f'(x) + \lambda g'(x) = 0$$



Lagrange Multiplier

Lagrange function

$$L(x,\lambda) = f(x) + \lambda^T g(x)$$

$$\min_x f(x)$$

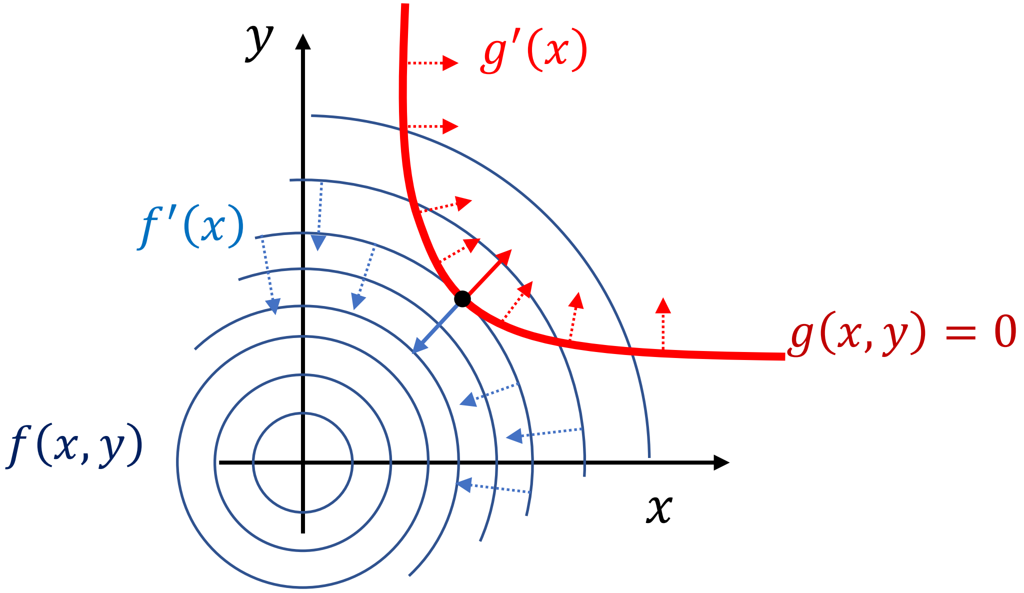
$$s.t. \quad g(x) = 0$$

x is optimal

$\Downarrow$

$f'(x)$ is parallel to $g'(x)$

$$f'(x) + \lambda g'(x) = 0$$



Lagrange Multiplier

$$\min_x f(x)$$

$$s.t.\ \ g(x) = 0$$

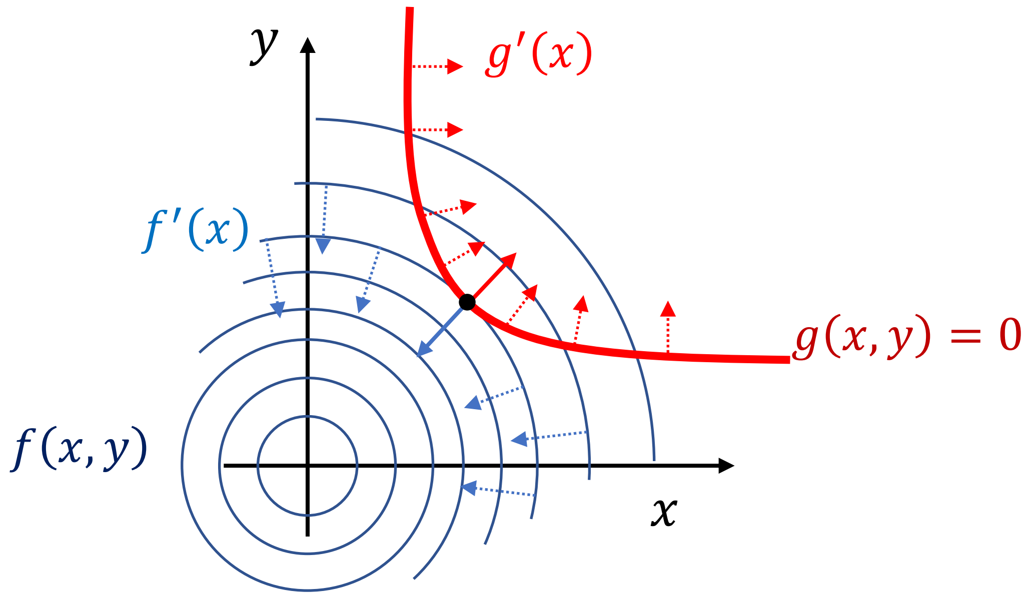
Lagrange function

$$L(x, \lambda) = f(x) + \lambda^T g(x)$$

We have the necessary condition for optimality:

$$\frac{\partial L}{\partial x} = f'^{(x)} + \lambda^T g'(x) = 0$$

$$\frac{\partial L}{\partial \lambda} = g(x) = 0$$



Lagrange Multiplier

$$\min_x f(x)$$

$$s.t. \;\; g(x) = 0$$

Lagrange function

$$L(x,\lambda) = f(x) + \lambda^T g(x)$$

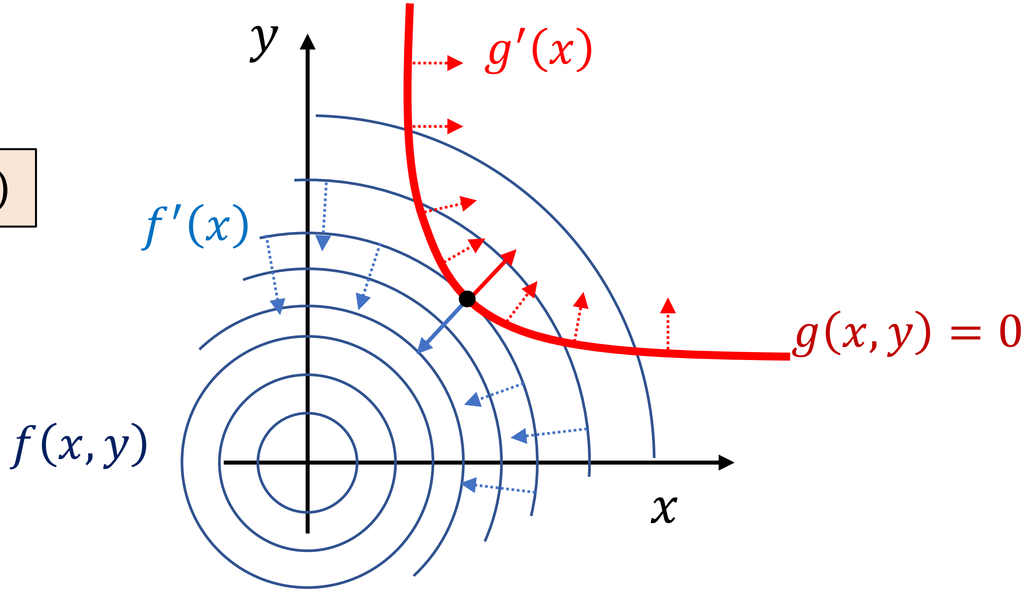
We have the necessary condition for optimality:

$f'(x)$ is parallel to $g'(x)$

$$\frac{\partial L}{\partial x} = f'^{(x)} + \lambda^T g'(x) = 0$$

$$\frac{\partial L}{\partial \lambda} = g(x) = 0$$

the original constraints



Solving Trajectory Optimization Problem

Find a control sequence $\{a_t\}$ that generates a state sequence $\{s_t\}$ start from s_0 minimizes

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0$$

for $0 \leq t < T$

Solving Trajectory Optimization Problem

Find a control sequence $\{a_t\}$ that generates a state sequence $\{s_t\}$ start from s_0 minimizes

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0$$

for $0 \leq t < T$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

Solving Trajectory Optimization Problem

Find a control sequence $\{a_t\}$ that generates a state sequence $\{s_t\}$ start from s_0 minimizes

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0$$

for $0 \leq t < T$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T \big(f(s_t, a_t) - s_{t+1}\big)$$

$$\frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$\frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

$$\frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$\frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

$$s_{t+1} = f(s_t, a_t)$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

$$\frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$\frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

$$\lambda_T = h_s'(s_T) \quad \Leftrightarrow \quad \frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$s_{t+1} = f(s_t, a_t) \quad \Leftrightarrow \quad \frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

$$\lambda_T = h_s'(s_T) \iff \frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1} \iff \frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$s_{t+1} = f(s_t, a_t) \iff \frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

λ is also called "costate"

$$\lambda_T = h_s'(s_T) \quad \Leftrightarrow \quad \frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1} \quad \Leftrightarrow \quad \frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$\frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$s_{t+1} = f(s_t, a_t) \quad \Leftrightarrow \quad \frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Solving Trajectory Optimization Problem

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

The Lagrange function

$$L(s, a, \lambda) = h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t) + \lambda_{t+1}^T (f(s_t, a_t) - s_{t+1})$$

λ is also called "costate"

$$\lambda_T = h_s'(s_T) \quad \Leftrightarrow \quad \frac{\partial L}{\partial s_T} = \frac{dh}{ds}(s_T) - \lambda_T = 0$$

$$\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1} \quad \Leftrightarrow \quad \frac{\partial L}{\partial s_t} = \frac{\partial h}{\partial s}(s_t, a_t) + \left(\frac{\partial f}{\partial s}(s_t, a_t)\right)^T \lambda_{t+1} - \lambda_t = 0$$

$$a_t = \arg\min_a h_a'(s_t, a_t) + \left(f_a'(s_t, a_t)\right)^T \lambda_{t+1} \quad \Leftrightarrow \quad \frac{\partial L}{\partial a_t} = \frac{\partial h}{\partial a}(s_t, a_t) + \left(\frac{\partial f}{\partial a}(s_t, a_t)\right)^T \lambda_{t+1} = 0$$

$$s_{t+1} = f(s_t, a_t) \quad \Leftrightarrow \quad \frac{\partial L}{\partial \lambda_t} = f(s_t, a_t) - s_{t+1} = 0$$

Pontryagin's Maximum Principle for discrete systems

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

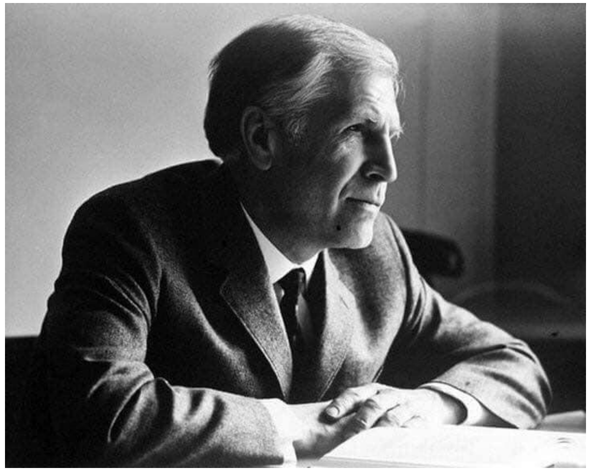
$\boxed{\lambda \text{ is also called “costate”}}$

$$\lambda_T = h_s'(s_T)$$

$$\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1}$$

$$a_t = \arg\min_a h_a'(s_t, a_t) + \left(f_a'(s_t, a_t)\right)^T \lambda_{t+1}$$

$$s_{t+1} = f(s_t, a_t)$$



Lev Semyonovich Pontryagin

Pontryagin's Maximum Principle for discrete systems

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

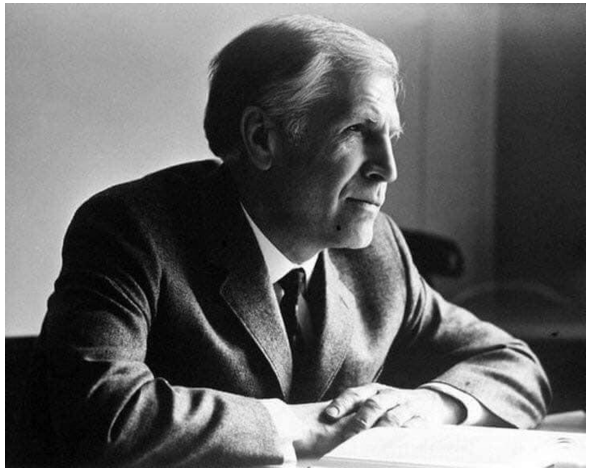
$$s_{t+1} = f(s_t, a_t)$$

"costate" dynamics

$$\lambda_T = h'_s(s_T)$$

$$\lambda_t = h'_s(s_t, a_t) + \left(f'_s(s_t, a_t)\right)^T \lambda_{t+1}$$

$$a_t = \arg\min_a h'_a(s_t, a_t) + \left(f'_a(s_t, a_t)\right)^T \lambda_{t+1}$$



Lev Semyonovich Pontryagin

Pontryagin's Maximum Principle for discrete systems

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \qquad \text{for } 0 \le t < T$$

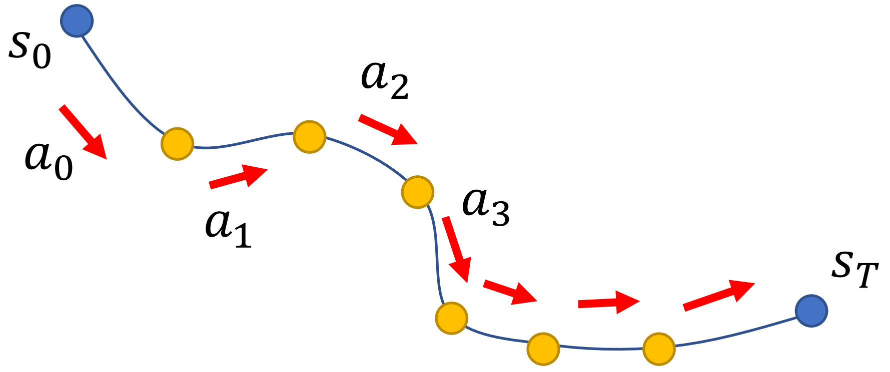
$$s_{t+1} = f(s_t, a_t)$$

"costate" dynamics

$$\lambda_T = h_s'(s_T)$$

$$\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1}$$

$$a_t = \arg\min_a h_a'(s_t, a_t) + \left(f_a'(s_t, a_t)\right)^T \lambda_{t+1}$$



Shooting method:

Given initial control $\{a_t\}$, iteratively update them by

- Forward pass: compute $\{s_t\}$ using

  $s_{t+1} = f(s_t, a_t), \quad t = 0, \dots, T-1$

- Backward pass: compute $\{\lambda_t\}$ using

  $\lambda_T = h_s'(s_T)$

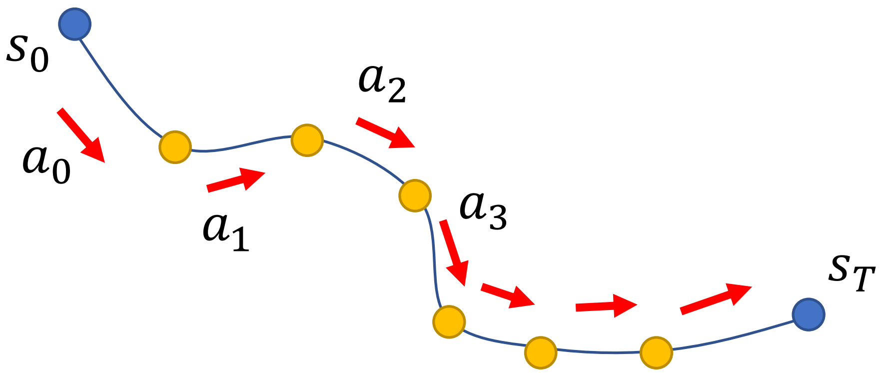
  $\lambda_t = h_s'(s_t, a_t) + \left(f_s'(s_t, a_t)\right)^T \lambda_{t+1}, \quad t = T-1, \dots, 0$

- Update $\{a_t\}$ using gradient descent

Optimal Control

Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Trajectory Optimization

Pontryagin's Maximum Principle

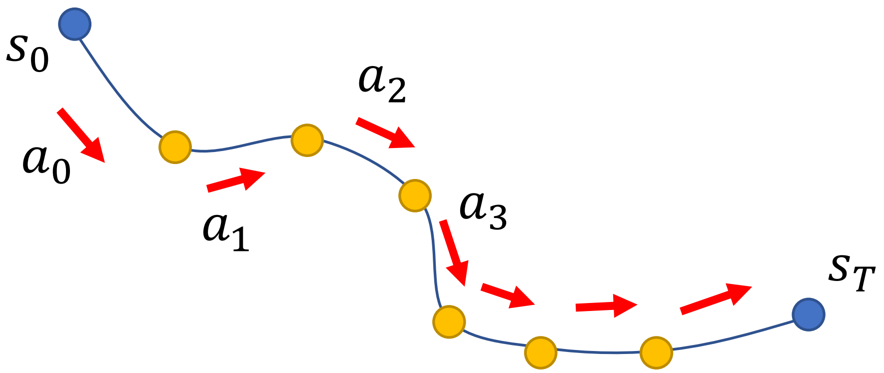
$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to $\quad f(s_t, a_t) - s_{t+1} = 0 \quad$ for $0 \le t < T$

Optimal Control

Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Shooting method directly applies PMP. However, it does not scale well to complicated problems such as motion control…

Need to be combined with collocation method, multiple shooting, etc. for those problems.

Or use derivative-free approaches.

