Enhancing Value Function Estimation through First-Order State-Action Dynamics in Offline Reinforcement Learning

Anonymous Authors[1]

Abstract

In offline reinforcement learning (RL), updating the value function with the discrete-time Bellman Equation often encounters challenges due to the limited scope of available data. This limitation stems from the Bellman Equation, which cannot accurately predict the value of unvisited states. To address this issue, we have introduced an innovative solution that bridges the continuous- and discrete-time RL methods, capitalizing on their advantages. Our method uses a discrete-time RL algorithm to derive the value function from a dataset while ensuring that the function's first derivative aligns with the local characteristics of states and actions, as defined by the Hamilton-Jacobi-Bellman equation in continuous RL. We provide practical algorithms for both deterministic policy gradient methods and stochastic policy gradient methods. Experiments on the D4RL dataset show that incorporating the first-order information significantly improves policy performance for offline RL problems.

1. Introduction

In discrete-time reinforcement learning (RL), interactions between the agent and the environment are discretized into steps, even in scenarios that are inherently continuous (Mnih et al., 2015; Silver et al., 2017). This approach updates the value function by applying the discrete-time form of the Bellman Equation (Haarnoja et al., 2018; Silver et al., 2014; Watkins & Dayan, 1992). While it works well in online settings, its limitations become evident in offline applications (Kostrikov et al., 2021; Wu et al., 2020; Kumar et al., 2019; Fujimoto & Gu, 2021; Wang et al., 2020). The main issue is that, in training agents, discrete-time RL relies solely on pre-collected data, which might not capture the entire scenario comprehensively.

Compared to the online setting, offline RL has to extrapolate the value function from the dataset, so generalization becomes a problem (Parag et al., 2022; Xu & Gu, 2020; Boyan & Moore, 1994). We show an example in Figure 1, where an agent is tasked to reach a target located at the bottom-left corner. However, due to inaccurate value function estimations of unvisited states, such as the wall area in this example, the learned value function may erroneously guide policies towards the left starting from the bottom-right corner, as shown in panel (b). We posit that incorporating additional derivative information of the value function with respect to states and actions (Dong et al., 2021; Vemula et al., 2019; Czarnecki et al., 2017) can aid in addressing the extrapolation issues, as depicted in Figure 1(c).

Continuous-time RL incorporates the derivatives of the value function. Unlike discrete-time methods, the value function in continuous-time RL is characterized by first- or second-order partial differential equations, known as the Hamilton-Jacobi-Bellman (HJB) Equation (Doya, 2000; Munos, 2000). The optimal policy is then derived by considering the given reward function and system dynamics through the HJB equation. However, the direct application of the HJB equation faces challenges. These methods are hindered by the curse of dimensionality when solving the HJB equation (Lutter et al., 2020; Shilova et al., 2023), and they require explicit knowledge of the reward and system dynamics.

Inspired by continuous-time RL, in this paper, we introduce a novel objective function that assesses the first-order consistency between the learned value function and the first-order properties of the HJB equation as induced by the dataset. This approach effectively eliminates the need for explicit knowledge of both the reward function and system dynamics. We then demonstrate how this loss function can be integrated with both deterministic (Fujimoto et al., 2018; Silver et al., 2014) and stochastic policy gradient models (Haarnoja et al., 2018). We conducted experiments on the D4RL benchmark dataset (Fu et al., 2020) and compared our method with current leading offline RL methods. The results confirmed significant improvements in policy performance when higher-order information from data is in-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

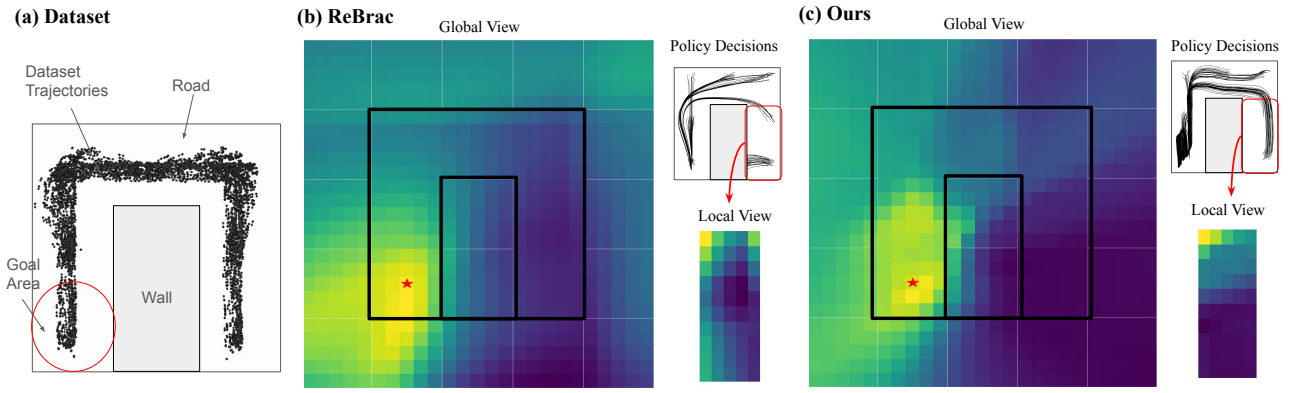

Figure 1. (a) In this 2D maze environment, we depict the path in white and the wall in gray. The goal is for the agents to reach the bottom-left corner from various starting positions. The agents can only receive rewards when they are within a 0.5-unit radius of the goal state circle. Each segment in the dataset represents a pair of (s, a, s'), where s, a, and s' represent the current state, action, and next state, respectively. (b) ReBrac's (Tarasov et al., 2023) value function landscape is visualized using a color map where the colors range from dark blue to green and yellow, indicating low and high values, respectively. The value function landscape of the bottom right region (i.e., local view) is highlighted with a re-normalized color mapping so that the gradients of the value function can be observed. It is evident that the increased value in the bottom-right side of the environment guides the policy to move towards the left and collide with the wall. The resulting navigation trajectories are shown on the top right. (c) Our system's value function is more accurate and can guide the policy to reach the goal state, behaving similarly to the collected data.

corporated. Specifically, our method achieved normalized score improvements over state-of-the-art methods by 4%, 11.4%, 6.1%, and 22.9% in the Mujoco, Antmaze, Adroit, and Franka Kitchen environments, respectively. We will make our code publicly available.

