

A control-theoretical perspective of continuous time reinforcement learning

Xin Guo *UC Berkeley*

email: xinguo@berkeley.edu

SURE-AI@ Star Midsummer on Reinforcement Learning

June 23-25, 2026, Oslo

Mini-course outline

- ▶ Introduction to Q and advantage functions in RL
- ▶ DPP & martingale characterization for Q and advantage rate functions in continuous-time RL
- ▶ DPP and martingale characterization for MFC with learning
- ▶ Regret analysis in RL via stability of HJB and BSDE
- ▶ Connecting RL, LLM, and generative diffusion models via PDEs and RDEs

Part I: Q function and advantage functions in RL

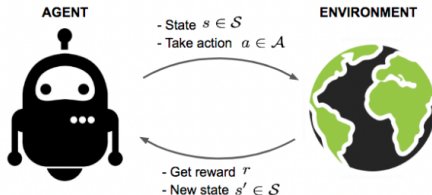


Figure: Illustration of reinforcement learning

- Learning the (stochastic) environment while trying to find optimal control to achieve certain objectives
- Examples: Chess and Go, video games, autonomous driving, robots
- Analytically, RL $\sim$ MDP/control plus learning

RL in continuous-time

For a given control process $(a_t)_{t \geq 0}$ taking values in $A \subset \mathbb{R}^k$, consider the state dynamics governed by the SDE:

$$dX_t = b(t, X_t, a_t)dt + \sigma(t, X_t, a_t)dW_t, \quad \forall t \geq 0.$$

Objective: determine the optimal control that maximizes the total reward:

$$\mathbb{E} \left[\int_0^T r(t, X_t, a_t) dt + g(X_T) \right],$$

without knowing b, σ, r, g .

Admissible controls

- ▶ Open loop: traffic light
- ▶ (*) Closed loop
 - intelligent traffic light
 - most relevant class in RL

Deterministic policy for classical control

Classical control

- ▶ b, σ, r, g are known
- ▶ the optimal control is characterized, under mild conditions, by

$$a_t^* = \mu^*(t, X_t^{\mu^*}), \quad \forall t \geq 0,$$

where

- ▶ $\mu^* : [0, T] \times \mathbb{R}^d \rightarrow A$ is an (optimal) deterministic policy (a.k.a. feedback control)
- ▶ μ^* is continuous/differentiable with proper regularity conditions
- ▶ Dynamic programming principle and HJB equations
- ▶ Martingale characterization and verification theorem approach
- ▶ Stochastic maximum principle and BSDE approach

Reinforcement learning (RL)

RL: try to learn this optimal policy **without knowing** the coefficients, and/or the reward functions

Key concepts: Q function and advantage function

Q function in Markov decision process: discret-time

- A tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, s_0)$

$$V(s) = \text{maximize}_{\pi} \quad \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s \right]$$

subject to $s_{t+1} \sim P(s_t, a_t), \quad a_t \sim \pi(s_t), \quad t = 0, 1, \dots$

- ▶ $s_t \in \mathcal{S}$ the state of the agent at time t
- ▶ $a_t \in \mathcal{A}$ the action of the agent at time t
- ▶ $a_t \sim \pi(s_t)$ where $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ is the policy of the agent
- ▶ Instantaneous reward $R(s_t, a_t)$ sampled from some distribution
- ▶ Time-homogeneous Markov transition kernel $P(s_t, a_t) \in \mathcal{P}(\mathcal{S})$

- Reinforcement learning in discrete time
 - ▶ MDP with *unknown* P and r
 - ▶ Policy
 - ▶ $\pi : \mathcal{S} \rightarrow \mathcal{A}$ pure strategy $\leftrightarrow$ strict control
 - ▶ $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ mixed strategy $\leftrightarrow$ relaxed control (to find optimal decision under **exploration-exploitation** context)

Q Function and Bellman Equation

- ▶ Q function, one of the basic quantities used for RL

$$Q(s, a) : = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right]$$

- ▶ The value function (of a given policy π)