Trajectory Optimization

Pontryagin's Maximum Principle

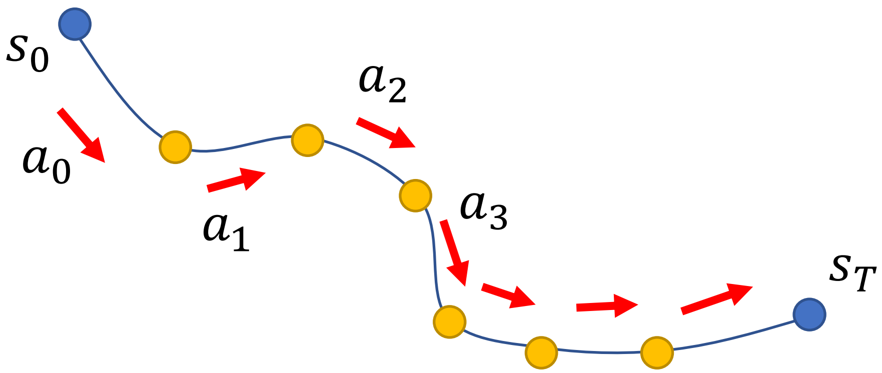
$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to $f(s_t, a_t) - s_{t+1} = 0 \quad$ for $0 \le t < T$

Optimal Control

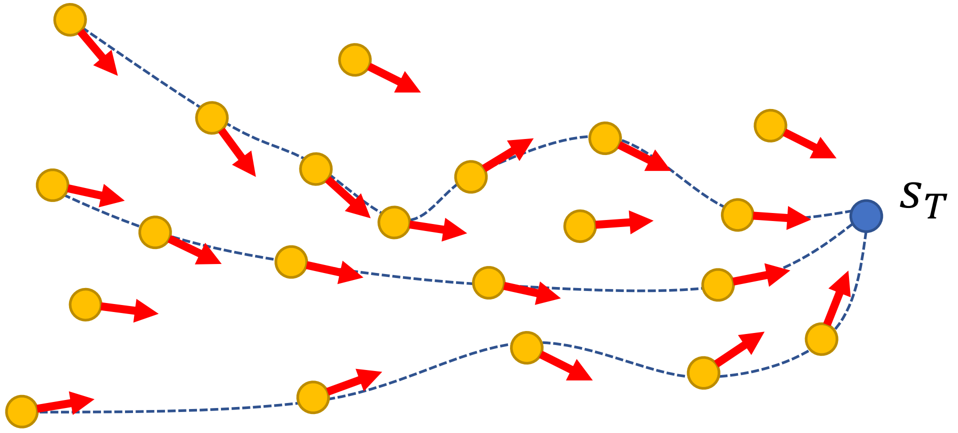
Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Feedback Control:

for any state s_t at time t, find the corresponding action $a_t = \pi(s_t, t)$ that eventually reach the goal



Trajectory Optimization

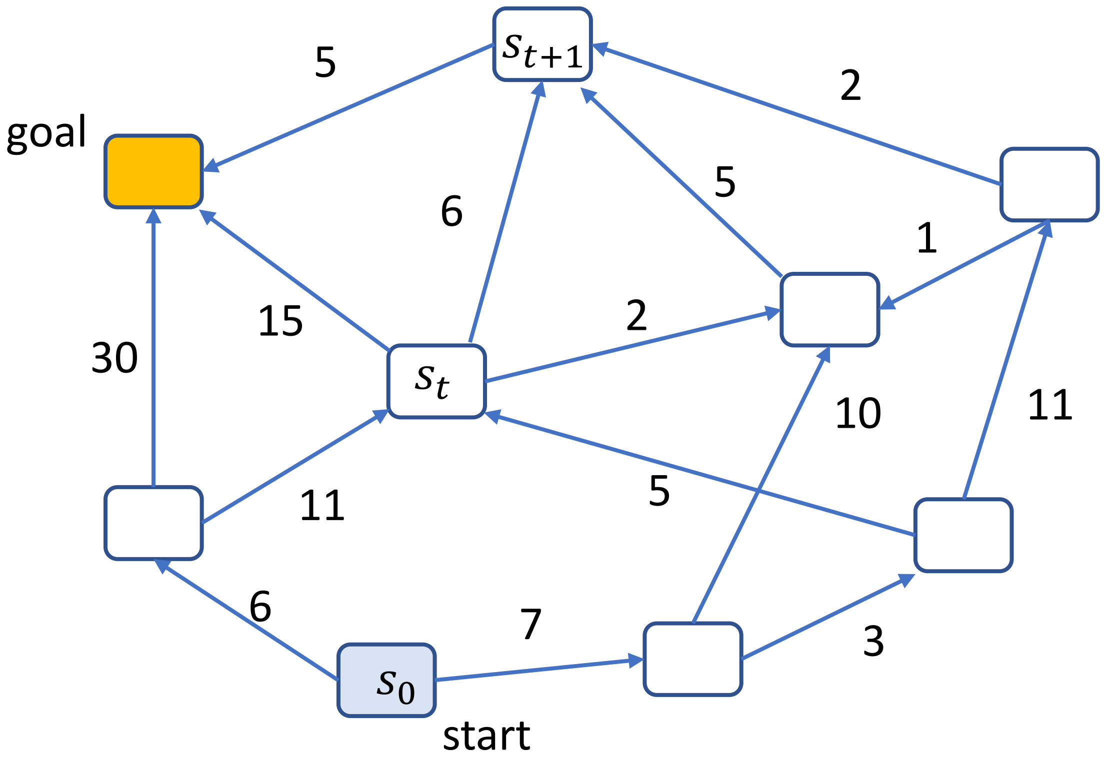
Pontryagin's Maximum Principle

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to $f(s_t, a_t) - s_{t+1} = 0 \quad$ for $0 \le t < T$

Dynamic Programming

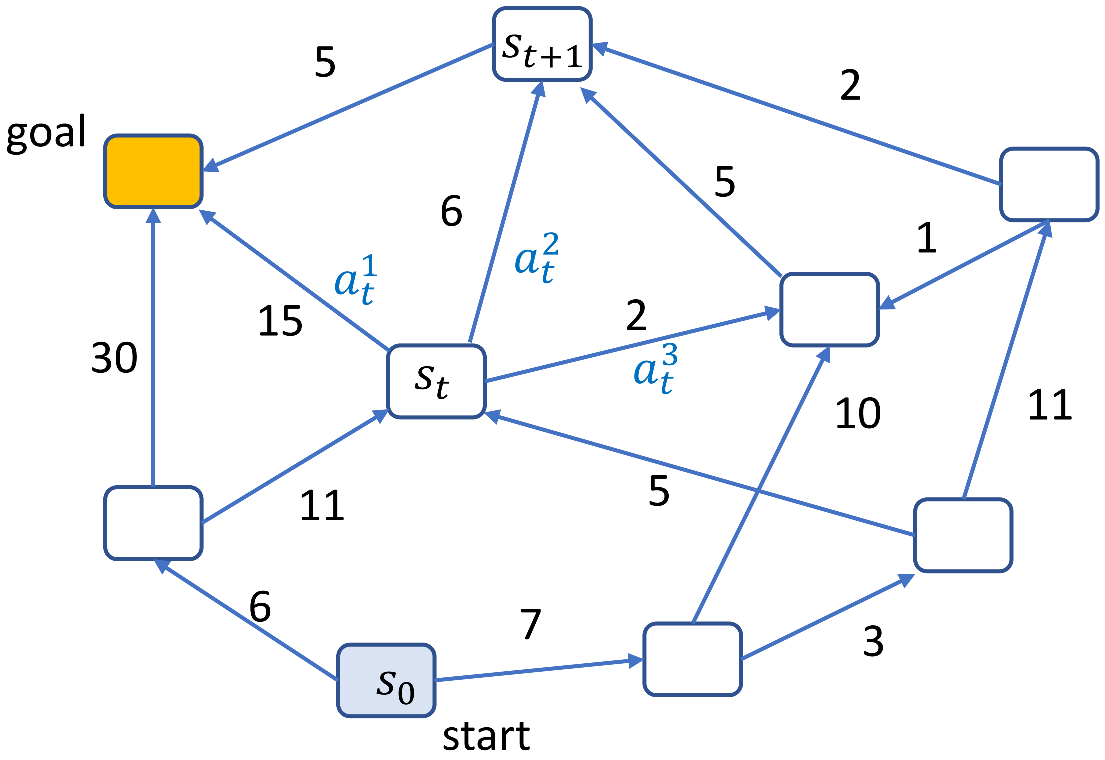
Dynamic Programming



Find a path $\{s_t\}$ that minimizes

$$J(s_0) = \sum_{t=0} h(s_t, s_{t+1})$$

Dynamic Programming



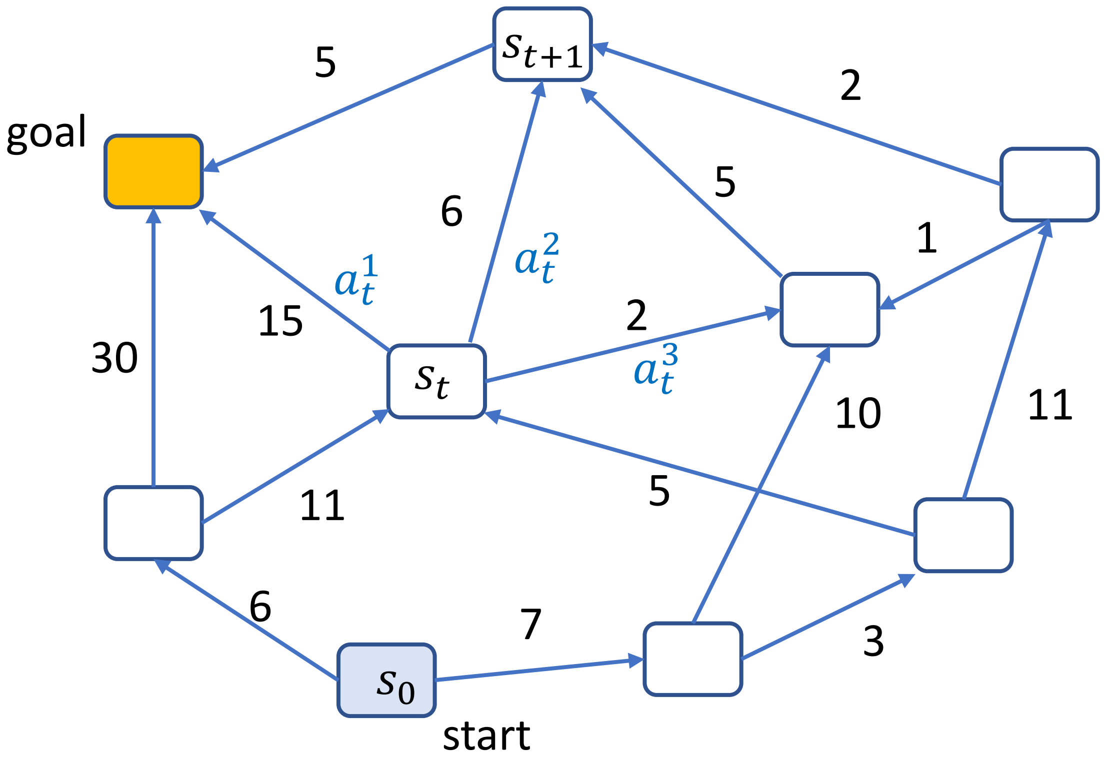
Find a sequence of action $\{a_t\}$ that minimizes

$$J(s_0) = \sum_{t=0} h(s_t, a_t)$$

subject to

$$s_{t+1} = f(s_t, a_t)$$

Dynamic Programming



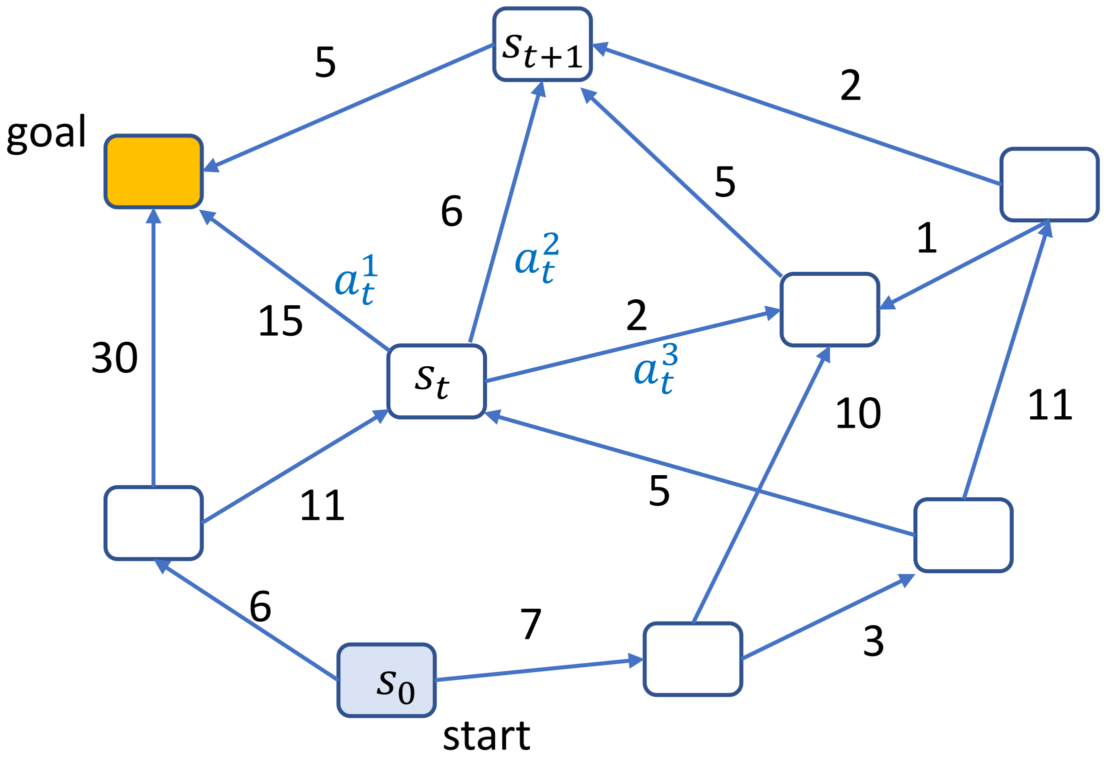
Find a policy $a_t = \pi(s_t, t)$ that minimizes

$$J(s_0) = \sum_{t=0} h(s_t, a_t)$$

subject to

$$s_{t+1} = f(s_t, a_t)$$

Dynamic Programming



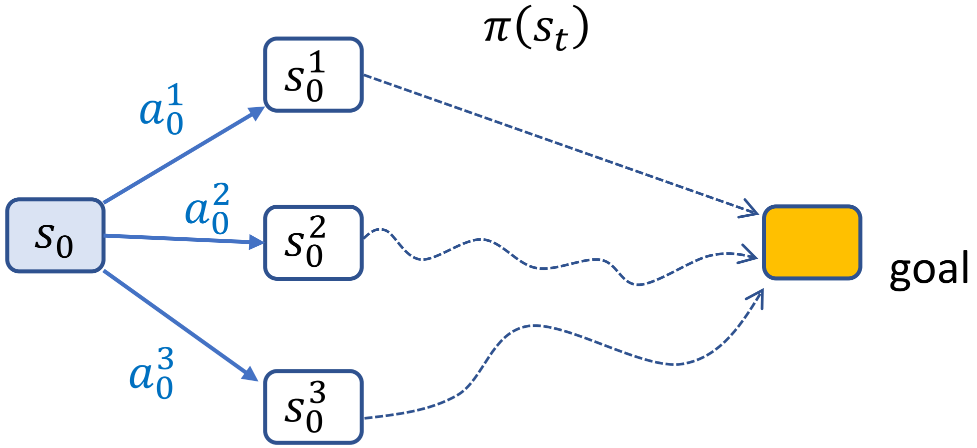
Find a policy $a_t = \pi(s_t)$ that minimizes

$$J(s_0) = \sum_{t=0} h(s_t, a_t)$$

subject to

$$s_{t+1} = f(s_t, a_t)$$

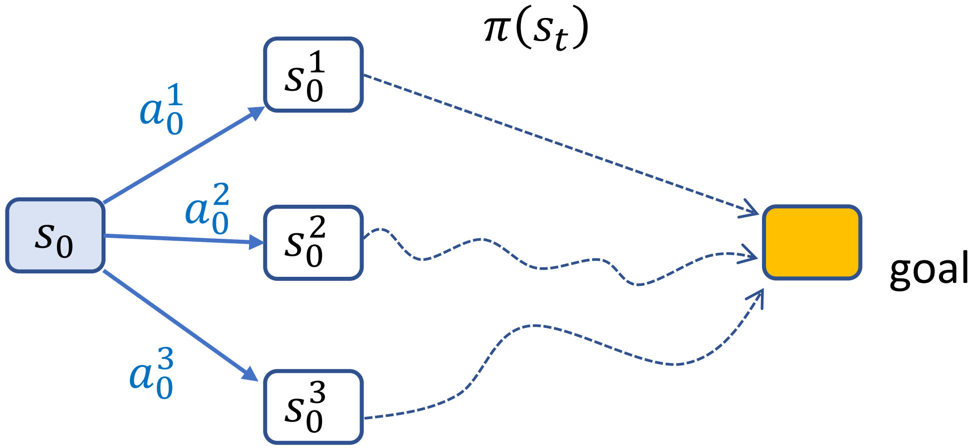
Bellman’s Principle of Optimality



An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Richard E. Bellman

Bellman's Principle of Optimality

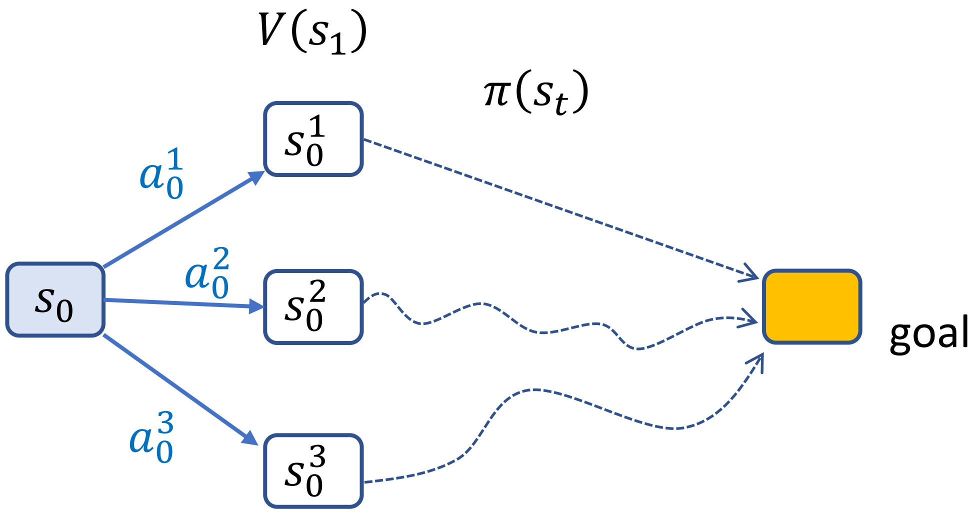


Richard E. Bellman

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

* The problem is said to have optimal substructure

Bellman's Principle of Optimality



Richard E. Bellman

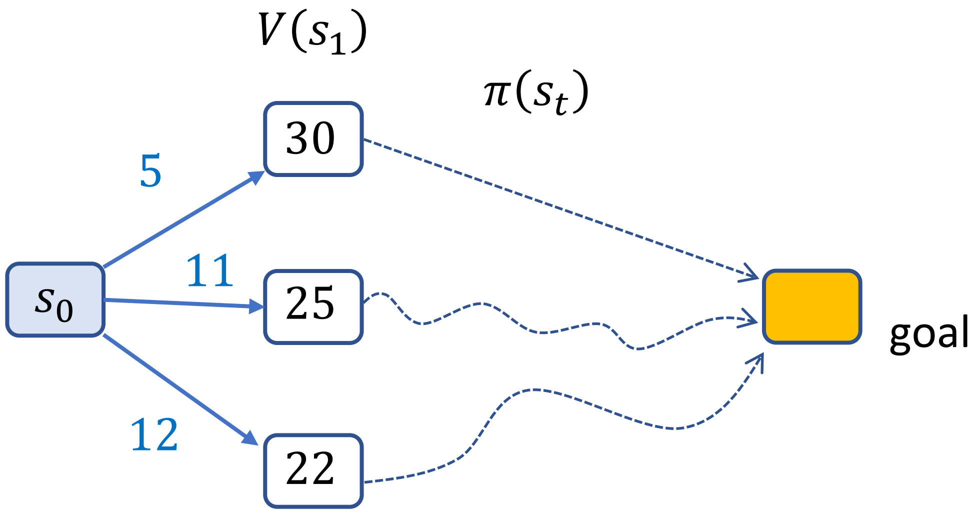
Value of a state $V(s)$:

- the minimal total cost for finishing the task starting from s

⇳

- the total cost for finishing the task starting from s using the optimal policy

Bellman's Principle of Optimality



Richard E. Bellman

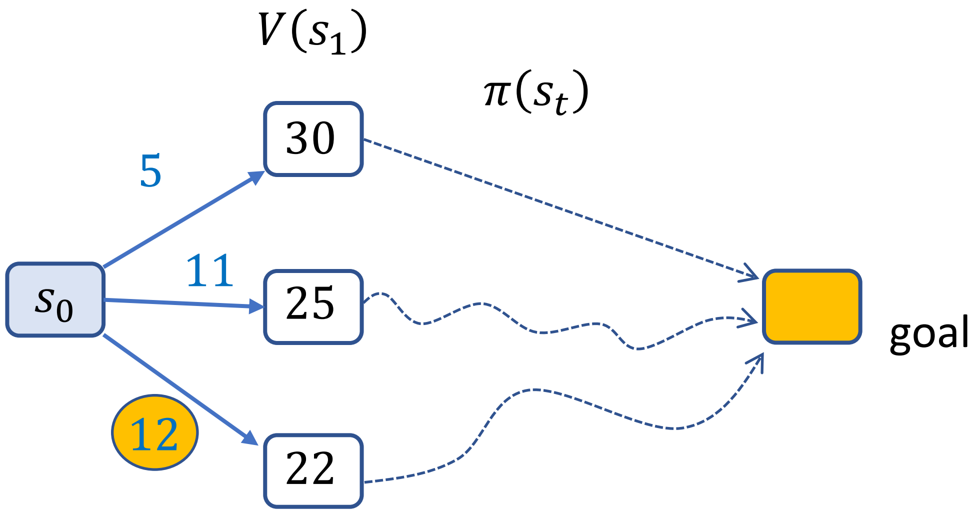
Value of a state $V(s)$:

- the minimal total cost for finishing the task starting from s

⇳

- the total cost for finishing the task starting from s using the optimal policy

Bellman's Principle of Optimality



Richard E. Bellman

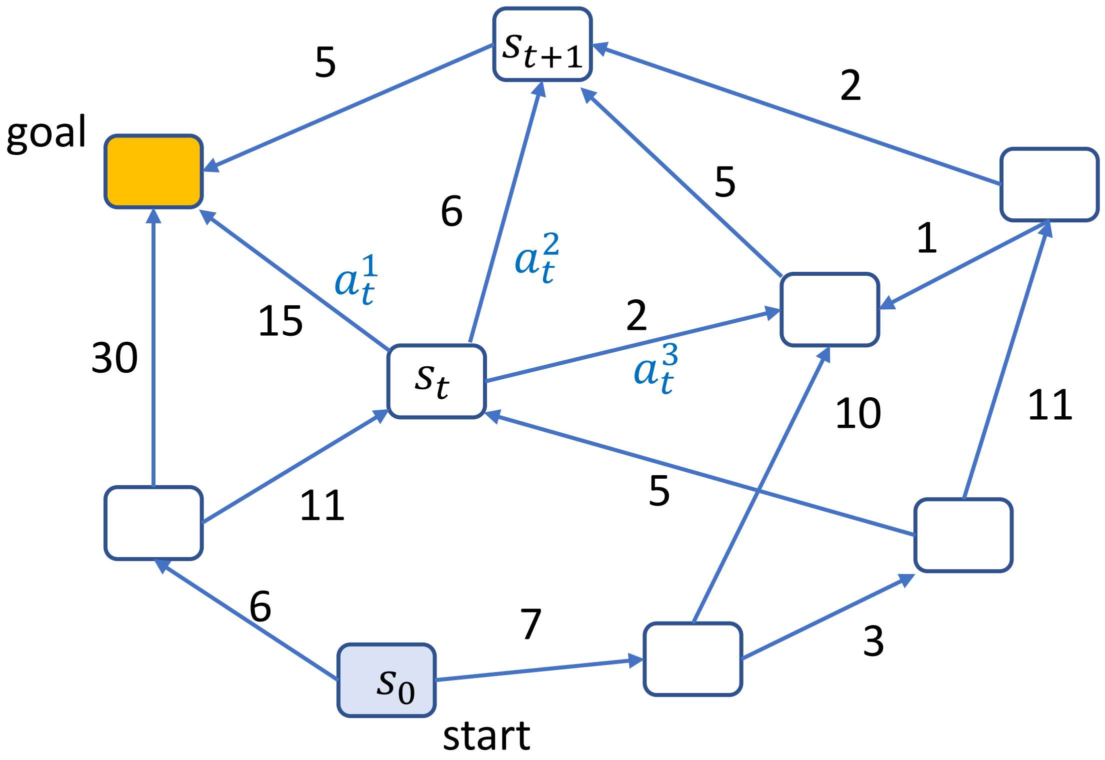
Value of a state $V(s)$:

- the minimal total cost for finishing the task starting from s

⇕

- the total cost for finishing the task starting from s using the optimal policy

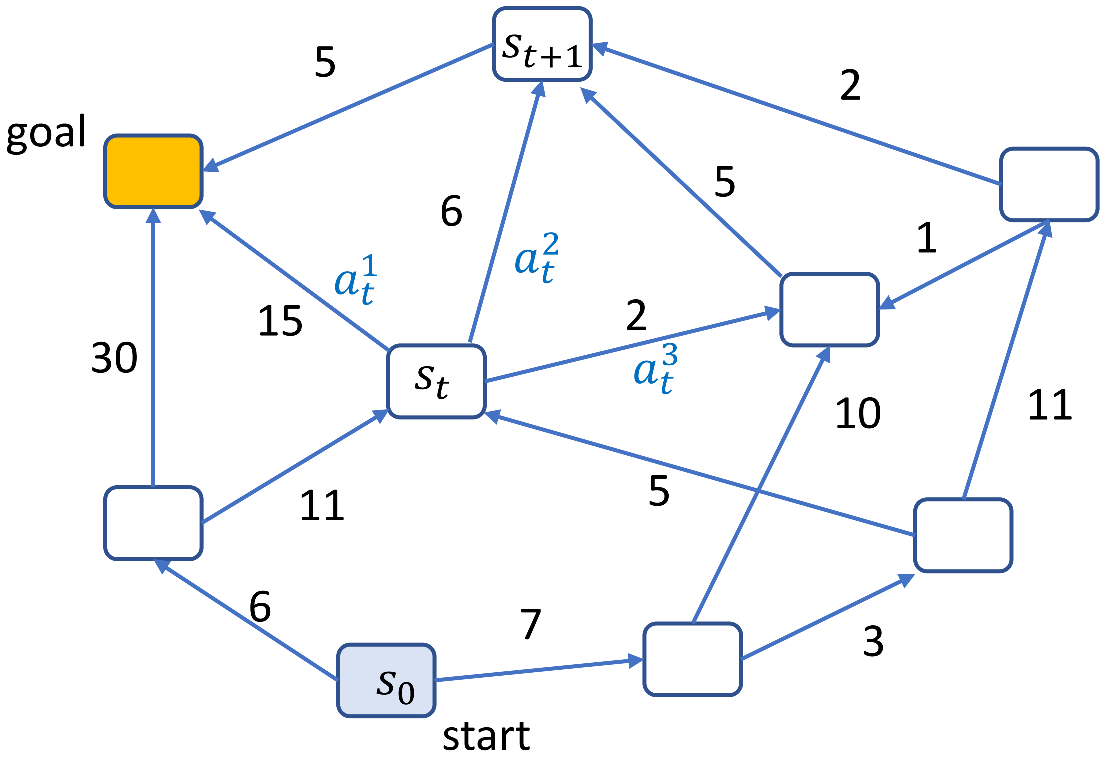
The Bellman Equation



The value function is then

$$V(s) = \min_{\pi} \sum_{t=0} h(s_t, a_t) \Big|_{s_0 = s}$$

The Bellman Equation



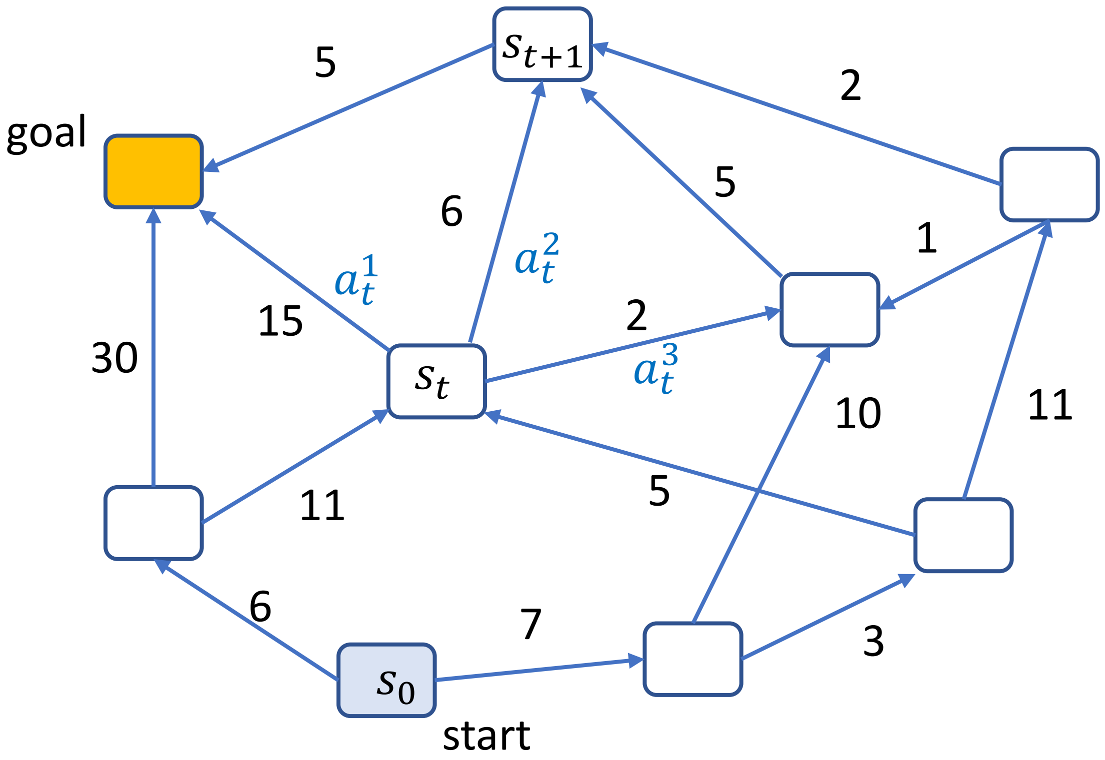
The value function is then

$$V(s) = \min_{\pi} \sum_{t=0} h(s_t, a_t) \Big|_{s_0 = s}$$

$$V(s_0) = \min_{\pi} \sum_{t=0} h(s_t, a_t)$$

$$= \min_{a_0} \left(h(s_0, a_0) + \min_{\pi} \sum_{t=1} h(s_t, a_t) \right)$$

The Bellman Equation



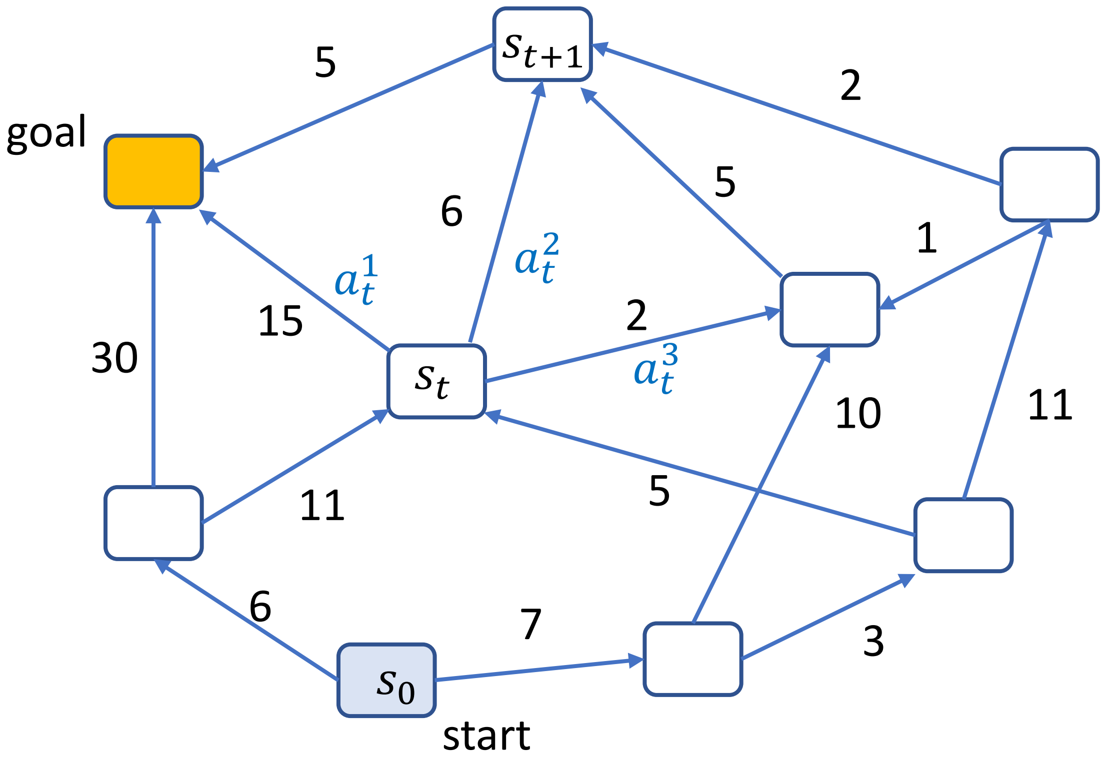
The value function is then

$$V(s) = \min_{\pi} \sum_{t=0} h(s_t, a_t) \Big|_{s_0 = s}$$

$$V(s_0) = \min_{\pi} \sum_{t=0} h(s_t, a_t)$$

$$= \min_{a_0} \left(h(s_0, a_0) + \min_{\pi} \sum_{t=1} h(s_t, a_t) \right)$$

The Bellman Equation



The value function is then

$$V(s) = \min_{\pi} \sum_{t=0} h(s_t, a_t) \Big|_{s_0 = s}$$

$$V(s_0) = \min_{\pi} \sum_{t=0} h(s_t, a_t)$$

$$= \min_{a_0} \Big(h(s_0, a_0) + V\big(s_1 = f(s_0, a_0)\big) \Big)$$

The Bellman Equation

Mathematically, an optimal value function $V(s)$ can be defined recursively as:

$$V(s) = \min_{a} \left(h(s, a) + V\big(f(s, a)\big) \right)$$

The Bellman Equation

Mathematically, an optimal value function $V(s)$ can be defined recursively as:

$$V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big) \right)$$

If we know this value function, the optimal policy can be computed as

$$\pi(s) = \arg\min_a \left(h(s,a) + V\big(f(s,a)\big) \right)$$

The Bellman Equation

Mathematically, an optimal value function $V(s)$ can be defined recursively as:

$$V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big) \right)$$

If we know this value function, the optimal policy can be computed as

$$\pi(s) = \arg\min_a \left(h(s,a) + V\big(f(s,a)\big) \right)$$

Or,

$$\pi(s) = \arg\min_a Q(s,a)$$

where $\quad Q(s,a) = h(s,a) + V\big(f(s,a)\big)$

Q-function
State-action value function

The Bellman Equation

Mathematically, an optimal value function $V(s)$ can be defined recursively as:

$$V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$$

Learning $V(s)$ and/or $Q(s,a)$ is the core of optimal control / reinforcement learning methods

If we know this value function, the optimal policy can be computed as

$$\pi(s) = \arg\min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$$

This arg max can be easily computed for discrete control problems.