2. Related Work

2.1. Offline Reinforcement Learning

In Offline RL, policies are trained exclusively on pre-collected datasets, without interacting with the environment.

A popular approach is the policy constraint methods, which align the learned policy with the behavior policy inferred from the dataset. Policy constraint methods can be categorized into two primary types: direct and implicit. Direct methods (Kostrikov et al., 2021; Wu et al., 2020; Fujimoto et al., 2019) explicitly model the behavior policy, whereas implicit methods (Kumar et al., 2019; Fujimoto & Gu, 2021; Wang et al., 2020) do not rely on a specific model representation of the behavior policy. Another approach in offline RL involves importance sampling (Nachum et al., 2019; Zhang et al., 2020). These methods adjust the state-action distribution within the offline dataset, offering a re-weighted perspective of the state-action density distribution. Regularization methods (Kumar et al., 2020; Yu et al., 2021; Singh et al., 2020), on the other hand, introduce penalty terms for those state-action pairs not represented in the dataset. Finally, uncertainty-based methods (An et al., 2021; Agarwal et al., 2020) mitigate the uncertainties inherent in offline data, by balancing between conservative RL strategies and off-policy techniques, and leveraging the ensemble models to determine the approach's conservativeness.

Our model distinguishes itself by exploiting first-order information in the dataset, in contrast to the predominant focus on output value information of the Bellman equation by the aforementioned strategies.

2.2. Continuous-time Reinforcement Learning

Continuous-time RL sets itself apart from discrete RL by adopting a continuous temporal formulation. This approach was initially explored in studies by Munos (1997); Munos & Bourgine (1997) and further elaborated in works by Krylov (2008) and Fleming & Soner (2006). Continuous-time RL centers on the principle that the expected return, or value function, conforms to a specific partial differential equation (PDE) known as the Hamilton-Jacobi-Bellman (HJB) equation. Unlike discrete-time RL methods that rely on the output value of the Bellman equation, continuous-time RL employs a higher-order function approximation to determine the value function. The HJB equation allows for the use of various numerical methods to address it (Doya, 2000; Munos, 2000). These methods enable the derivation of policies aimed at maximizing expected returns, which are not only optimal in terms of effectiveness but also avoid the inaccuracies and computational challenges often encountered with time discretization.

The application of Hamilton-Jacobi-Bellman (HJB) equations has recently attracted significant attention in both RL and control theory. For instance, Kim et al. (2021) proposed

a novel HJB formulation tailored for Q Networks, where the control action is assumed to remain Lipschitz continuous over time. This method has been shown to outperform deep deterministic policy gradient (DDPG) methods without requiring an actor network. Similarly, Wiltzer et al. (2022) introduced a distributional HJB equation, laying the groundwork for the online Wasserstein gradient flow fitted Q-iteration algorithm (Martin et al., 2020). This algorithm more precisely models return distributions than quantile regression temporal difference learning (Dabney et al., 2018), especially in particle-control tasks. They offer a practical solution for approximating the distributional HJB equation with the method presented by Jordan et al. (1998), making it suitable for online control algorithms.

We, unlike the conventional approaches that directly solve the HJB equation, leverage the HJB equation to analyze the first-order behavior of the value function. Our strategy enhances the accuracy of the value function while avoiding the high computational costs associated with solving PDEs.

3. Preliminaries

Reinforcement Learning

A Markov Decision Process (MDP) is characterized by a set of components, denoted as $M = (S, A, P_0, r, \rho_0, \gamma)$. Here, S and A represent the state and action spaces, the reward function $r(s, a)$ with state s and action a has range $[-r_{\max}, r_{\max}]$, $P_0(s'|s, a)$ is the transition function, ρ_0 denotes the initial state distribution, and $\gamma \in (0, 1)$ stands for the discount factor. We focus on Markovian policies, $\pi \in \Pi$, which are mappings from states to distributions over actions. The value function, $V^\pi(s) = \mathbb{E}_{a_t \sim \pi, s_t \sim P_0} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, represents the anticipated discounted return, considering policies starting from an initial state distribution, can be expressed as $J_{\rho_0}(\pi, P_0) = \sum_{s \in S} \rho_0(s) V^\pi(s)$. Additionally, the state-action value function is defined as $Q^\pi(s, a) = \mathbb{E}_{a_t \sim \pi, s_t \sim P_0} \left[r(s, a) + \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)\right]$.

Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) is a state-of-the-art off-policy actor-critic algorithm. In TD3, the critic parameterized by ϕ minimizes the Bellman error:

$$Q^\pi = \arg\min_{Q^\pi} \mathbb{E}_{(s,a,s')} \Big[r(s, a) + \gamma\, Q^\pi_{\bar{\phi}}(s', \pi_\theta(s')) - Q^\pi_\phi(s, a) \Big]^2, \tag{1}$$

where $\bar{\phi}$ is the critic target network. The actor parameterized by θ is updated with the deterministic policy gradient (Silver et al., 2014):

$$\pi_\theta = \text{argmax}_{\pi_\theta} \mathbb{E}_s \left[Q^\pi(s, \pi_\theta(s))\right]. \tag{2}$$

Specifically, the policy parameters θ are updated in proportion to the gradient $\nabla_\theta Q(s, \pi_\theta(s))$. By applying the chain rule, we have:

$$\theta \leftarrow \theta + \mathbb{E}_s \left[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s, a)|_{a=\pi_\theta(s)}\right]. \tag{3}$$

This means that the optimal policy is the action at an extremal of the Q function where $\nabla_a Q^\pi(s, a) = 0$.