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} [Q^{\pi}(s, a)]$$

- ▶ Advantage function: quantifying the quality of a specific action a over average actions given a specific state s

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

Bellman Equation for Q Function

- ▶ The Bellman equation for the Q -function given by

$$Q^\pi(s, a) = \mathbb{E}[R(s, a)] + \gamma \mathbb{E}_{s' \sim P(s, a), a' \sim \pi(s')} [Q^\pi(s', a')]$$

- ▶ $V(s) = \max_{a \in \mathcal{A}} Q(s, a)$, then $\pi^*(s) \in \arg \max_{a \in \mathcal{A}} Q(s, a)$

Q-function update for $T = \infty$

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \beta_t(s_t, a_t)) \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \beta_t(s_t, a_t) \underbrace{\left[R(s_t, a_t) + \gamma \max_{a'} Q_t(s_{t+1}, a') \right]}_{\text{new value}}$$

- ▶ β_t : learning rate
- ▶ $n^i(s, a)$: i -th time to visit (s, a)
- ▶ (Watkins and Dayan 1992),

$$\sum_{i=1}^{\infty} \beta_{n^i(s,a)} = \infty, \quad \sum_{i=1}^{\infty} (\beta_{n^i(s,a)})^2 < \infty,$$

then $Q_t(s, a) \rightarrow Q(s, a)$ as $t \rightarrow \infty$, $\forall s, a$ with probability 1

Outline

Q function in RL

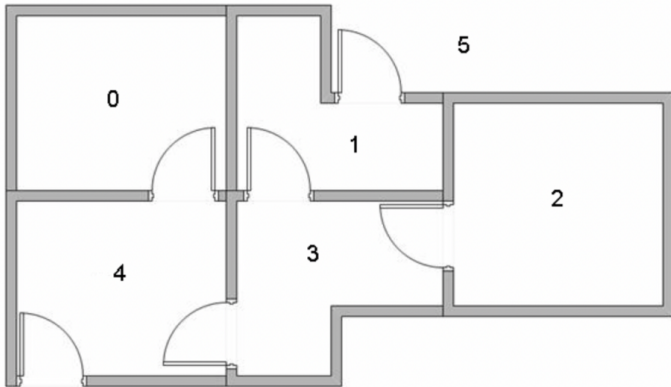
- Q function

- Q function: illustration example

Characterization of Q in continuous-time RL

Example of Q-function

Consider this house. You start in 2 and you want to go outside. Quickest path? (Ok, not too hard here, but imagine 100 rooms...)



Example

Note: problem and pictures (with minor modifications) from <http://mnemstudio.org/path-finding-q-learning-tutorial.htm>

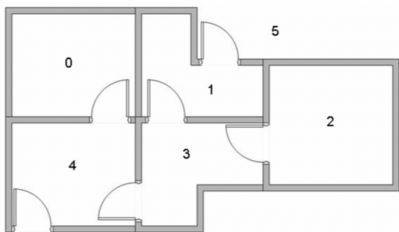
Let us solve the problem by the Q learning approach. First, let us formalize the model

- ▶ State space $\mathcal{S} = \{0, 1, 2, 3, 4, 5\}$. That is, $s_t = 5$ means that at time t , we are in room number 5
- ▶ Action space $\mathcal{A} = \{0, 1, 2, 3, 4, 5\}$. That is, $a_t = 5$ means that at time t , our action is move to room number 5
- ▶ Transition function $s_{t+1} = P(s_t, a_t) = a_t$. Note that no noise here, deterministic system
- ▶ Reward function $R = ?$

Example

Recall: $R(s, a)$ is the reward if you are in state s and choose action a (that is, you are in room s and move to room a). We choose

- $R(s, a) = 0$ if (s, a) are linked (*alternative* = -1)
- $R(s, a) = -1$ if (s, a) are not linked (*alternative* = $-\infty$)
- $R(s, a) = 100$ if a is the target state



State	Action					
	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Example

Algorithm.