But there are not always closed-forms solution for continuous control problems.

Or,

$$\pi(s) = \arg\min_a Q(s,a)$$

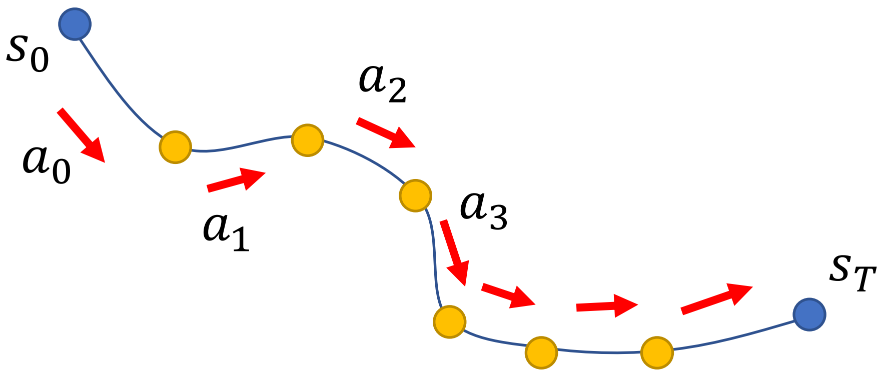
where $Q(s,a) = h(s,a) + V\big(f(s,a)\big)$

Q-function
State-action value function

Optimal Control

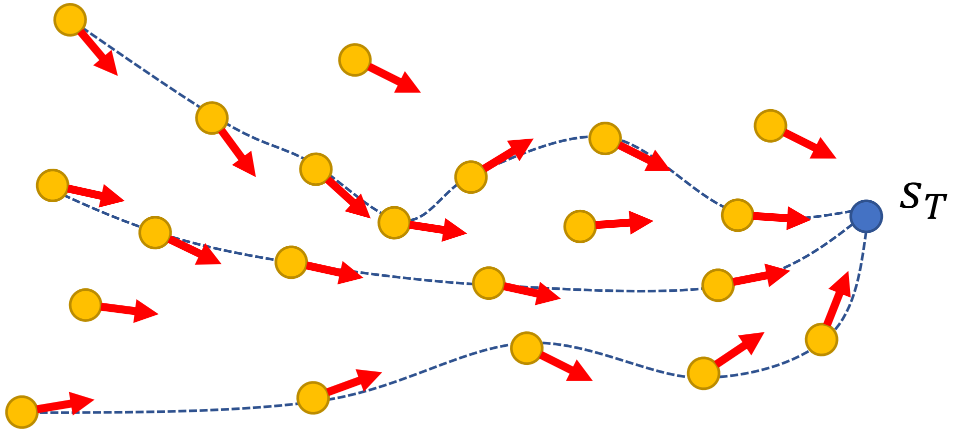
Open-loop Control:

given a start state s_0, compute sequence of actions $\{a_t\}$ to reach the goal



Feedback Control:

for any state s_t at time t, find the corresponding action $a_t = \pi(s_t, t)$ that eventually reach the goal



Trajectory Optimization

Pontryagin's Maximum Principle

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

subject to $f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$

Dynamic Programming

Bellman's Principle of Optimality

Linear Quadratic Regulator (LQR)

$$\min h(s_T) + \sum_{t=0}^{T-1} h(s_t, a_t)$$

objective function

subject to

$$f(s_t, a_t) - s_{t+1} = 0$$

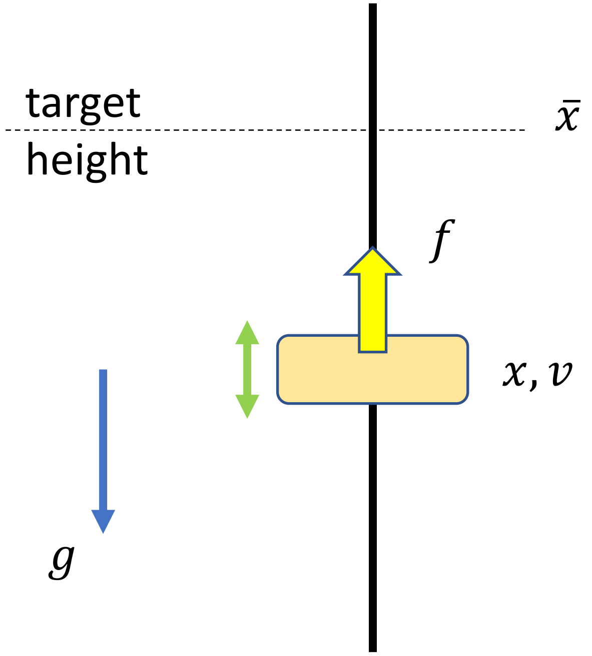
dynamic function

for $0 \leq t < T$

- LQR is a special class of optimal control problems with
  - Linear dynamic function
  - Quadratic objective function

A very simple example

$$f = k_p(\bar{x} - x) - k_d v$$

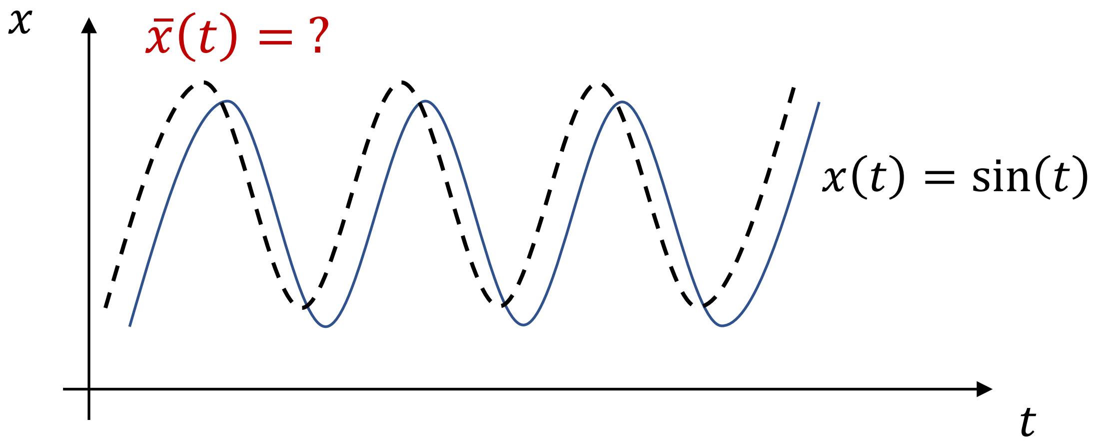


Compute a target trajectory $\bar{x}(t)$
such that the simulated trajectory $x(t)$
is a sine curve.

$$\min_{\{x_n, v_n, \bar{x}_n\}} \sum_{n=0}^{N} (\sin(t_n) - x_n)^2 + \sum_{n=0}^{N} \bar{x}_n^2$$

$$s.t. \quad v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$



Linear Quadratic Regulator (LQR)

- LQR is a special class of optimal control problems with
  - Linear dynamic function
  - Quadratic objective function

objective function

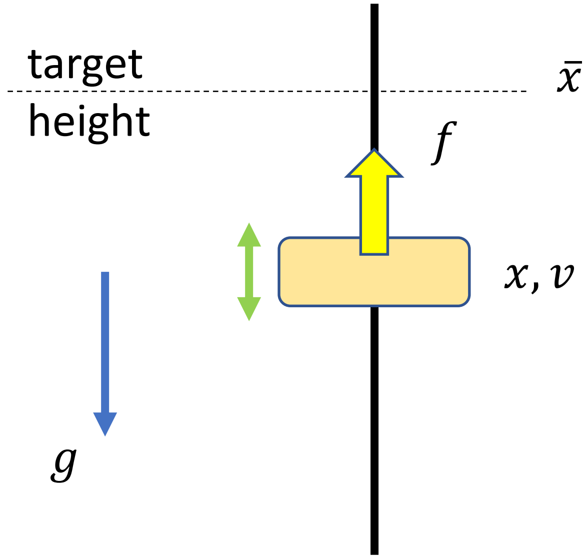
$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \qquad \text{for } 0 \le t < T$$

dynamic function

$$f = k_p(\bar{x} - x) - k_d v$$



$$v_{n+1} = v_n + h\big(k_p(\bar{x}_n - x_n) - k_d v_n\big)$$

$$x_{n+1} = x_n + h v_{n+1}$$

Linear Quadratic Regulator (LQR)

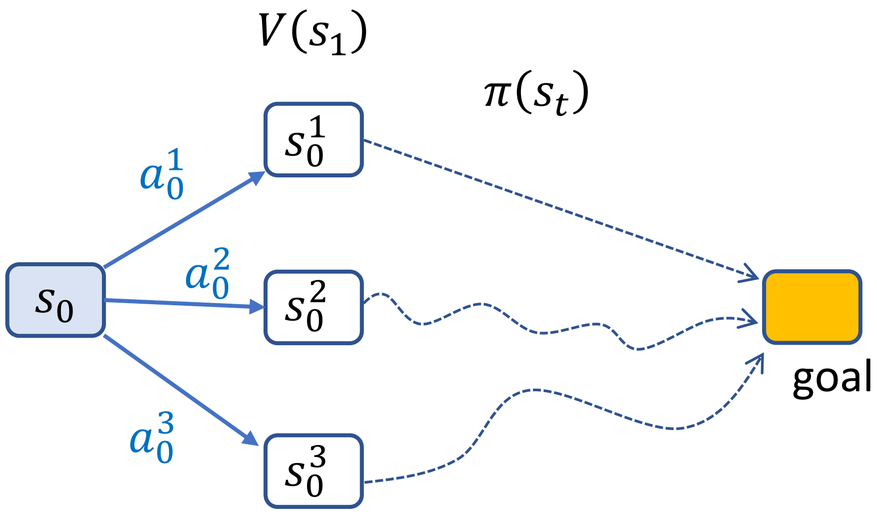
The Bellman Equations:

$$V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big) \right)$$

Or,

$$V(s) = \min_a Q(s,a)$$

$$Q(s,a) = h(s,a) + V\big(f(s,a)\big)$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

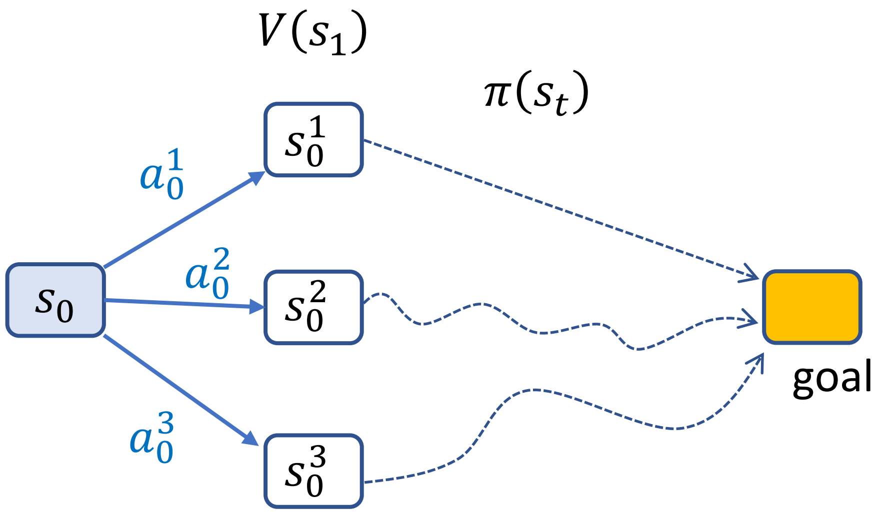
subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

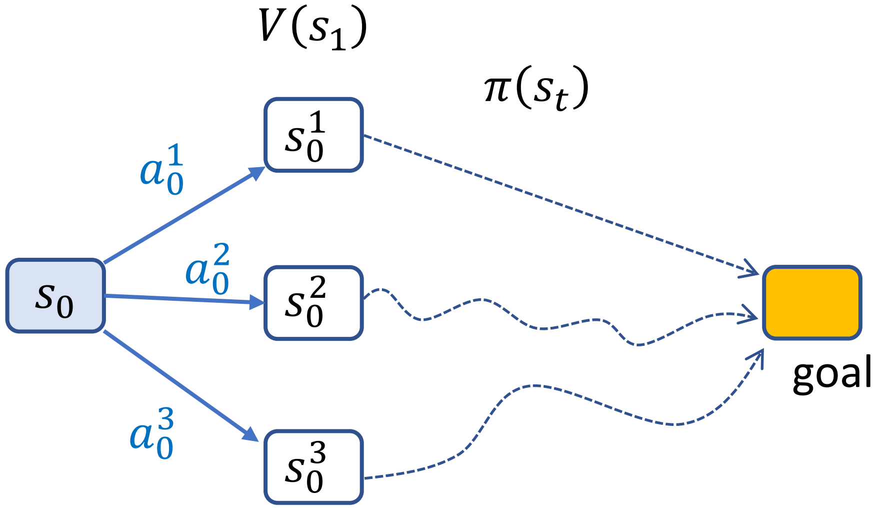
$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step $T-1$:

bellman equation

$$Q(s_{T-1}, a_{T-1}) = s_{T-1}^T Q_{T-1} s_{T-1} + a_{T-1}^T R_{T-1} a_{T-1} + V(s_T)$$

$$= s_{T-1}^T Q_{T-1} s_{T-1} + a_{T-1}^T R_{T-1} a_{T-1} + s_T^T Q_T s_T$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \leq t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step $T-1$:

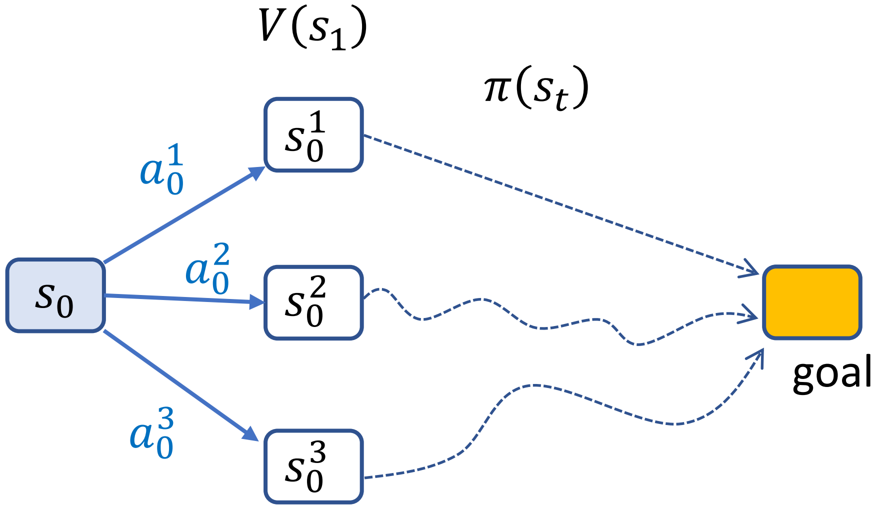
bellman equation

$$Q(s_{T-1}, a_{T-1}) = s_{T-1}^T Q_{T-1} s_{T-1} + a_{T-1}^T R_{T-1} a_{T-1} + V(s_T)$$

$$= s_{T-1}^T Q_{T-1} s_{T-1} + a_{T-1}^T R_{T-1} a_{T-1} + s_T^T Q_T s_T$$

apply dynamic function

$$= s_{T-1}^T Q_{T-1} s_{T-1} + a_{T-1}^T R_{T-1} a_{T-1}$$
$$+ (A_{T-1} s_{T-1} + B_{T-1} a_{T-1})_T^T Q_T (A_{T-1} s_{T-1} + B_{T-1} a_{T-1})$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

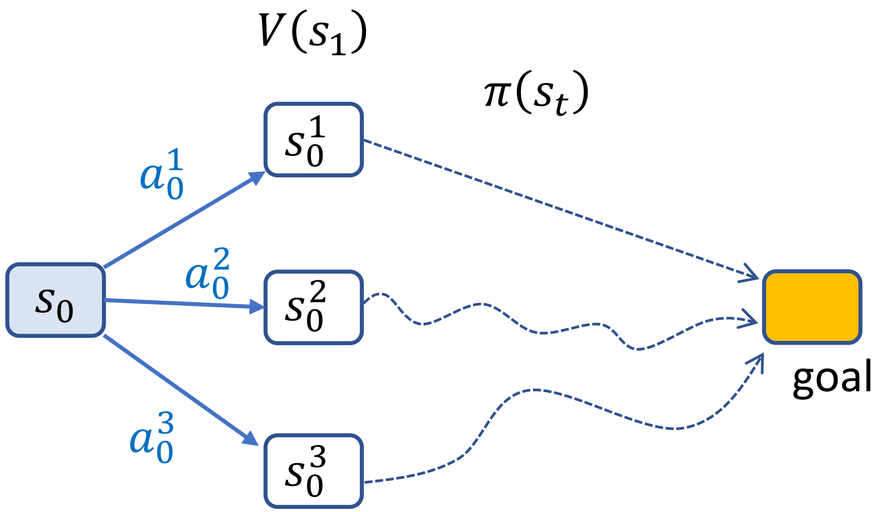
Linear Quadratic Regulator (LQR)

Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step $T-1$:

$$\begin{aligned} Q(s_{T-1}, a_{T-1}) &= s_{T-1}^T \left(Q_{T-1} + A_{T-1}^T Q_T A_{T-1}\right) s_{T-1} \\ &+ a_{T-1}^T \left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right) a_{T-1} \\ &+ 2 s_{T-1}^T A_{T-1}^T Q_T B_{T-1} a_{T-1} \end{aligned}$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