Offline Reinforcement Learning

In the offline setting, the agent relies on an existing dataset denoted as $B = \{(s_i, a_i, r_i, s'_i, a'_i)\}_{i=1}^{|B|}$ generated by a behavior policy denoted as μ. The datasets usually cover only a portion of all possible state-action pairs. To mitigate extrapolaion errors (Fujimoto et al., 2019; Kumar et al., 2019), behavior regularization is often applied, which involves adding regularizers to ensure the learned policy does not deviate too much from the pre-collected data (Kostrikov et al., 2021; Shilova et al., 2023).

Revisited BRAC (ReBrac) is a state-of-the-art offline method introduced by Shilova et al. (2023). The core idea behind ReBrac is to apply actor and critic penalizations in an actor-critic algorithm. In ReBrac, the policy update is built on TD3 with an L^2 actor penalization:

$$\pi_\theta = \text{argmax}_{\pi_\theta} \mathbb{E}_{(s,a)\sim B} \Big[Q^\pi_\phi(s, \pi_\theta(s)) - \lambda_\pi \cdot (\pi_\theta(s) - a)^2 \Big], \tag{4}$$

where λ_π is the weight of the regularization term. The state-action value function update with critic penalization is defined by:

$$Q_\phi = \text{argmin}_{Q_\phi} \mathbb{E}_{(s,a,s')\sim B} \Big[r(s, a) + \gamma\, Q^\pi_{\bar{\phi}}(s', \pi(s')) - Q^\pi_\phi(s, a) - \lambda_Q \cdot (\pi_\theta(s) - a)^2 \Big]^2, \tag{5}$$

where λ_Q is the weight of the regularization term. Previous methods in offline RL emphasized the regularization of the actions. In contrast, we improve generalization using the derivatives of the state-action value function Q for both the state and action space.

Continuous-time Reinforcement Learning and Hamilton-Jacobi-Bellman (HJB) Equation

To incorporate the first-order information of the value function, we formulate the value function as an HJB equation in a continuous-time dynamical system (Doya, 2000):

$$\dot{s}(t) = f(s(t), a(t)), \quad t \geq 0, \tag{6}$$

where $s(t)$ is the state of the system at time t, and $a(t)$ denotes the control action. The function f represents the system's dynamics, determining how the state evolves in response to the control actions.

The goal of optimal control is to maximize a cumulative measure of performance, typically framed as an infinite-horizon discounted optimal control problem. The Q-function is defined by (Kim et al., 2021):

$$Q^{\text{HJB}}(s,a) := \max_{a\in A} \int_0^\infty e^{-\gamma t} r(s(t),a(t))dt. \quad (7)$$

Here, r is the reward that quantifies the instantaneous benefit of being in state $s(t)$ while taking action $a(t)$, and γ is the discount factor. This Q-function represents the maximum possible return, given initial conditions $s(0)$ and $a(0)$, and then subsequently following the optimal control strategy.

The dynamic programming equation of the Q-function in the continuous-time cases is expressed as an HJB equation, which can be derived by the dynamic programming principle and using Taylor expansion (Kim et al., 2021) :

$$\gamma Q^{\text{HJB}}(s,a) = \max_{a\in A} \Big[r(s,a) + \nabla_s Q^{\text{HJB}}(s,a)\cdot f(s,a) + \nabla_a Q^{\text{HJB}}(s,a)\cdot \dot{a} \Big], \quad (8)$$

where $\nabla_s Q^{\text{HJB}}(s,a)$ and $\nabla_a Q^{\text{HJB}}(s,a)$ represent the gradients of the Q-function with respect to the state and action, respectively. Under this formulation, the optimal policy is given by the action that can maximize the right-hand side of the HJB equation:

$$\pi^{\text{HJB}}(s) = \arg\max_{a\in A} \Big[r(s,a) + \nabla_s Q^{\text{HJB}}(s,a)\cdot f(s,a) + \nabla_a Q^{\text{HJB}}(s,a)\cdot \dot{a} \Big]. \quad (9)$$

In addition, multimodal policies can be defined using the Boltzmann distribution:

$$\pi^{\text{HJB}}(a|s) = \frac{\exp\left[r(s,a) + \nabla_s Q^{\text{HJB}}(s,a)\dot{s} + \nabla_a Q^{\text{HJB}}(s,a)\dot{a}\right]}{\int \exp\left[r(s,a) + \nabla_s Q^{\text{HJB}}(s,a)\dot{s} + \nabla_a Q^{\text{HJB}}(s,a)\dot{a}\right] da}, \quad (10)$$

which simplifies the likelihood of action a given state s being directly proportional to the aforementioned exponential term.

In the previous work by Kim et al. (2021), the HJB equation is adapted for use with a standard Q-Learning. In their approach, they use the HJB equation to determine the optimal action. It is beneficial since their Q-function is correctly learned from an online setting. Our approach, however, targets a different challenge – the difficulty of correctly learning a Q-function for the unvisited states in an offline setting. To address this, we leverage the HJB equation as a means to describe the dynamics captured in the dataset and integrate it into the Q-function learning process.

4. Optimizing First-Order State-Action Dynamics in Q function

A key challenge in offline RL is to properly evaluate out-of-distribution states and actions. To improve the value function estimation, our approach incorporates additional derivative information from the value function. We first introduce an objective function for first-order consistency, grounded in the HJB equation, which is detailed in Section 4.1. Next, Section 4.2 illustrates how we embed this first-order consistency objective within deterministic policy gradient methods. Finally, in Section 4.3, we outline a practical algorithm that enhances the optimization stability.