1. Initialize the matrix Q with all zeros.
- 2a. Randomly choose the initial state $s \in S$.
- 2b. Randomly choose an (admissible) action $a \in A$ and use approximated dynamic programming to update $Q(s, a)$. You are now in state $s = a$.
- 2c. Repeat (2a)-(2b) until you touch the target state.
3. Repeat (2a)-(2b)-(2c) for N iterations.
4. Compute the optimal control.

Example

Recall the equation to update the matrix Q :

$$Q(s, a) \leftarrow (1 - \eta)Q(s, a) + \eta \left(R(s, a) + \gamma \max_{a' \in A} Q(f(s, a, \bar{e}_t), a') \right).$$

To simplify, we here take $\eta = 1$ (all the focus on the present iteration), and $\gamma = 0.8$. The update then reads

$$Q(s, a) \leftarrow R(s, a) + 0.8 \max_{a' \in A, a' \text{ admissible}} Q(a, a').$$

Let's see the details...

Example

We first initialize the matrix Q to zero.

$$R = \begin{array}{c|cccccc} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Example

Randomly choose an initial state, say 1. The admissible actions are $\{3, 5\}$. Randomly choose one, say 5. Then we update $Q(1, 5)$ as

$$\begin{aligned} Q^{new}(1, 5) &= R(1, 5) + 0.8 \max_{a' \in A} Q(5, a') \\ &= R(1, 5) + 0.8 \max\{Q(5, 1), Q(5, 4), Q(5, 5)\} \\ &= 100 + 0.8 \max\{0, 0, 0\} = 100. \end{aligned}$$

State	Action					
	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Example

We are now in state 5. Since this is the target state, we have finished this episode.

We now run a second episode, starting from another a randomly chosen initial state...

Example

Randomly choose an initial state, say 3. The admissible actions are $\{1, 2, 4\}$. Randomly choose one, say 1. We update $Q(3, 1)$ as

$$\begin{aligned}Q^{new}(3, 1) &= R(3, 1) + 0.8 \max_{a' \in A} Q(1, a') \\&= R(3, 1) + 0.8 \max\{Q(1, 3), Q(1, 5)\} \\&= 0 + 0.8 \max\{0, 100\} = 80.\end{aligned}$$

State	Action					
	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Example

We are now in state 1, not the target state, so we go on. The admissible actions are $\{3, 5\}$. Randomly choose one, say 5. We update $Q(1, 5)$ as

$$\begin{aligned} Q^{new}(1, 5) &= R(1, 5) + 0.8 \max_{a' \in A} Q(1, a') \\ &= R(1, 5) + 0.8 \max\{Q(5, 1), Q(5, 3), Q(5, 5)\} \\ &= 100 + 0.8 \max\{0, 0, 0\} = 100. \quad (\text{no update}) \end{aligned}$$

		Action					
State		0	1	2	3	4	5
0	$R =$	-1	-1	-1	-1	0	-1
1		-1	-1	-1	0	-1	100
2		-1	-1	-1	0	-1	-1
3		-1	0	0	-1	0	-1
4		0	-1	-1	0	-1	100
5		-1	0	-1	-1	0	100

$$Q = \begin{matrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Example

We are now in state 5. Since this is the target state, we have finished this episodes.

We repeat several times...