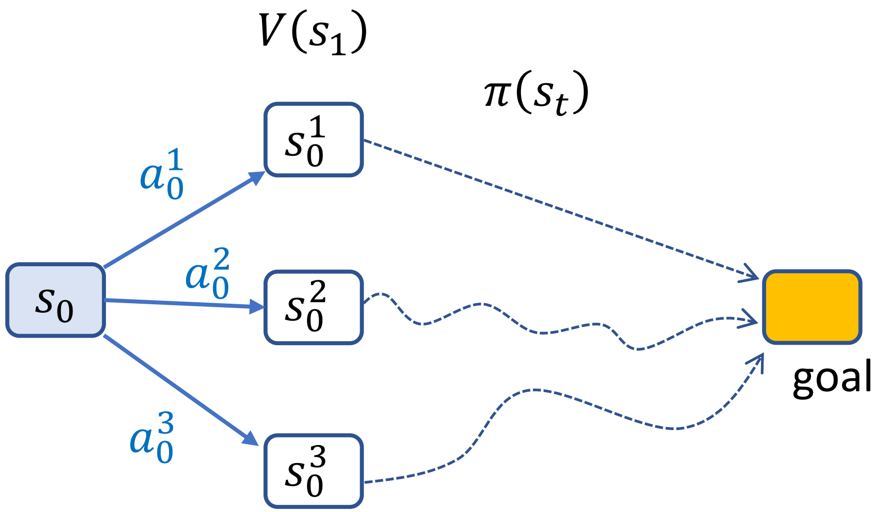
Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step $T-1$:

$$\begin{aligned} Q(s_{T-1}, a_{T-1}) &= s_{T-1}^T \left(Q_{T-1} + A_{T-1}^T Q_T A_{T-1}\right) s_{T-1} \\ &+ a_{T-1}^T \left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right) a_{T-1} \\ &+ 2 s_{T-1}^T A_{T-1}^T Q_T B_{T-1} a_{T-1} \end{aligned}$$

$$\begin{aligned} a_{T-1}^* &= \arg\min_{a_{T-1}} Q(s_{T-1}, a_{T-1}) \\ &= -\left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right)^{-1} B_{T-1}^T Q_T A_{T-1} s_{T-1} \end{aligned}$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

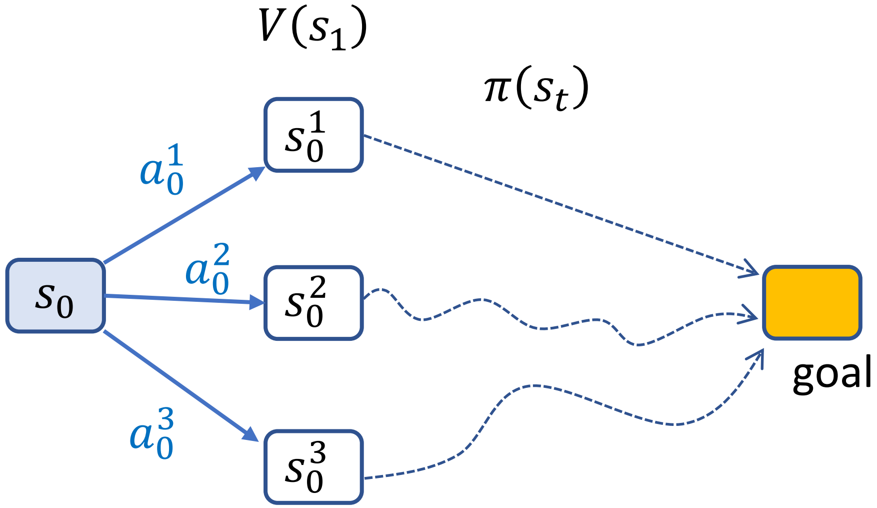
Solve for step $T-1$:

$$\begin{aligned} Q(s_{T-1}, a_{T-1}) \; &= s_{T-1}^T \left(Q_{T-1} + A_{T-1}^T Q_T A_{T-1}\right) s_{T-1} \\ &+ a_{T-1}^T \left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right) a_{T-1} \\ &+ 2 s_{T-1}^T A_{T-1}^T Q_T B_{T-1} a_{T-1} \end{aligned}$$

$$\begin{aligned} a_{T-1}^* &= \arg\min_{a_{T-1}} Q(s_{T-1}, a_{T-1}) \\ &= -K_{T-1} s_{T-1} \end{aligned}$$

Linear feedback policy!

$$K_{T-1} = \left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right)^{-1} B_{T-1}^T Q_T A_{T-1}$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

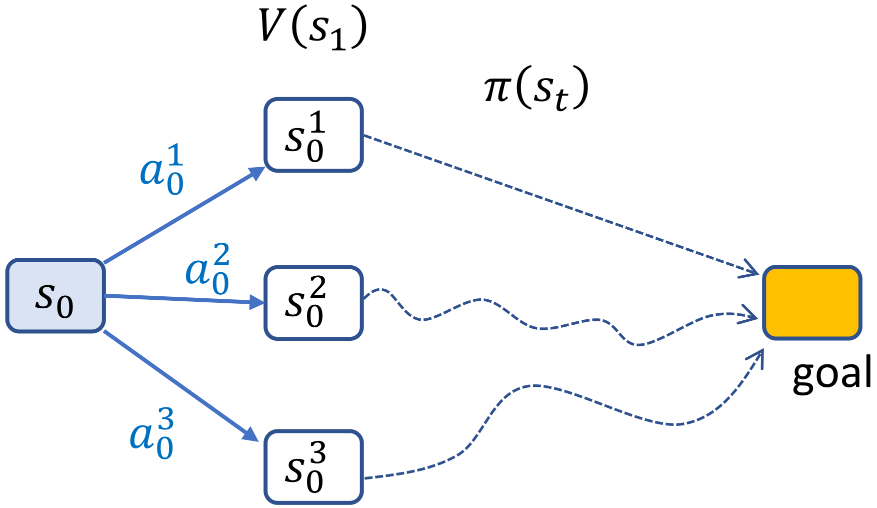
$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step $T-1$:

$$\begin{aligned} Q(s_{T-1}, a_{T-1}) \ &= s_{T-1}^T \left(Q_{T-1} + A_{T-1}^T Q_T A_{T-1}\right) s_{T-1} \\ &+ a_{T-1}^T \left(R_{T-1} + B_{T-1}^T Q_T B_{T-1}\right) a_{T-1} \\ &+ 2 s_{T-1}^T A_{T-1}^T Q_T B_{T-1} a_{T-1} \end{aligned}$$

$$\begin{aligned} V(s_{T-1}) &= \min_{a_{T-1}} Q(s_{T-1}, a_{T-1}) \\ &= s_{T-1}^T P_{T-1} s_{T-1} \qquad P_{T-1} = \cdots \end{aligned}$$

Quadratic value function!



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

Solve for the last step:

$$V(s_T) = \min_{a_T} s_T^T Q_T s_T = s_T^T Q_T s_T$$

Solve for step t, $t = T-1, T-2, \dots, 0$:

$$a_t^* = -K_t s_t$$

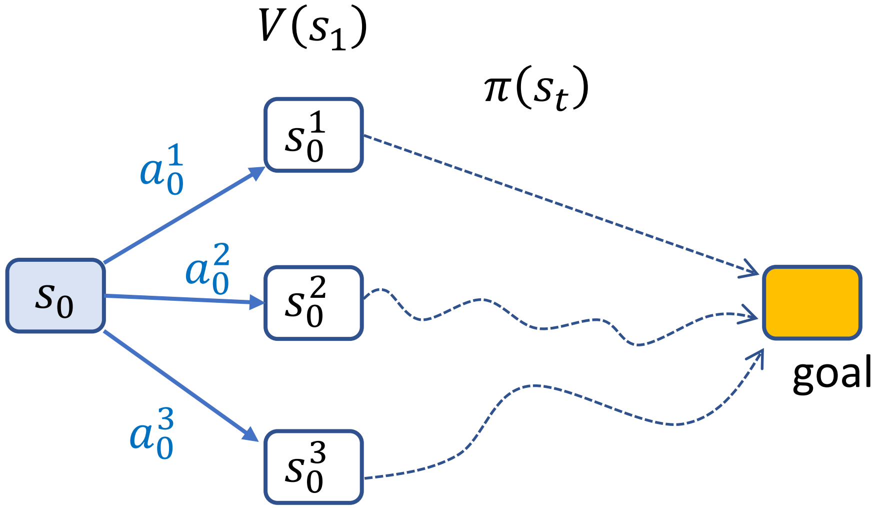
Linear feedback policy!

$$K_t = \left(R_t + B_t^T P_{t+1} B_t\right)^{-1} B_t^T P_{t+1} A_t$$

$$V(s_t) = s_t^T P_t s_t$$

Quadratic value function!

$$P_t = F(P_{t+1}) = \cdots$$



$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

Linear Quadratic Regulator (LQR)

- LQR is a special class of optimal control problems with
  - Linear dynamic function
  - Quadratic objective function

- Solution of LQR is a linear feedback policy

$$\min s_T^T Q_T s_T + \sum_{t=0}^{T} s_t^T Q_t s_t + a_t^T R_t a_t$$

subject to

$$s_{t+1} = A_t s_t + B_t a_t \quad \text{for } 0 \le t < T$$

$$a_t^* = -K_t s_t$$

$$K_t = \left(R_t + B_t^T P_{t+1} B_t\right)^{-1} B_t^T P_{t+1} A_t$$

Linear Quadratic Regulator (LQR)

- LQR is a special class of optimal control problems with
  - Linear dynamic function
  - Quadratic objective function

- How to deal with
  - Nonlinear dynamic function?
  - Non-quadratic objective function?

Linear Quadratic Regulator (LQR)

- Nonlinear problems

$$\min \sum_{t=0}^{T-1} h(s_t, a_t)$$

objective function

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \leq t < T$$

dynamic function

Linear Quadratic Regulator (LQR)

- Nonlinear problems

$$\min \sum_{t=0}^{T-1} h(s_t, a_t)$$

objective function

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$$

dynamic function

Approximate cost function as a quadratic function:

$$h(s_t, a_t) \approx h(\bar{s}_t, \bar{a}_t) + \nabla h(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}^T \nabla^2 h(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}$$

Linear Quadratic Regulator (LQR)

- Nonlinear problems

$$\min \sum_{t=0}^{T-1} h(s_t, a_t)$$

objective function

subject to

$$f(s_t, a_t) - s_{t+1} = 0 \quad \text{for } 0 \le t < T$$

dynamic function

Approximate cost function as a quadratic function:

$$h(s_t, a_t) \approx h(\bar{s}_t, \bar{a}_t) + \nabla h(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}^T \nabla^2 h(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}$$

Approximate dynamic function as a linear function:

$$f(s_t, a_t) \approx f(\bar{s}_t, \bar{a}_t) + \nabla f(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}$$

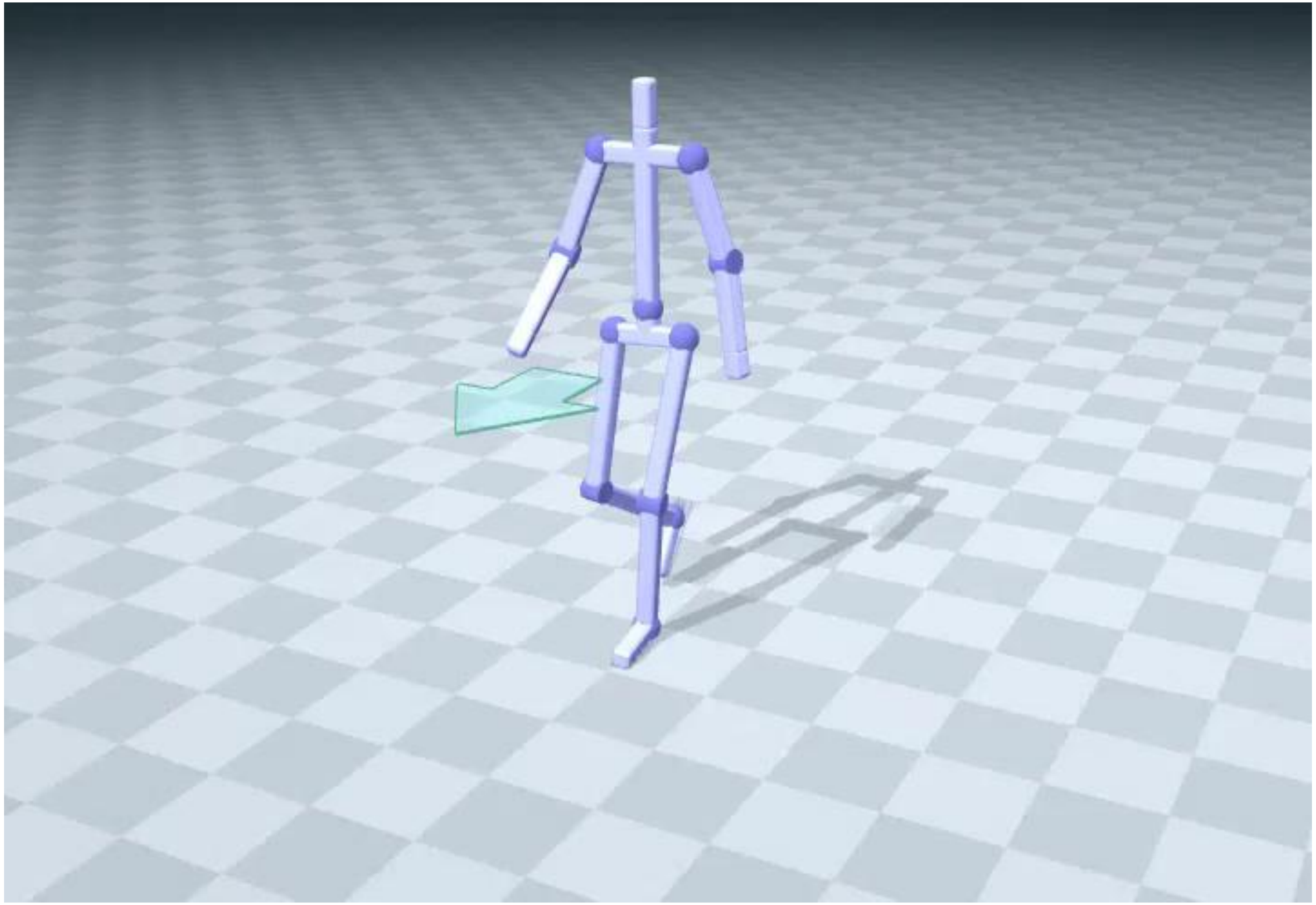
iLQR: iterative LQR

Or a quadratic function:

$$f(s_t, a_t) \approx *** + \frac{1}{2} \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}^T \nabla^2 f(\bar{s}_t, \bar{a}_t) \begin{bmatrix} s_t - \bar{s}_t \\ a_t - \bar{a}_t \end{bmatrix}$$

DDP: Differential Dynamic Programming

Locomotion Using Optimal Control



[Muico et al 2011 - Composite Control of Physically Simulated Characters]

Model-based Method vs. Model-free Method

$$\min \sum_{t=0}^{T-1} h(s_t, a_t)$$

objective function

subject to

$$s_{t+1} = f(s_t, a_t) \quad \text{for } 0 \le t < T$$

dynamic function

What if the dynamic function $f(s, a)$ is not know?

What if the dynamic function $f(s, a)$ is not accurate?

What if the system has noise?

What if the system is highly nonlinear?