4.1. First-Order Consistency

We propose a novel approach that bridges between discrete- and continuous-time RL. Firstly, we use discrete-time RL to learn the state-action value function from offline data, ensuring that it aligns with the properties of the first-order HJB equation. Our primary insight here is that the first-order HJB equation values can be approximated directly from the dataset, without the need to solve PDE explicitly. Specifically, with a dataset containing tuples $(s,a,r,s',a') \sim B$, we approximate the dynamics $f(s,a)$ and $\dot{a}$ using finite differences: $f(s,a) = (s'-s)/\Delta_t$ and $\dot{a} = (a'-a)/\Delta_t$, where Δ_t is the time step size used in the environment. In the majority of offline datasets, particularly those derived from simulators, the control frequency is consistent. This implies Δ_t is a constant. For the sake of simplicity and without losing generality in our analyses and modeling, we assume $\Delta_t = 1$. Hence, we calculate the change in states as $f(s,a) = s' - s$ and the change in actions as $\dot{a} = a' - a$, which are crucial for Equations (9) and (10).

Remark 1. For datasets where Δ_t is a not constant, we can compute the $f(s,a) = (s'-s)/\Delta_t$ and $\dot{a} = (a'-a)/\Delta_t$ with the actual Δ_t.

Our second key insight pertains to the consistency between discrete- and continuous-time RL. Rather than striving to fit identical functions across these two paradigms, we focus on consistency in decision-making. Specifically, if a value function learned via an RL algorithm closely approximates the solution to the HJB equation, it will result in similar decision-making outcomes. This perspective allows us to use the values derived from the HJB equation based on dataset information. We formally define first-order consistency as the following definition.

Definition 1 (Offline First-Order Consistency with dataset dynamic)**.** Let Q^π be a state-action value function learned from any RL algorithm. Give a datapoint in an offline dataset, (s,a,s',a'), a learned Q^π function is first-order consistent with the dataset dynamic if and only if the action

Algorithm 1 Updating Deterministic Policy Gradient with First-Order Consistency

Require: Offline dataset $B = \{s_i, a_i, r_i, s'_i\}_{i=1}^{|B|}$, number of iterations T, number of update samples T_{upd}

1: Initialize policy π_{θ_0}, value function Q_{ϕ_0}
2: **for** $t = 0, \cdots, T-1$ **do**
3: Sample a tuple $T_{\text{upd}} = \{s, a, r, s'\}$ from the offline dataset
4: Compute the Bellman update Loss and first-order consistency loss and update Q function by Equation 14:

$$Q = \text{argmin}_Q \mathbb{E}_{(s,a,s')}\Big[\Big(r + \gamma\, Q_{\bar{\phi}_t}(s', \pi(s')) - Q_{\phi_t}(s,a)\Big)^2 - \frac{\nabla_s Q_{\phi_t}(s,a)\dot{s}}{\|\nabla_s Q_{\phi_t}(s,a)\|\|\dot{s}\|} + \|\nabla_a Q_{\phi_t}(s,a)\|^2\Big].$$

5: Update policy $\pi = \text{argmax}_\pi \mathbb{E}_{(s,a)}\left[Q_{\phi_t}(s, \pi_{\theta_t}(s))\right]$
6: **end for**

a in the dataset satisfies the following condition:

$$a = \arg\max_{\pi(s)} \Big[r(s,\pi(s)) + \nabla_s Q^\pi(s,\pi(s))(s'-s) + \nabla_a Q^\pi(s,\pi(s))(a'-a)\Big], \tag{11}$$

or

$$\pi(a|s) \propto \exp\Big[r(s,\pi(a|s)) + \nabla_s Q^\pi(s,\pi(a|s))(s'-s) + \nabla_a Q^\pi(s,\pi(a|s))(a'-a)\Big]. \tag{12}$$

Remark 2. First-order consistency is characterized by the alignment of Q^π function landscape to the HJB equation (Equations (9) - (12)). It is a local consistency which implies that the gradient of Q^π with respect to the state s, $\nabla_s Q^\pi(s,a)$, aligns in a similar direction to $\dot{s}$ itself, and analogously, the gradient with respect to action a, $\nabla_a Q^\pi(s,a)$, aligns with $\dot{a}$.

4.2. Deterministic Policy Gradient with First-Order Consistency

In this section, we primarily focus on the deterministic policy aspect of the actor-critic algorithm. However, we also discuss a practical algorithm tailored for stochastic policy algorithms in Section 6, for completeness.

We incorporate first-order consistency with the deterministic policy of an actor-critic algorithm, TD3 (Fujimoto et al., 2018). In Equation (3), for a policy in TD3 method to achieve the first-order consistency at a data sample (s,a) in the dataset, the condition $\nabla_a Q^\pi(s,a) = 0$ must be satisfied. Following Equation (11), the corresponding loss function of first-order consistency in Definition 1 is adapted as follows:

$$Q_\phi = \text{argmin}_{Q_\phi} \mathbb{E}_{(s,a,s')}\Big[(r + \gamma Q^\pi_{\bar{\phi}}(s', \pi_\theta(s')) - Q^\pi_\phi(s,a))^2 - \lambda_s \nabla_s Q^\pi_\phi(s,a)\dot{s} + \lambda_d \|\nabla_a Q^\pi_\phi(s,a)\|^2\Big]. \tag{13}$$

The first term in the expectation is from the TD3 Bellman error in Equation (1). The second term accounts for the first-order consistency of the state derivative in Equation (11). The last term comes from the TD3 policy constraint $\nabla_a Q^\pi(s,a) = 0$. In practice, we turn the hard policy constraint into a soft constraint with L^2 norm to stabilize the training. We balance between the loss function terms using weights λ_s and λ_d.