After several episodes, we get

$$R = \begin{array}{c|cccccc} & \text{Action} \\ \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 400 & 0 \\ 1 & 0 & 0 & 0 & 320 & 0 & 500 \\ 2 & 0 & 0 & 0 & 320 & 0 & 0 \\ 3 & 0 & 400 & 256 & 0 & 400 & 0 \\ 4 & 320 & 0 & 0 & 320 & 0 & 500 \\ 5 & 0 & 400 & 0 & 0 & 400 & 500 \end{array}$$

Example

Recall that the optimal control is (s the initial state):

$$a_0^* = \arg \max_{a \in A} Q(s, a), \quad a_1^* = \arg \max_{a \in A} Q(a_0^*, a), \quad \dots$$

Example

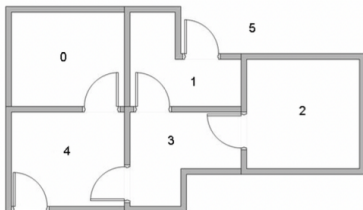
Recall that the optimal control is (s the initial state):

$$a_0^* = \arg \max_{a \in A} Q(s, a), \quad a_1^* = \arg \max_{a \in A} Q(a_0^*, a), \quad \dots$$

Example: if you start in 0, then $0 \rightarrow 4 \rightarrow 5$.

Example: if you start in 2, then either $2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ or $2 \rightarrow 3 \rightarrow 1 \rightarrow 5$.

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{matrix}$$



Part II: Q and advantage rate functions in continuous-time RL and MFC

Recall: RL in continuous-time

For a given control process $(a_t)_{t \geq 0}$ taking values in $A \subset \mathbb{R}^k$, consider the state dynamics governed by the SDE:

$$dX_t = b(t, X_t, a_t)dt + \sigma(t, X_t, a_t)dW_t, \quad \forall t \geq 0.$$

Objective: determine the optimal control that maximizes the total reward:

$$\mathbb{E} \left[\int_0^T r(t, X_t, a_t) dt + g(X_T) \right],$$

without knowing b, σ, r, g .

Recall: deterministic policy for classical control

Classical control

- ▶ b, σ, r, g are known
- ▶ the optimal control is characterized, under mild conditions, by

$$a_t^* = \mu^*(t, X_t^{\mu^*}), \quad \forall t \geq 0,$$

where

- ▶ $\mu^* : [0, T] \times \mathbb{R}^d \rightarrow A$ is an (optimal) deterministic policy (a.k.a. feedback control)
- ▶ μ^* is continuous/differentiable with proper regularity conditions

RL: try to learn this optimal policy **without knowing** the coefficients, and/or the reward functions

Two key quantities in continuous time RL

- ▶ Q-function?
- ▶ Advantage function?

First approach: stochastic policy with entropy regularization

Exploratory formulation [Wang, Zariphopoulou and Zhou (2020)]

Consider **stochastic policies** $\pi : [0, T] \times \mathbb{R}^d \rightarrow \mathcal{P}(A)$, and the corresponding **exploratory** state dynamics:

$$dX_t^\pi = \tilde{b}^\pi(t, X_t^\pi)dt + \tilde{\sigma}^\pi(t, X_t^\pi)dW_t,$$

where

$$\tilde{b}^\pi(t, x) = \int_A b(t, x, a)\pi(da|t, x), \quad \tilde{\sigma}^\pi(t, x) = \sqrt{\int_A \sigma^2(t, x, a)\pi(da|t, x)}.$$

Maximize the **entropy-regularized** objective function over π :

$$\mathbb{E} \left[\int_0^T (\tilde{r}^\pi(t, X_t^\pi) - \gamma \text{KL}(\pi(\cdot|t, X_t^\pi))) dt + g(X_T^\pi) \right],$$

where $\tilde{r}^\pi(t, x) = \int_A r(t, x, a)\pi(a|t, x)da$, and $\gamma > 0$ is a given parameter.

Stochastic policy with entropy regularization

- ▶ Stochastic policy and exploratory dynamics capture **exploration**: at each state, sampling actions according to a distribution.

$$dX_t = b(t, X_t, a_t)dt + \sigma(t, X_t, a_t)dW_t, \quad a_t \sim \pi(da|t, X_t).$$

- ▶ Optimal exploratory policy is Gaussian in LQ-RL
- ▶ Based on this framework, many discrete-time RL algorithms are extended to continuous-time, e.g., temporal difference learning [Jia and Zhou (2022a)], q -learning [Jia and Zhou (2022b)], policy gradient and actor-critic learning [Jia and Zhou (2023)].