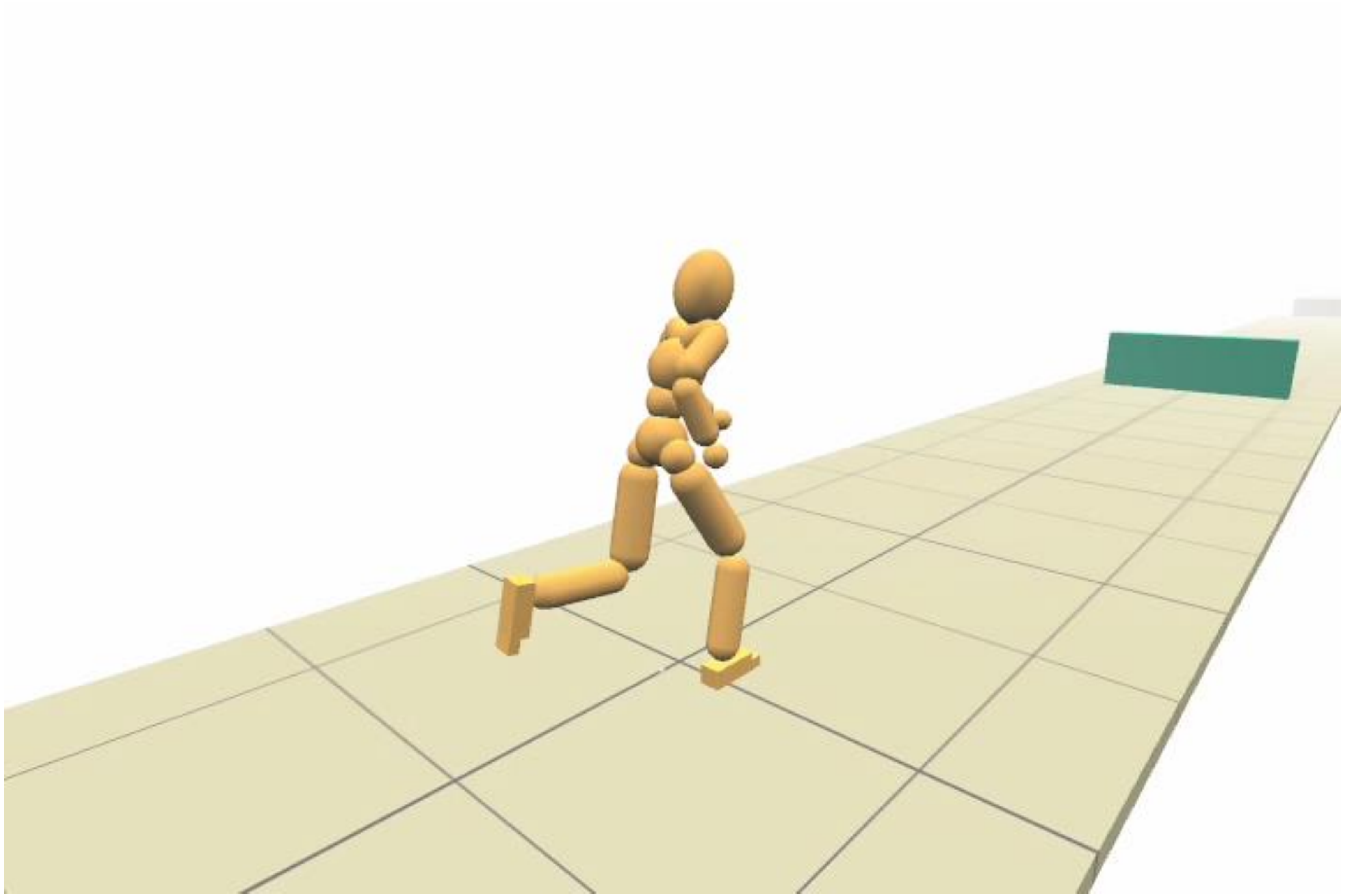
Sampling-based Policy Optimization

- Iterative methods
  - Goal: find the optimal policy $\pi(s;\theta)$ that minimize the objective $J(\theta) = \sum_{t=0} h(s_t, a_t)$
  - Initialize policy parmeters $\pi(x;\theta)$
  - Repeat:
    - Propose a set of candidate parameters $\{\theta_i\}$ according to θ
    - Simulate the agent under the control of each $\pi(\theta_i)$
    - Evaluate the objective function $J(\theta_i)$ on the simulated state-action sequences
    - Update the estimation of θ based on $\{J(\theta_i)\}$

Sampling-based Policy Optimization

- Iterative methods
  - Goal: find the optimal policy $\pi(s;\theta)$ that minimize the objective $J(\theta)=\sum_{t=0} h(s_t,a_t)$
  - Initialize policy parmeters $\pi(x;\theta)$
  - Repeat:
    - Propose a set of candidate parameters $\{\theta_i\}$ according to θ
    - Simulate the agent under the control of each $\pi(\theta_i)$
    - Evaluate the objective function $J(\theta_i)$ on the simulated state-action sequences
    - Update the estimation of θ based on $\{J(\theta_i)\}$
- Example: CMA-ES

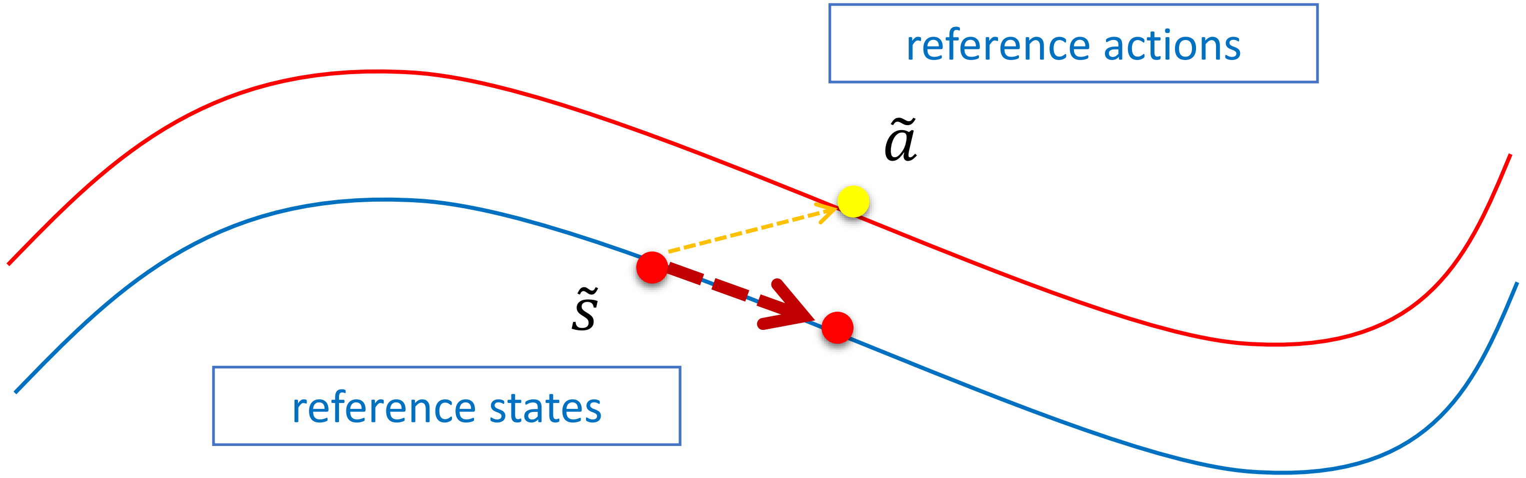
Example: Locomotion Controller with Linear Policy



[Liu et al. 2012 – Terrain Runner]

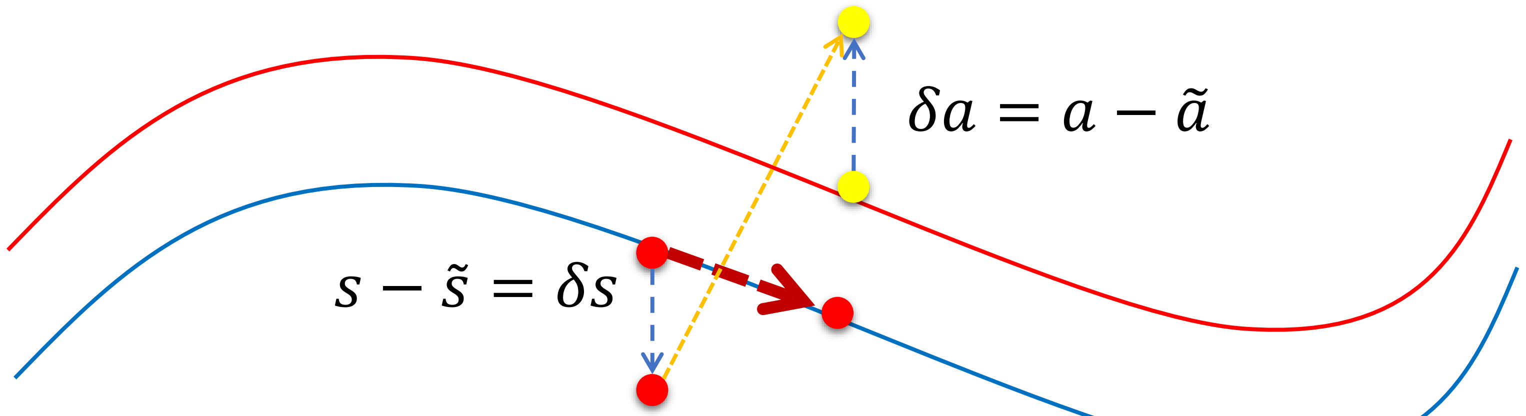
Stage 1a: Open-loop Policy

Find open-loop control using SAMCON



Stage 1b: Linear Feedback Policy

$$\boxed{\delta a = M\,\delta s + \hat{a}}$$

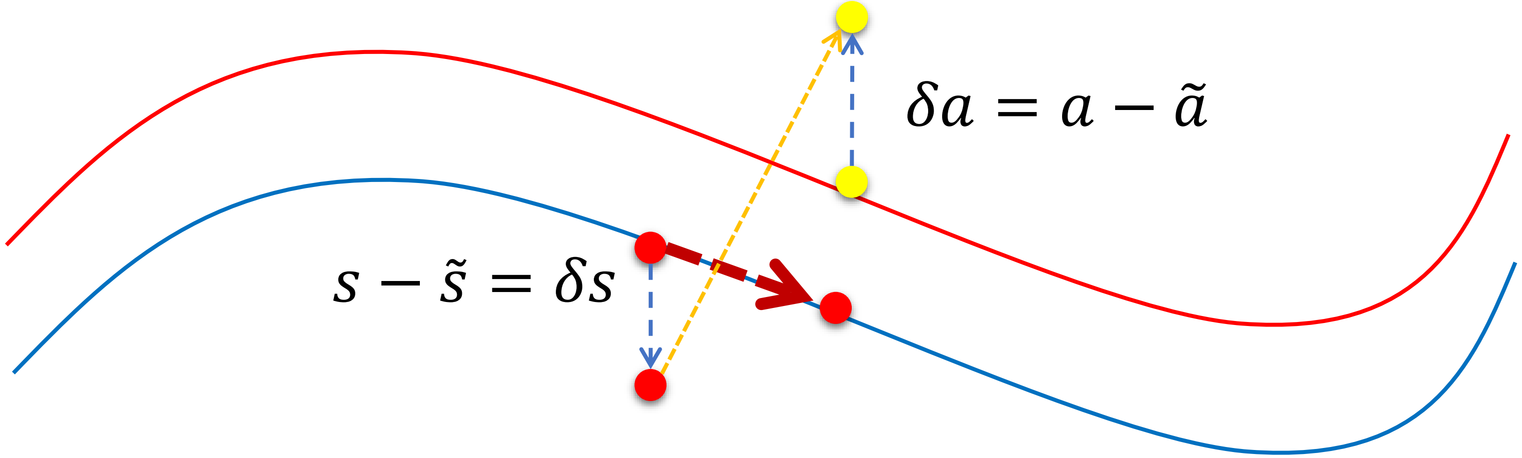


Stage 1b: Linear Feedback Policy

change in control

change in states

$$\boxed{\delta a} = M \boxed{\delta s} + \hat{a}$$



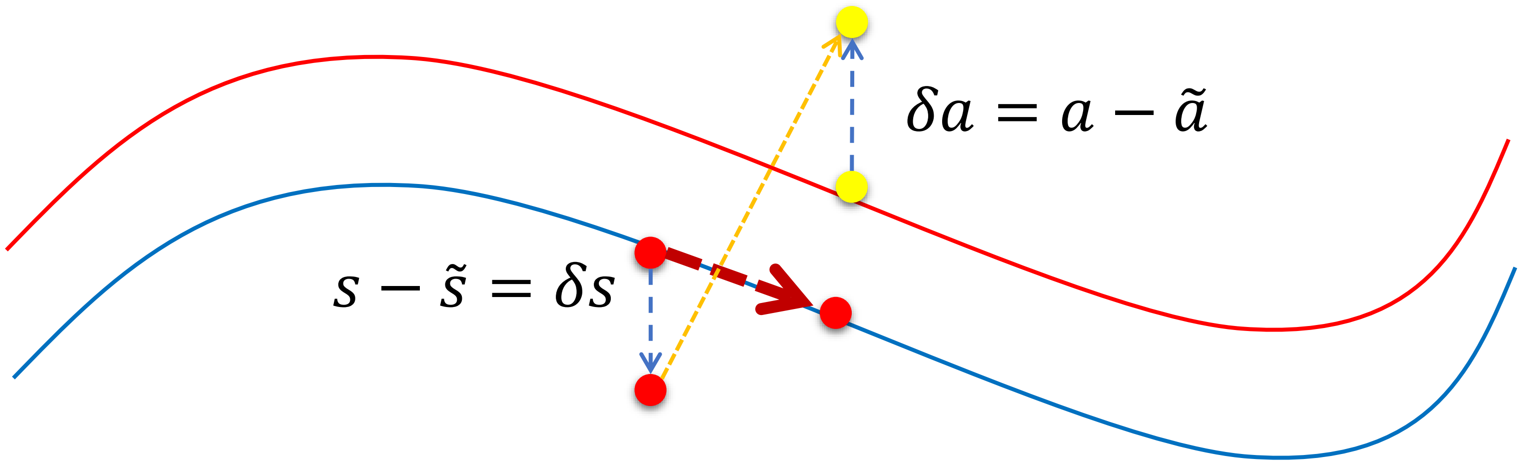
Stage 1b: Reduced-order Closed-loop Policy

$M_{ap} \cdot M_{sp}$

change in control

change in states

$$\boxed{\delta a} = \boxed{M}\,\boxed{\delta s} + \hat{a}$$



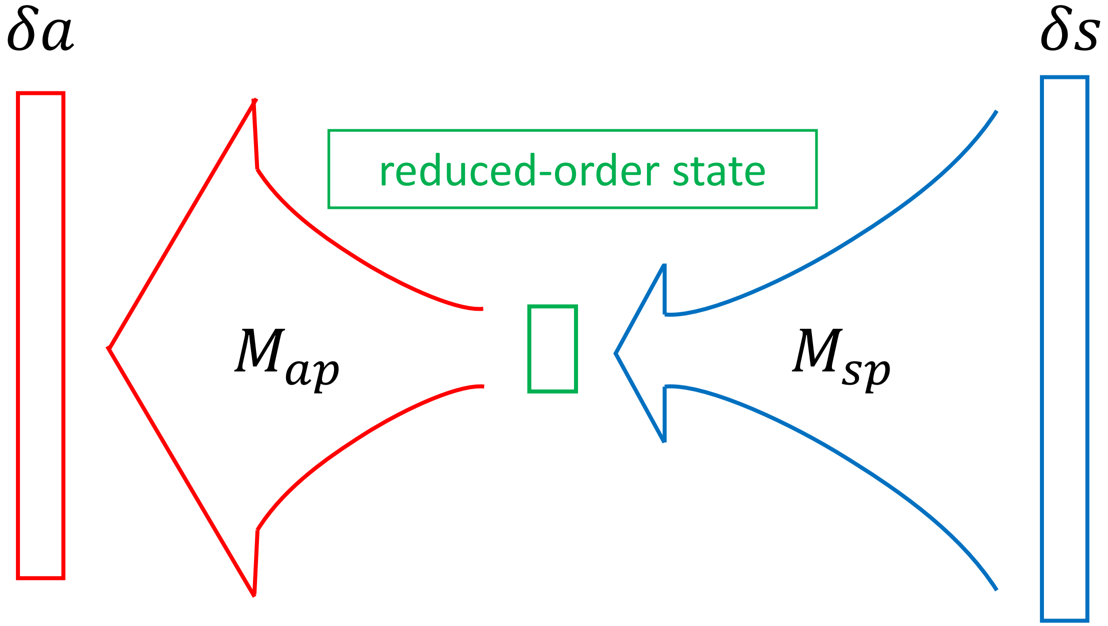
Stage 1b: Reduced-order Closed-loop Policy