To maintain numerical stability and prevent the state gradient $\nabla_s Q^\pi(s,a)$ from becoming excessively large, which could disproportionately influence the loss, we set the weight λ_s to normalize the term by its magnitude. We then set $\lambda_d = 1$:

$$Q_\phi = \text{argmin}_{Q_\phi} \mathbb{E}_{(s,a,s')}\Big[(r + \gamma Q^\pi_{\bar{\phi}}(s', \pi_\theta(s')) - Q^\pi_\phi(s,a))^2 - \frac{\nabla_s Q^\pi_\phi(s,a)\dot{s}}{\|\nabla_s Q^\pi_\phi(s,a)\|\|\dot{s}\|} + \|\nabla_a Q^\pi_\phi(s,a)\|^2\Big]. \tag{14}$$

We detail the algorithm in Algorithm 1.

4.3. Implementations of First-Order Consistency

We apply first-order consistency (Definition 1) to a deterministic policy gradient method, ReBrac (Tarasov et al., 2023). The details are outlined in Algorithm 2 and explained in the following paragraphs.

Weighted Consistency of Derivatives

The data gathered for training may not always be optimal. To address this, we use a weighting function W to adjust the first-order consistency loss:

$$\mathrm{W}(r + \mathrm{w_c}), \tag{15}$$

where r is the reward associated with each sample and $\mathrm{w_c}$ is a constant. This adjustment is based on the reward that measures the effectiveness of a particular action in a given state. By guiding the value function to fit more closely with the derivatives of the data when policies produce better actions, we can improve the accuracy of the value function.

In our implementation, we set $W(x) = \exp(x)$ in the Mujoco task. In Antmaze, we modified the reward function

Algorithm 2 Practical Offline First-Order Consistency Algorithm

Require: Offline dataset $B = \{s_i, a_i, r_i, s'_i\}_{i=1}^{|B|}$, number of iterations T, number of update samples T_{upd}, action gradient threshold g_a, Weighted function W, weighted constant w_c, actor regularization weight λ_π, critic regularization weight λ_Q, first-order consistency weight λ_f,

1: Initialize policy π_{θ_0}, value function Q_{ϕ_0}
2: **for** $t = 0, \cdots, T-1$ **do**
3: Sample a tuple $T_{\text{upd}} = \{s, a, r, s'\}$ from the offline dataset
4: Compute the Bellman update Loss and first-order consistency loss and update Q function by Equation 16, 17, 15:

$$Q = \arg\min_{Q} \Bigg[\mathbb{E}_{(s,a,s')} \left[r + \gamma\, Q_{\bar{\phi}_t}(s', \pi(s')) - Q_{\phi_t}(s,a) - \lambda_Q \cdot (\pi_{\theta_t}(s) - a)^2 \right]^2$$
$$+ \lambda_f \mathrm{W}(r + \mathrm{w_c}) \Bigg[\underbrace{- \frac{\nabla_s Q_{\phi_t}(s,a) \cdot \dot{s}}{\|\nabla_s Q_{\phi_t}(s,a)\| \|\dot{s}\|} - \underbrace{\frac{\nabla_s Q_{\phi_t}(\text{Near}(s),a) \cdot \dot{s}}{\|\nabla_s Q_{\phi_t}(\text{Near}(s),a)\| \|\dot{s}\|}}_{\text{Nearby State Consistency}}}_{\text{State Consistency}} + \underbrace{\underbrace{\mathbb{1}_{\|\nabla_a Q_{\phi_t}(s,a)\|^2 > g_a}}_{\text{Action Indicator}} \cdot \|\nabla_a Q_{\phi_t}(s,a)\|^2}_{\text{Action Consistency}} \Bigg] \Bigg].$$

5: Update policy $\pi = \text{argmax}_\pi \mathbb{E}_{(s,a)} \left[Q_{\phi_t}(s, \pi_{\theta_t}(s)) - \lambda_\pi \cdot (\pi_{\theta_t}(s) - a)^2 \right]$
6: **end for**

by setting $W(x) = 100x$, following the approach of Chen et al. (2022) and Shilova et al. (2023). In Adroit, we choose $W(x) = x$ since the maximum reward value in this environment can exceed 100. Regarding the constant parameter $\mathrm{w_c}$, we select $\mathrm{w_c}$ to be the negative average reward of the dataset in Mujoco. For other environments, we set $\mathrm{w_c} = 0$.

Consistency of State Derivatives

We anticipate that the derivatives of the value function $\nabla_s Q^\pi(s,a)$ will be in accordance with the state dynamics $\dot{s}$ in the dataset. In addition to the current state s, we also retain the consistency of nearby state derivatives to enhance the structure of the value function landscape. Here, the nearby state is defined by $\text{Near}(s) = s - e(s' - s)$, where s' represents the next state and e is a number uniformly sampled from $[0, 1]$. Therefore, we formulate the loss of the state consistency as follows:

$$\mathbb{E}_{(s,a,s')} \left[- \frac{\nabla_s Q^\pi(s,a)\dot{s}}{\|\nabla_s Q^\pi(s,a)\| \|\dot{s}\|} - \frac{\nabla_s Q^\pi(\text{Near}(s),a)\dot{s}}{\|\nabla_s Q^\pi(\text{Near}(s),a)\| \|\dot{s}\|} \right]. \tag{16}$$

The aim of this regularization is to restrict the gradient directions of neighboring points in the state space. This ensures that the change in the state from s to s' aligns consistently across neighboring points. As a result, it helps us effectively capture the underlying structure of the state space.