Issues with stochastic policy approach

- ▶ Inconsistent with classical control framework with deterministic policy
- ▶ Recovering a deterministic policy from a learned stochastic one is challenging (except the LQ setting).
- ▶ Exploratory dynamics requires continuously sampling independent actions at different states, which is infeasible both theoretically and practically [Jia, Ouyang and Zhang (2025)].
- ▶ Frequent sampling makes actions discontinuous over time, in contrast to deterministic policy.

Latest approach: directly characterize a deterministic policy!

Issues with stochastic policy approach

- ▶ Inconsistent with classical control framework with deterministic policy
- ▶ Recovering a deterministic policy from a learned stochastic one is challenging (except the LQ setting).
- ▶ Exploratory dynamics requires continuously sampling independent actions at different states, which is infeasible both theoretically and practically [Jia, Ouyang and Zhang (2025)].
- ▶ Frequent sampling makes actions discontinuous over time, in contrast to deterministic policy.

Latest approach: directly characterize a deterministic policy!

General idea (Cheng, G., Zhang (2025))

Approximate the (optimal) deterministic policy (DPG) in a parametric form $\{\mu_\phi\}_{\phi \in \mathbb{R}^n}$, and optimize the policy parameterization by gradient methods.

Consider maximizing the reward over $\phi \in \mathbb{R}^n$:

$$V^\phi(t, x) := \mathbb{E} \left[\int_t^T r(s, X_s^\phi, \mu_\phi(s, X_s^\phi)) ds + g(X_T^\phi) \mid X_t^\phi = x \right],$$

where

$$dX_s^\phi = b(s, X_s^\phi, \mu_\phi(s, X_s^\phi))ds + \sigma(s, X_s^\phi, \mu_\phi(s, X_s^\phi))dW_s.$$

Deterministic policy gradient in continuous-time RL

Theorem 1 (Cheng, G., Zhang (2025))

Under suitable regularity conditions,

$$\partial_{\phi} V^{\phi}(t, x) = \mathbb{E}^{t, x} \left[\int_t^T \partial_{\phi} \mu_{\phi}(s, X_s^{\phi})^{\top} \partial_a A^{\phi}(s, X_s^{\phi}, \mu_{\phi}(s, X_s^{\phi})) ds \right],$$

where $A^{\phi}(t, x, a) := \mathcal{L}[V^{\phi}](t, x, a) + r(t, x, a)$ is the Hamiltonian, with

$$\begin{aligned} \mathcal{L}[\varphi](t, x, a) &:= \partial_t \varphi(t, x) + b(t, x, a)^{\top} \partial_x \varphi(t, x) \\ &\quad + \frac{1}{2} \text{tr}(\sigma \sigma^{\top}(t, x, a) \partial_{xx}^2 \varphi(t, x)). \end{aligned}$$

- ▶ A^{ϕ} is analogous to the advantage function in discrete-time DPG [Silver et.al (2014)], and the q -function for stochastic policy gradient [Jia and Zhou (2023)].

Limit of discrete-time DPG

$$\partial_{\phi} V^{\phi}(t, x) = \mathbb{E}^{t, x} \left[\int_t^T \partial_{\phi} \mu_{\phi}(s, X_s^{\phi})^{\top} \partial_a A^{\phi}(s, X_s^{\phi}, \mu_{\phi}(s, X_s^{\phi})) ds \right]$$