$$M_{ap} \cdot M_{sp}$$

change in control

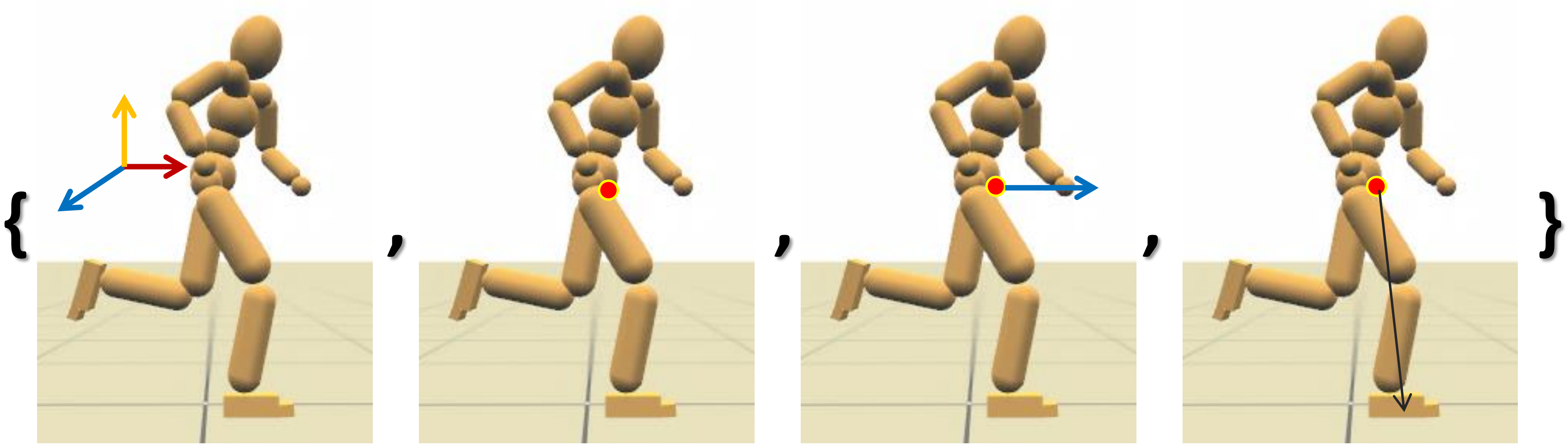
change in states

$$\boxed{\delta a} = \boxed{M}\, \boxed{\delta s} + \hat{a}$$



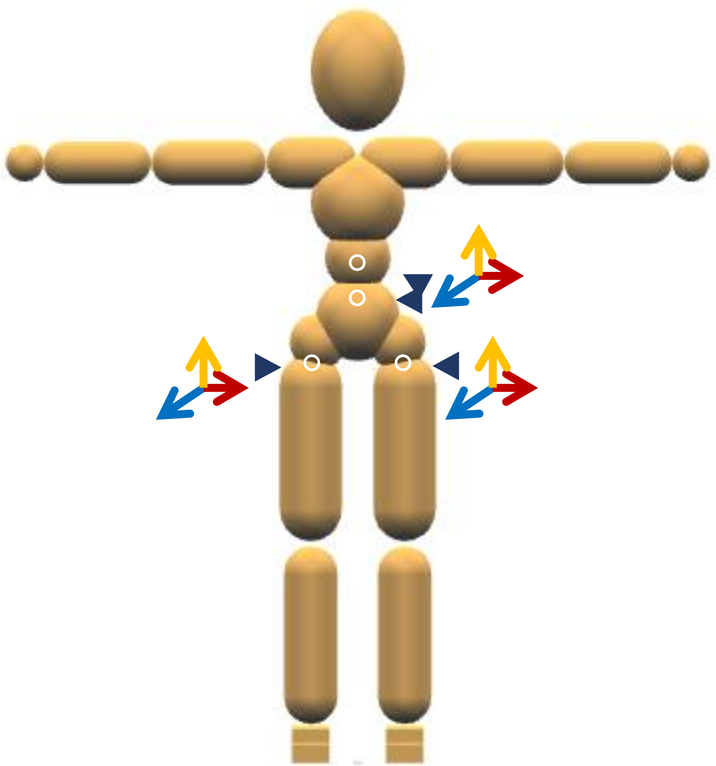
Manually-selected States: s

- Running: 12 dimensions



Manually-selected Controls: a

- for all skills: 9 dimensions

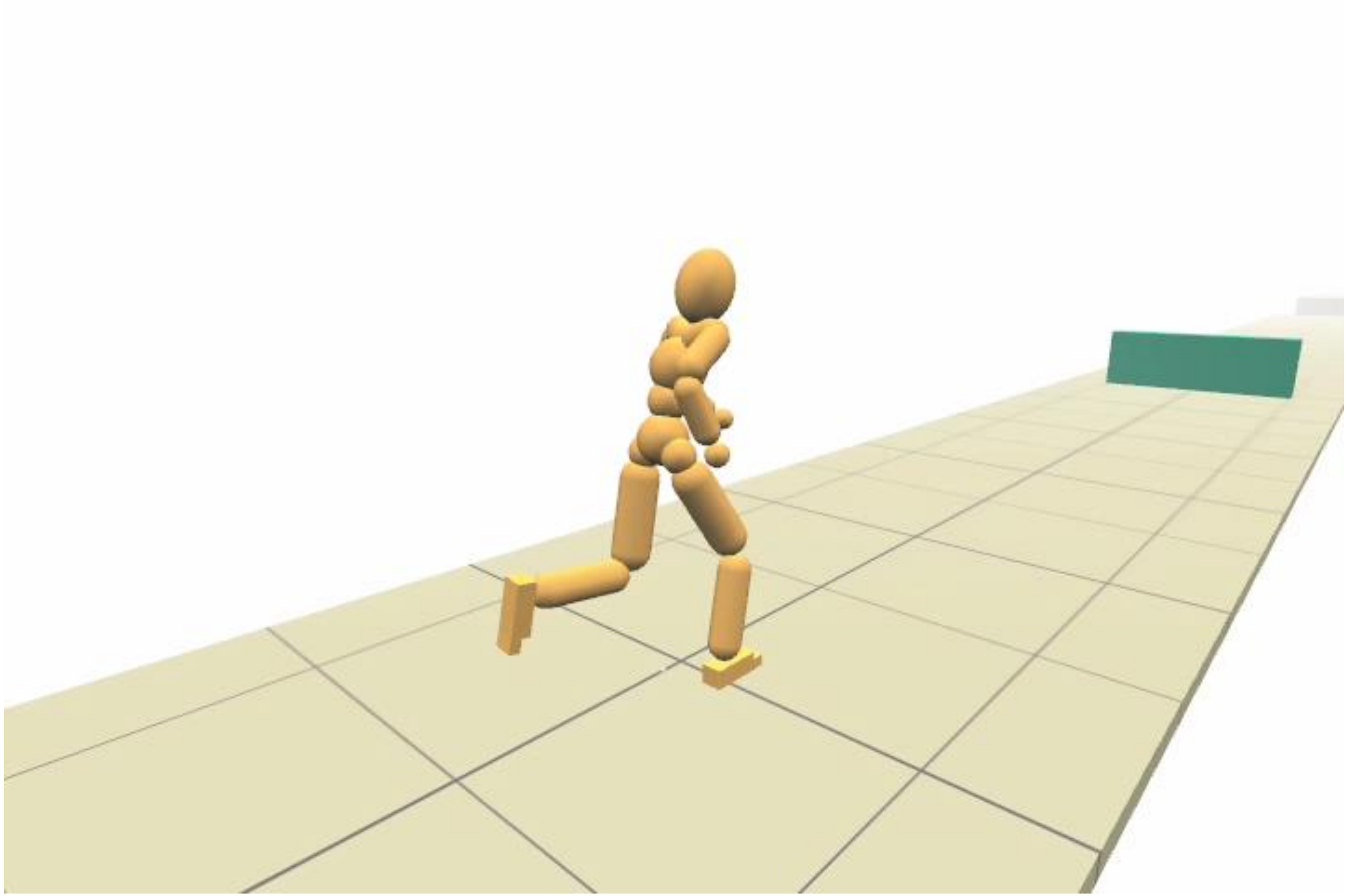


Optimization

$$\delta a = M\delta s + \hat{a}$$

- Optimize M
  - CMA, Covariance Matrix Adaption ([Hansen 2006])
  - For the running task:
    - #optimization variables: 12*9 = 108 / (12*3+3*9) = 63
    - 12 minutes on 24 cores

Example: Locomotion Controller with Linear Policy



[Liu et al. 2012 – Terrain Runner]

Optimal Control ⇔ Reinforcement Learning

- RL shares roughly the same overall goal with Optimal Control

$$\max \sum_{t=0} r(s_t, a_t)$$

Optimal Control ⇔ Reinforcement Learning

- RL shares roughly the same overall goal with Optimal Control

$$\max \sum_{t=0} r(s_t, a_t)$$

- But RL typically does not assume perfect knowledge of system

$$\cancel{f(s_t, a_t) - s_{t+1} = 0}$$

Optimal Control ⇔ Reinforcement Learning

- RL shares roughly the same overall goal with Optimal Control

$$\max \sum_{t=0} r(s_t, a_t)$$

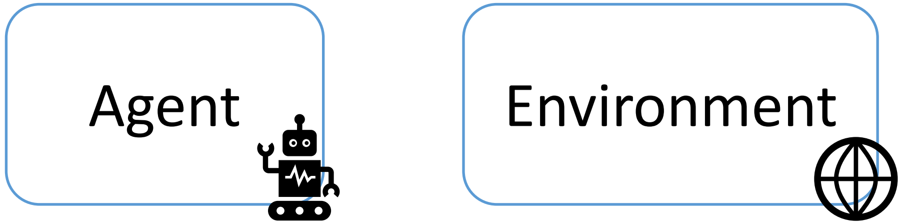
- But RL typically does not assume perfect knowledge of system

$$\cancel{f(s_t, a_t) - s_{t+1} = 0}$$

- RL can still take advantage of a system model → model-based RL
  - The model can be learned from data

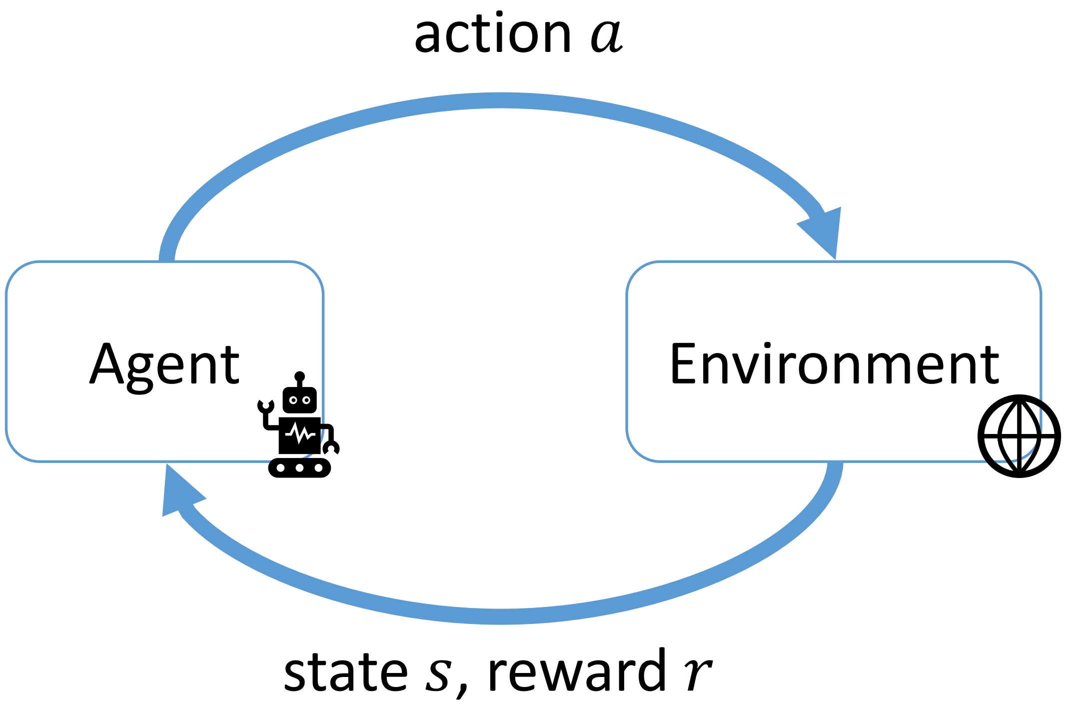
$$s_{t+1} = f(s_t, a_t; \theta)$$

Markov Decision Process (MDP)



Markov Decision Process (MDP)

$$
\begin{tikzcd}
s_0 \arrow[r] \arrow[d] & s_1 \arrow[r] \arrow[d] & s_2 & \cdots \\
a_0 \arrow[ur] & a_1 \arrow[ur] & &
\end{tikzcd}
$$



State s_t Action a_t

Policy $a_t \sim \pi(\cdot\,|s_t)$

Transition probability

$$s_{t+1} \sim p(\,\cdot\,|s_t, a_t)$$

Reward $r_t = r(s_t, a_t)$

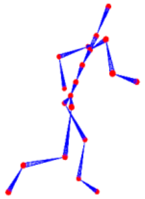
Return $R = \sum_t \gamma^t r(s_t, a_t)$

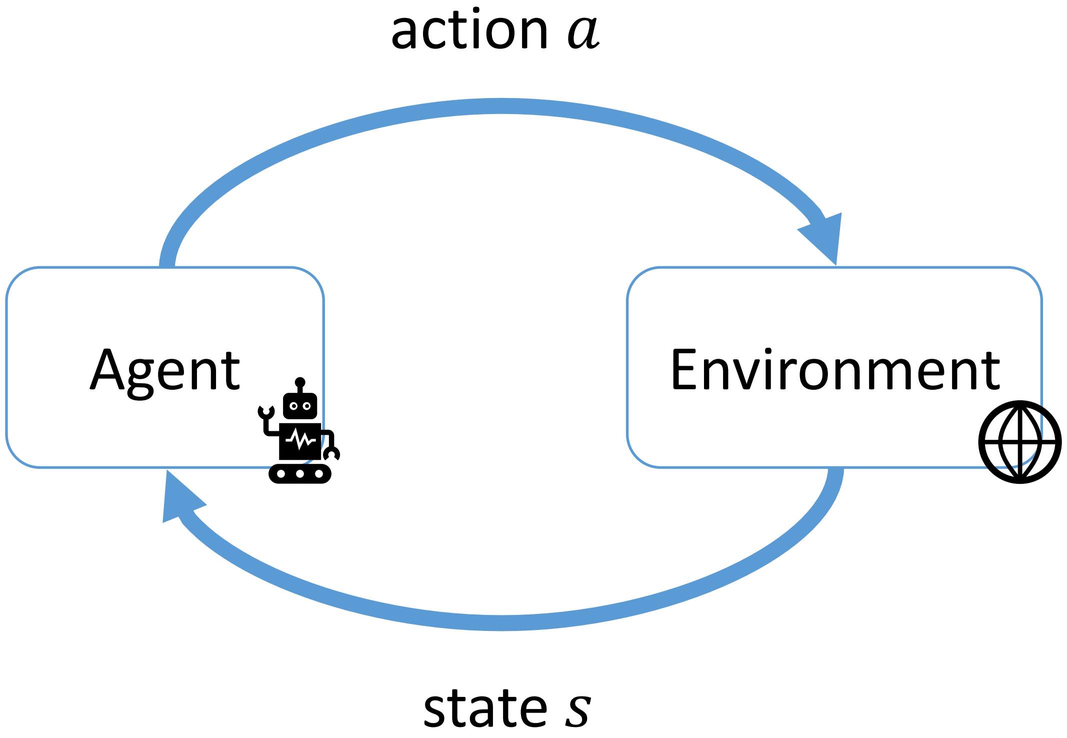
Markov Decision Process (MDP)

Trajectory

$$\tau = s_0 \quad a_0 \quad s_1 \quad a_1 \; s_2 \; \ldots$$

Reward

$r(s_t, a_t) = \|$  $-$  $\| + \cdots$



.

Markov Decision Process (MDP)

MDP is a discrete-time stochastic control process.
It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

State s_t Action a_t

Policy $a_t \sim \pi(\cdot|s_t)$

Transition probability

$$s_{t+1} \sim p(\cdot|s_t, a_t)$$

Reward $r_t = r(s_t, a_t)$

Return $R = \sum_t \gamma^t r(s_t, a_t)$

Markov Decision Process (MDP)

MDP is a discrete-time stochastic control process.
It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

A MDP problem:

$$\mathcal{M} = \{S, A, p, r\}$$

S: state space

A: action space

State $s_t \in S$ Action $a_t \in A$

Policy $a_t \sim \pi(\cdot\,|s_t)$

Transition probability

$$s_{t+1} \sim p(\,\cdot\,|s_t, a_t)$$

Reward $r_t = r(s_t, a_t)$

Return $R = \sum_t \gamma^t r(s_t, a_t)$

Markov Decision Process (MDP)

MDP is a discrete-time stochastic control process.
It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

A MDP problem:

$$\mathcal{M} = \{S, A, p, r\}$$

Solve for a policy $\pi(a|s)$ that optimize the expected return

$$J = E[R] = E_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t)\right]$$

Overall all trajectories $\tau = \{s_0, a_0, s_1, a_1, \dots\}$ induced by π

State $s_t \in S$ Action $a_t \in A$

Policy $a_t \sim \pi(\cdot|s_t)$

Transition probability

$$s_{t+1} \sim p(\cdot|s_t, a_t)$$

Reward $r_t = r(s_t, a_t)$

Return $R = \sum_t \gamma^t r(s_t, a_t)$

Bellman Equations

In optimal control:

Value function: $$V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$$

Optimal policy: $$\pi(s) = \arg\min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$$

Optimal Q-function / state-action value function: $$Q(s,a) = h(s,a) + V\big(f(s,a)\big)$$

Bellman Equations

In optimal control:

Value function: $V(s) = \min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$

Optimal policy: $\pi(s) = \arg\min_a \left(h(s,a) + V\big(f(s,a)\big)\right)$

Optimal Q-function / state-action value function: $Q(s,a) = h(s,a) + V\big(f(s,a)\big)$

In RL control:

Value function for a policy π: $V^\pi(s) = E_{\tau\sim\pi}[r(s,a) + V(s')]$

This is not necessarily optimal

Q-function for a policy π: $Q^\pi(s,a) = r(s,a) + E_{\tau\sim\pi}[V(s')]$

How to Solve MDP

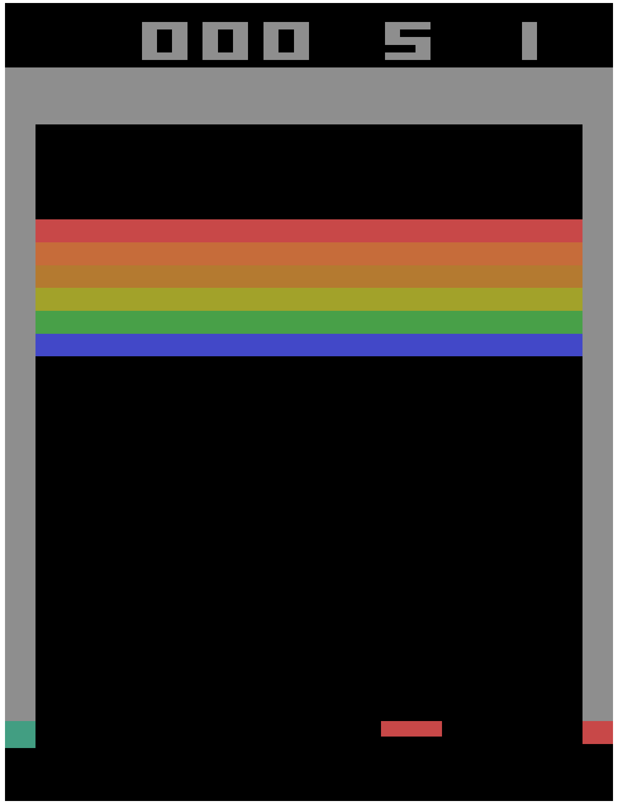
- Value-based Methods
  - Learning the value function/Q-function using the Bellman equations
  - Evaluation the policy as