Consistency of Action Derivatives

When integrating the first-order consistency with deterministic policy gradient methods, the optimal action a must meet the condition $\nabla_a Q^\pi(s,a) = 0$. Since actions in the dataset may not always be optimal, we introduce a threshold g_a to penalize the value function only when $\nabla_a Q^\pi(s,a)$ exceeds this threshold. This is done by adding an indicator function to the consistency loss of action derivatives:

$$\mathbb{E}_{(s,a,s')} \left[\mathbb{1}_{\{\|\nabla_a Q^\pi(s,a)\|^2 > g_a\}} \|\nabla_a Q^\pi(s,a)\|^2 \right]. \tag{17}$$

This adjustment is important to prevent the policy from overfitting to non-optimal actions in the dataset. By introducing an indicator function, we increase the flexibility and robustness of the learning process, which is essential when dealing with imperfect data.

Hyperparameters

Algorithm 2 describes our implementation details, highlighting the first-order consistency method in blue. Our base algorithm is ReBrac (Tarasov et al., 2023), which is an ensemble-free actor-critic framework. In this framework, the actor is updated with a penalty of $\lambda_\pi \cdot (\pi_{\theta_t}(s) - a)^2$, and the critic is updated with $\lambda_Q \cdot (\pi_{\bar{\theta}_t}(s) - a)^2$. We set general hyper-parameters such as the discount factor γ, penalty weights λ_π, and λ_Q as suggested by Tarasov et al. (2023). The weight of the first-order consistency λ_f and the threshold for the action derivatives g_a are provided in Appendix C.

5. Results and Evaluation

5.1. Performance

We conducted an evaluation of our approach on four D4RL tasks, namely Gym-MuJoCo, AntMaze, Adroit, and Franka Kitchen. To assess the performance of our approach, we compared it to several ensemble-free baselines, including ReBrac (Tarasov et al., 2023), RVS (Emmons et al.,

	Ours	ReBrac	RVS	ATAC	F-BRC	DT	TD3+BC	CQL	IQL	BC
Walker2d-medium	87.0 ± 5.9	82.5	71.7	**89.6**	78.8	74.0	83.7	79.5	80.7	75.3
Walker2d-medium-replay	87.5 ± 8.2	77.3	60.6	**92.5**	41.8	66.6	81.8	76.8	75.4	26.0
Hopper-medium	**102.0 ± 1.0**	**102.0**	60.2	85.6	99.4	67.6	59.3	61.9	65.2	52.9
Hopper-medium-replay	**101.3 ± 0.8**	98.1	73.5	102.5	35.6	82.7	60.9	86.3	89.6	18.1
Halfcheetah-medium	**66.9 ± 3.9**	65.6	41.6	53.3	41.3	42.6	48.3	46.9	50.0	42.6
Halfcheetah-medium-replay	**52.9 ± 0.5**	51.0	38.0	48	43.2	36.6	44.6	45.3	42.1	36.6
Mujoco Performance	**82.6**	79.4	57.6	78.6	56.7	61.7	63.1	66.1	67.2	41.9
Umaze	**98.0 ± 1.5**	97.8	65.4	-	-	65.6	78.6	74.0	83.3	54.6
Medium-play	**88.4 ± 4.3**	84.0	58.1	-	-	1.0	10.6	61.2	64.6	45.6
Large-play	**75.0 ± 5.0**	60.4	32.4	-	-	0.0	0.2	15.8	42.5	0.0
Umaze-diverse	**94.0 ± 2.2**	88.3	60.9	-	-	51.2	71.4	84.0	70.6	0.0
Medium-diverse	**88.2 ± 3.5**	76.3	67.3	-	-	0.6	3.0	53.7	61.7	0.0
Large-diverse	**70.4 ± 4.4**	54.4	36.9	-	-	0.2	0.0	14.9	27.6	0.0
Antmaze Performance	**85.7**	76.9	53.5	-	-	19.8	27.3	50.6	58.4	16.7
Pen-cloned	**107.8 ± 9.6**	91.8	-	43.7	-	-	61.4	39.2	77.2	56.9
Pen-expert	**154.7 ± 7.4**	154.1	-	136.2	-	-	146.0	107.0	133.6	85.1
Hammer-cloned	**21.1 ± 6.6**	6.7	-	1.1	-	-	0.8	2.1	1.1	0.8
Hammer-expert	**135.6 ± 0.5**	133.8	-	126.9	-	-	117.0	86.7	129.6	125.6
Relocate-cloned	**1.2 ± 1.1**	0.9	-	0.2	-	-	-0.1	-0.1	0.2	-0.1
Relocate-expert	**110.5 ± 1.1**	106.6	-	99.4	-	-	107.3	95.0	106.5	101.3
Door-cloned	0.2 ± 0.3	1.1	-	**3.7**	-	-	0.1	0.4	0.8	-0.1
Door-expert	**105.7 ± 0.9**	104.6	-	99.3	-	-	84.6	101.5	105.3	34.9
Adorit Performance	**79.6**	75.0	-	63.8	-	-	64.6	54.0	69.3	50.6
Partial	**52 ± 2.0**	-	51.4	-	-	-	-	-	49.8	38
Complete	**74.5 ± 12.6**	-	38	-	-	-	-	-	43.8	65
Franka Preformance	**63.3**	-	44.7	-	-	-	-	-	46.8	51.5

Table 1. The performance of first-order consistency was benchmarked against baseline models, with results averaged across four random seeds. Following the work of Fu et al. (2020), the scores in this table have been normalized using $(S_o - S_r)/(S_e - S_r)$, where S_o, S_r, and S_e denote the rewards achieved by the offline policy, random policy, and expert policy. Note that the baseline results were copied from the papers of ReBrac and RVS.