Let $\beta = 0$, and consider the time discretized objective

$$J_{\Delta t}(\phi) := \mathbb{E} \left[\sum_{i=0}^{N-1} r(t_i, X_{t_i}^{\Delta t, \phi}, \mu_{\phi}(t_i, X_{t_i}^{\Delta t, \phi})) \Delta t + g(X_T^{\Delta t, \phi}) \right],$$

where

$$\begin{aligned} X_{t_{i+1}}^{\Delta t, \phi} &= X_{t_i}^{\Delta t, \phi} + b(t_i, X_{t_i}^{\Delta t, \phi}, \mu_{\phi}(t_i, X_{t_i}^{\Delta t, \phi})) \Delta t \\ &\quad + \sigma(t_i, X_{t_i}^{\Delta t, \phi}, \mu_{\phi}(t_i, X_{t_i}^{\Delta t, \phi})) \sqrt{\Delta t} \omega_{t_i}. \end{aligned}$$

The discrete-time DPG in [Silver et.al (2014)] gives

$$\partial_{\phi} J_{\Delta t}(\phi) = \mathbb{E} \left[\sum_{i=0}^{N-1} \partial_{\phi} \mu_{\phi}(t_i, X_{t_i}^{\Delta t, \phi})^{\top} \partial_a A^{\Delta t, \phi}(t_i, X_{t_i}^{\Delta t, \phi}, \mu_{\phi}(t_i, X_{t_i}^{\Delta t, \phi})) \Delta t \right],$$

where $A^{\Delta t, \phi}(t, x, a) := \frac{Q^{\Delta t, \phi}(t, x, a) - V^{\Delta t, \phi}(t, x)}{\Delta t}$ is the discrete-time advantage rate function.

How to learn A^ϕ ?

$$A^\phi(t, x, a) := \mathcal{L}[V^\phi](t, x, a) + r(t, x, a)$$

Theorem 2 (Cheng, G., Zhang (2025))

Let $\hat{V} \in C^{1,2}([0, T] \times \mathbb{R}^d)$ and $\hat{q} \in C([0, T] \times \mathbb{R}^d \times A)$ satisfy for all $(t, x) \in [0, T] \times \mathbb{R}^d$,

$$\hat{V}(T, x) = g(x), \quad \hat{q}(t, x, \mu_\phi(t, x)) = 0,$$

and there exists a neighborhood $\mathcal{O}_{\mu_\phi(t, x)}$ of $\mu_\phi(t, x)$ s.t. $\forall a \in \mathcal{O}_{\mu_\phi(t, x)}$,

$$\left(\hat{V}(s, X_s^{t, x, a}) + \int_t^s (r - \hat{q})(u, X_u^{t, x, a}, \alpha_u) du \right)_{s \in [t, T]} \text{ is a martingale,}$$

where

$$dX_s^{t, x, a} = b(s, X_s^{t, x, a}, \alpha_s) ds + \sigma(s, X_s^{t, x, a}, \alpha_s) dW_s, \quad X_t^{t, x, a} = x,$$

and $(\alpha_s)_{s \geq t}$ is a control process with $\lim_{s \searrow t} \alpha_s = a$.

Then $\hat{V} = V^\phi$, and $\hat{q} = A^\phi$ in the neighborhood of μ_ϕ .

Key insight from deterministic policy

- ▶ Explore the neighborhood of current actions, rather than the entire action space
- ▶ Use data from the original state, rather than the exploratory state
- ▶ The Bellman condition only evaluates at the current action, while the stochastic policy counterpart requires

$$\int_A (\hat{q}(t, x, a) - \gamma \log \pi(a|t, x)) \pi(da|t, x) = 0,$$

which needs an additional Monte Carlo step to approximate the integral.

Policy evaluation

Let V_θ, q_ψ be the parameterized value and advantage function.
For martingale condition, we utilize semi-gradient Temporal Difference (TD) update:

$$\Delta := V_\theta(t, x_t) - \int_t^{t+\delta} e^{-\beta(s-t)} [r - q_\psi](s, x_s, a_s) ds - e^{-\beta\delta} V_\theta(t + \delta, x_{t+\delta})$$

$$\theta \leftarrow \theta - \eta \partial_\theta V_\theta(t, x_t) \cdot \Delta$$

$$\psi \leftarrow \psi - \eta \partial_\psi q_\psi(t, x_t, a_t) \cdot \Delta$$