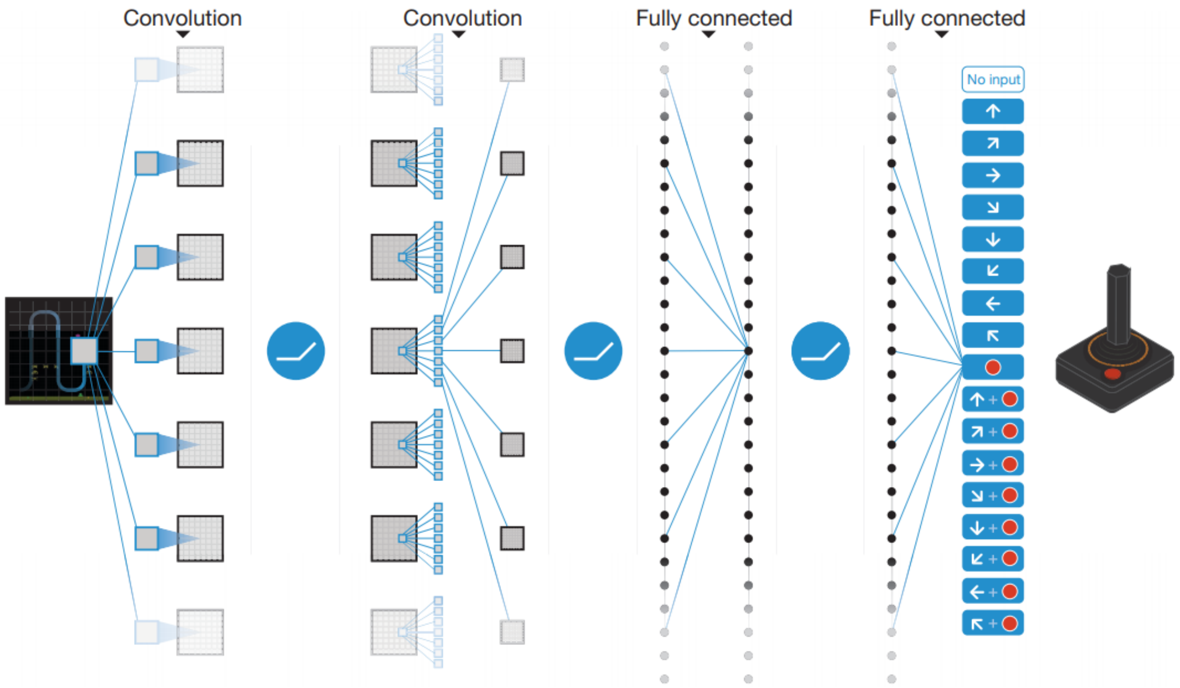
$$\pi(s) = \arg\min_{a} Q(s, a)$$

  - Typically used for discrete problems
  - Example: Value iteration, Q-learning, DQN, …

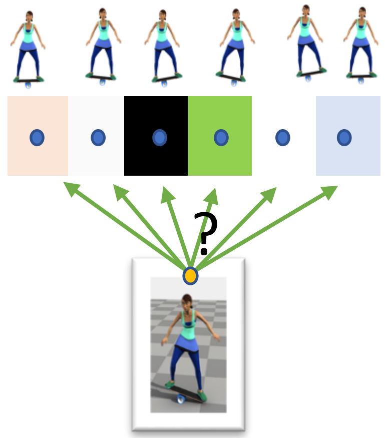
How to Solve MDP

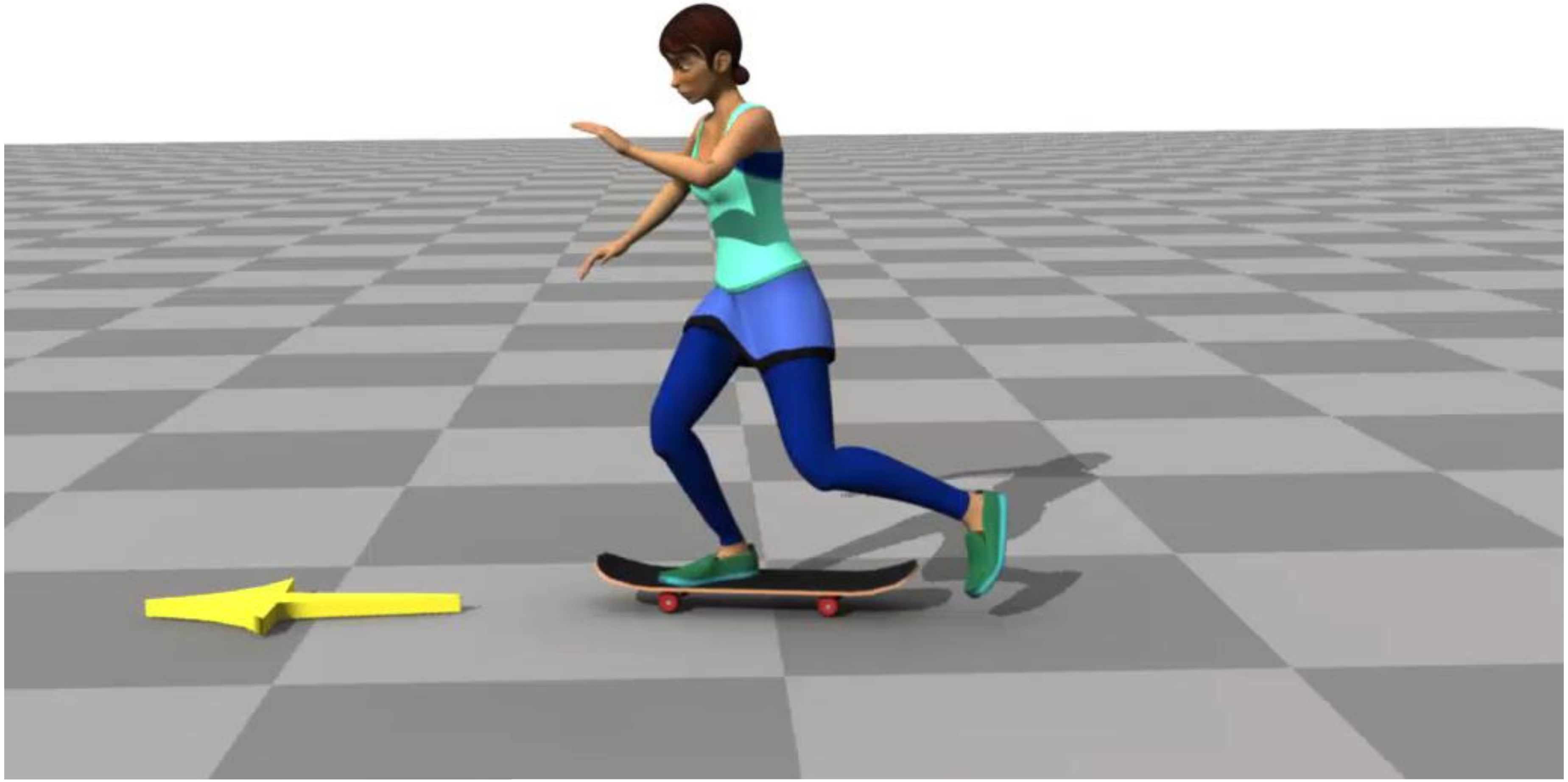
DQN [Mnih et al. 2015, Human-level control through deep reinforcement learning]





Multi-skill Characters





[Liu et al. 2017: Learning to Schedule Control Fragments]

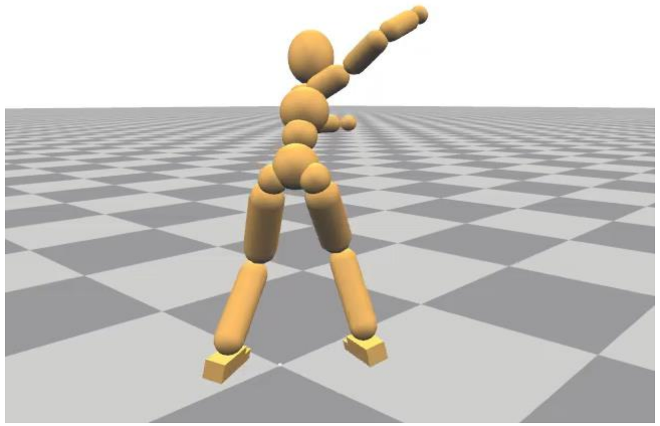
How to Solve MDP

- Policy Gradient approach
  - Learning the value function/Q-function using the Bellman equations
  - Compute approximate policy gradient according to value functions using Monte-Carlo method
  - Update the policy using policy gradient

  - Suitable for continuous problems

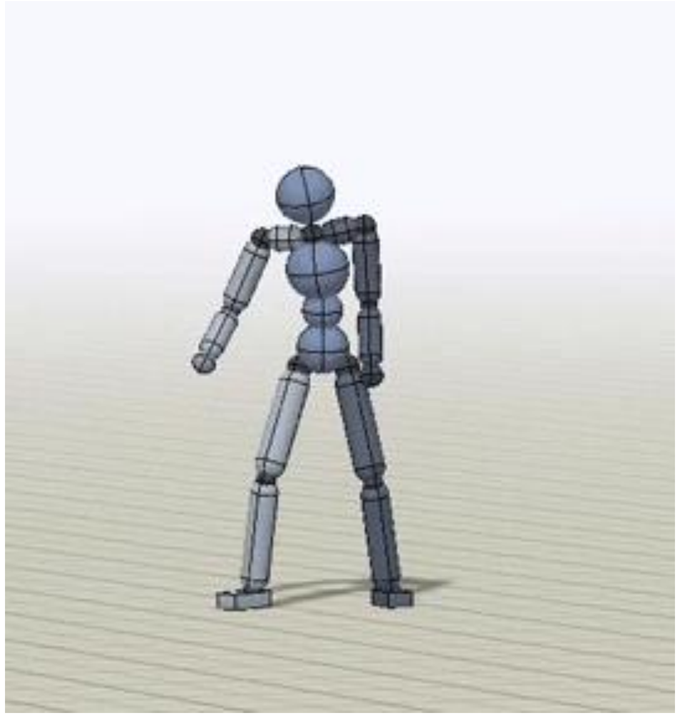
  - Example: REINFORCE, TRPO, PPO, …

How to Solve MDP



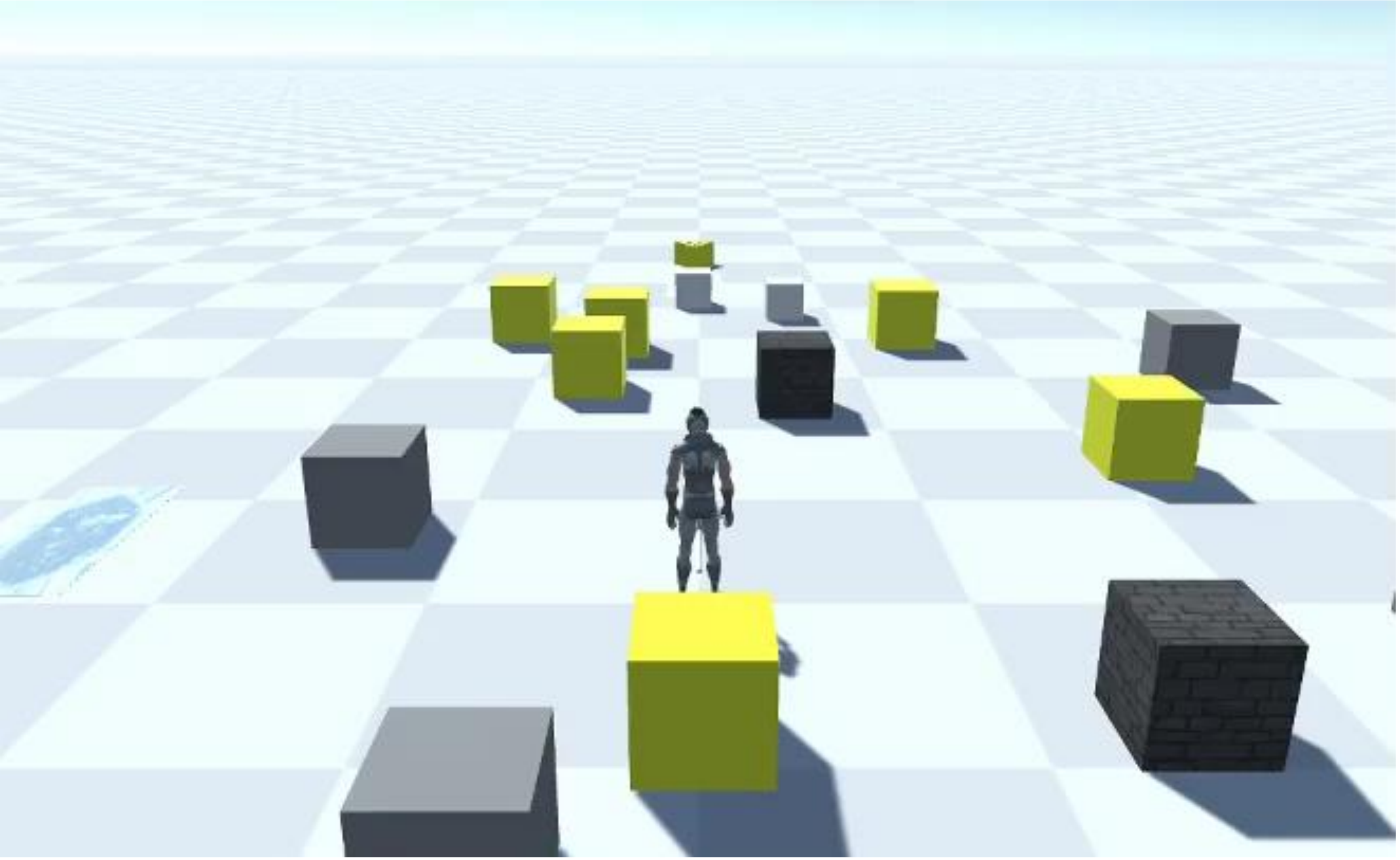
[Liu et al. 2016. ControlGraphs]

[Liu et al. 2018]



[Peng et al. 2018. DeepMimic]

Multi-skill Characters



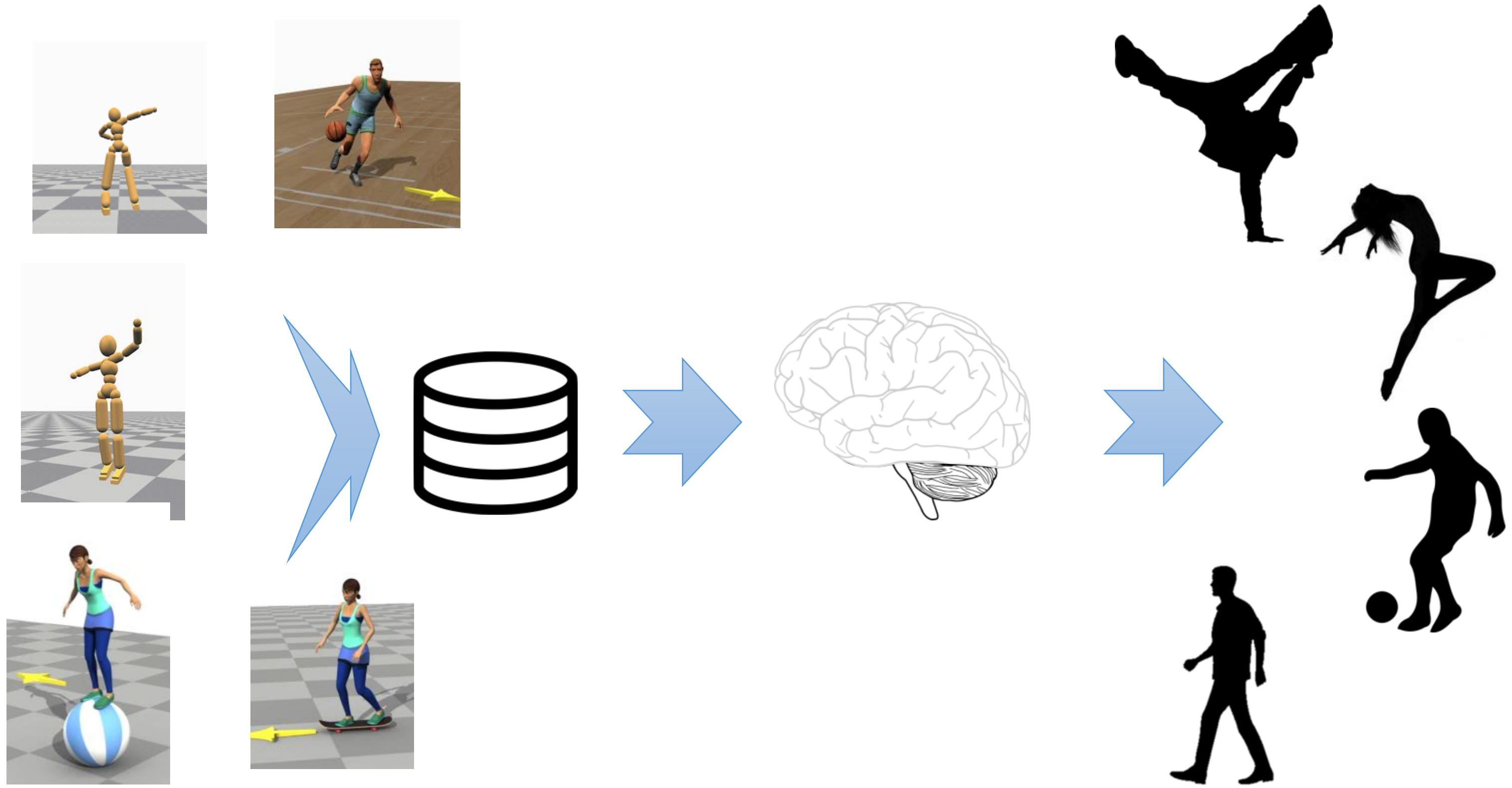
State Machines of
Tracking Controllers

Generative Control Policies

[Yao et al. Control VAE]

What's Next?

- Digital Cerebellum – Large Pretrained Model for Motion Control



What’s Next?

- Cross-modality Generation
  - ⇔ LLM ⇔ Text/Audio ⇔ Motion/Control ⇔ Image/Video ⇔
  - Digital Actor?

Hello, ChatGPT. I want you to act as a public speaking coach.

I will provide you with a speech transcript. Then, you need to provide detailed suggestions about gesture style in a parenthetical after each sentence...

The speech transcript is “I’m glad to come here. We are brave enough to face all challenges. ”

“I’m glad to come here (stand tall and relaxed with open posture). We are brave enough to face all challenges (stand confidently with feet shoulder-width apart and hands on hips or in fists).”

What's Next

That’s all for GAMES 105
Thank you!

aban·don [əˈband(ə)n]

adj. 常看常新