2022), IQL (Kostrikov et al., 2022), DT (Chen et al., 2021), TD3+BC (Fujimoto & Gu, 2021), and CQL (Kumar et al., 2020), F-BRC (Kostrikov et al., 2021).

The results on the aforementioned tasks are available in Table 1, and we report our scores over five seeds. The best-performing algorithms are highlighted in bold face. Notably, our approach outperformed baselines on overall performance across the tasks.

It is worth mentioning that, from an implementation perspective, F-BRC (Kostrikov et al., 2021) can be considered as a simplified version of our first-order consistency, although F-BRC was derived based on the idea of policy regularization, and it is essentially different from ours. Specifically, F-BRC minimizes $\|\nabla_a Q^\pi(s,a)\|^2$ when integrating with existing offline RL algorithms. In this subsection, the base algorithm and the network architecture of F-BRC were different from ours. We will compare our approach with F-BRC in the ablation study using the same base algorithm and network architecture.

5.2. Ablation Study

We have evaluated the effectiveness of each component in the first-order consistency for offline RL applications. These components include state consistency, nearby state consistency, action indicator, and action consistency. For the details on these components, please refer to Algorithm 2. Additionally, we have also evaluated the effectiveness of normalization in state consistency. Lastly, we have analyzed the performance of the first-order consistency by removing both state consistency and action indicator. We evaluated this variant because it is the same as F-BRC (Kostrikov et al., 2021) from the implementation perspective.

Our analysis, presented in Table 2, demonstrates that our first-order consistency is effective on the Antmaze datasets,

	Ours	w/o Nearby State Cosistency	w/o State Consistency	w/o Action Indicator	w/o Action Consistency	State Consistency w/o normalization	w/o State Consistency & Action Indicator
Umaze-diverse	94.0 ± 2.2	91.4 ± 3.5	80.4 ± 17.2	95.2 ± 2.1	93.6 ± 1.9	72.0 ± 18.2	85.2 ± 13.7
Medium-diverse	88.2 ± 3.5	75.6 ± 18.7	84.4 ± 4.5	75.6 ± 17.9	77.2 ± 20.7	80.6 ± 17.3	81.8 ± 5.6
Large-diverse	70.4 ± 4.4	48.5 ± 30.0	50.6 ± 20.7	63.4 ± 6.2	66.2 ± 14.7	50.8 ± 31.1	43.8 ± 39.9

Table 2. The results of the ablation study indicate the effectiveness of each component in the presented first-order consistency. Details of each component are described in Algorithm 2.

and none of the components can be eliminated. For a more detailed analysis of Mujoco and Antmaze datasets, please refer to Appendix B.

6. Discussion: Stochastic Policy Gradient with First-Order Consistency

We have integrated first-order consistency into Soft-Actor-Critic (SAC) (Haarnoja et al., 2018), a stochastic actor-critic algorithm, to show the flexibility of our method. In SAC, the policy's objective function is defined as:

$$D_{\mathrm{KL}}\left(\pi(\cdot|s)\middle\|\frac{\exp(Q_\phi^\pi(s,\cdot))}{Z^\pi(s)}\right), \tag{18}$$

where $Z^\pi(s)$ is used to normalize the distribution.

To ensure first-order consistency, the policy induced by Q^π must satisfy Equation (12). An intuitive way to integrate this equation into SAC is to minimize

$$D_{\mathrm{KL}}\left(\pi(\cdot|s)\middle\|\frac{\exp(r(s,\cdot)+\nabla_s Q_\phi^\pi(s,\cdot)\dot{s}+\nabla_a Q_\phi^\pi(s,\cdot)\dot{a})}{Z^{\mathrm{HJB}}(s)}\right). \tag{19}$$

Similarly, $Z^{\mathrm{HJB}}(s)$ is used to normalize the distribution. However, the term $Q_\phi^\pi(s,a)$ in Equation (18) accounts for future returns in a long trajectory, whereas Equation (19) focuses solely on local characteristics around specific states. Therefore, we utilize the reward shaping technique (Hu et al., 2020) to incorporate first-order consistency into the Bellman equation by redefining the reward as $r(s,a)+\lambda_r \exp\left(r(s,a)+\nabla_s Q_\phi^\pi(s,a)\dot{s}+\nabla_a Q_\phi^\pi(s,a)\dot{a}\right)$, where λ_r is a balancing weight. The corresponding loss for training the soft value function then becomes

$$\begin{aligned} J_{Q^\pi}(\phi) =& \mathbb{E}_{(s,a)\sim\mathcal{D},a'\sim\pi_\theta(\cdot|s')}\Bigg[\Bigg(Q_\phi^\pi(s,a)-\Big(r(s,a) \\ &+\lambda_r \exp\left(r(s,a)+\nabla_s Q_\phi^\pi(s,a)\dot{s}+\nabla_a Q_\phi^\pi(s,a)\dot{a}\right) \\ &+\gamma\left(Q_{\bar{\phi}}^\pi(s',a')-\alpha\log\pi_\theta(a'|s')\right)\Big)\Bigg)^2\Bigg]. \end{aligned} \tag{20}$$

In Equation (20), the reward shaping term is highlighted in blue, and the other terms are the SAC Bellman error.

Stochastic Policy	EDAC	First-Order Consistency
Walker2d-medium	92.5 ± 0.8	**95.1 ± 1.4**
Walker2d-medium-replay	87.1 ± 2.4	**94.1 ± 2.0**

Table 3. The performance of first-order consistency on stochastic policy gradient method in Algorithm 3.

We show the implementation details in Algorithm 3 in Appendix A.

We have implemented Algorithm 3 on an offline stochastic policy gradient method, EDAC (An et al., 2021). Evaluations were conducted on the Walker2d-medium and Walker2d-medium-replay datasets, with results reported over five seeds. Experiment results in Table 3 indicate that our first-order consistency approach can benefit the stochastic policy gradient framework as well.

7. Conclusions

We have developed a new strategy to learn value functions from pre-collected data more effectively. Our approach combines both discrete- and continuous-time RL techniques to improve policy efficacy. Specifically, we use the Bellman equation of discrete RL to learn the value function and ensure that its first derivative aligns with the local characteristics defined by the HJB equation. This helps to integrate first-order information into the learning process, resulting in more accurate value function estimations. The strategy also sidesteps the direct computation involved in continuous system dynamics, enabling the integrated algorithm to handle states and actions in high-dimensional space. Our method improves the performance of existing policy gradient algorithms, as demonstrated in the experiments.

Impact Statements

This paper aims to contribute to the advancement of the Machine Learning field. While recognizing that our work could have various societal implications, we believe that none of these consequences necessitate specific emphasis within this document.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

Boyan, J. and Moore, A. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, 7, 1994.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chen, X., Ghadirzadeh, A., Yu, T., Gao, Y., Wang, J., Li, W., Liang, B., Finn, C., and Zhang, C. Latent-variable advantage-weighted policy optimization for offline rl. *arXiv preprint arXiv:2203.08949*, 2022.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Dong, K., Yang, J., and Ma, T. Provable model-based nonlinear bandit and reinforcement learning: Shelve optimism, embrace virtual curvature. *Advances in neural information processing systems*, 34:26168–26182, 2021.

Doya, K. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *International Conference on Learning Representations*, 2022.

Fleming, W. H. and Soner, H. M. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hu, Y., Wang, W., Jia, H., Wang, Y., Chen, Y., Hao, J., Wu, F., and Fan, C. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Kim, J., Shin, J., and Yang, I. Hamilton-jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *The Journal of Machine Learning Research*, 22(1):9363–9396, 2021.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Krylov, N. V. *Controlled diffusion processes*, volume 14. Springer Science & Business Media, 2008.

Kumar, A., Singh, A., Tian, S., Finn, C., and Levine, S. Stabilizing off- policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Lutter, M., Belousov, B., Listmann, K., Clever, D., and Peters, J. Hjb optimal feedback control with deep differential value functions and action constraints. In *Conference on Robot Learning*, pp. 640–650. PMLR, 2020.

Martin, J., Lyskawinski, M., Li, X., and Englot, B. Stochastically dominant distributional reinforcement learning. In *International conference on machine learning*, pp. 6745–6754. PMLR, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Munos, R. A convergent reinforcement learning algorithm in the continuous case based on a finite difference method. In *IJCAI (2)*, pp. 826–831, 1997.

Munos, R. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40:265–299, 2000.

Munos, R. and Bourgine, P. Reinforcement learning for continuous stochastic control problems. *Advances in neural information processing systems*, 10, 1997.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in neural information processing systems*, 32, 2019.

Parag, A., Kleff, S., Saci, L., Mansard, N., and Stasse, O. Value learning from trajectory optimization and sobolev descent: A step toward reinforcement learning with superlinear convergence properties. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 01–07. IEEE, 2022.

Shilova, A., Delliaux, T., Preux, P., and Raffin, B. Revisiting continuous-time reinforcement learning. a study of hjb solvers based on pinns and fems. In *Sixteenth European Workshop on Reinforcement Learning*, 2023.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395. PMLR, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. In *Nature*, volume 550, pp. 354–359. Nature Publishing Group, 2017.

Singh, A., Yu, A., Yang, J., Zhang, J., Kumar, A., and Levine, S. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

Tarasov, D., Kurenkov, V., Nikulin, A., and Kolesnikov, S. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2023.

Vemula, A., Sun, W., and Bagnell, J. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2926–2935. PMLR, 2019.

Wang, Y., Agarwal, A., Kakade, S., Langford, J., Mott, A., and Zhang, Y. Critic regularized regression. *arXiv preprint arXiv:2002.09005*, 2020.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Wiltzer, H. E., Meger, D., and Bellemare, M. G. Distributional hamilton-jacobi-bellman equations for continuous-time reinforcement learning. In *International Conference on Machine Learning*, pp. 23832–23856. PMLR, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

Xu, P. and Gu, Q. A finite-time analysis of q-learning with neural network function approximation. In *International Conference on Machine Learning*, pp. 10555–10565. PMLR, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. Gendice: Generalized offline estimation of stationary values. *International Conference on Learning Representations*, 2020.

A. Stochastic Policy Gradient with First-Order Consistency Algorithm

Algorithm 3 Updating Stochastic Policy Gradient with First-Order Consistency

Require: Offline dataset $B = \{s_i, a_i, r_i, s'_i, a'_i\}_{i=1}^{|B|}$, objective function J, step size parameter η, number of iterations T, number of update samples T_{upd}, balancing weight λ_r

1: Initialize policy π_{θ_0}, value function Q_{ϕ_0}
2: **for** $t = 0, \cdots, T-1$ **do**
3: Sample a tuple $T_{\text{upd}} = \{s, a, r, s', a'\}$ from the offline dataset
4: Compute the Bellman update Loss and first-order consistency loss and update Q by Equation 20:

$$Q = \mathbb{E}_{(s,a)\sim\mathcal{D}, a'\sim\pi_\theta(\cdot|s')}\Bigg[\Bigg(Q_\phi(s,a) - \Big(r + \lambda_r \exp\Big(r + \nabla_s Q_\phi(s,a)\dot{s} + \nabla_a Q_\phi(s,a)\dot{a}\Big) \tag{21}$$

$$+ \gamma\Big(Q_{\bar{\phi}}(s',a') - \alpha \log \pi_\theta(a'|s')\Big)\Big)\Bigg)^2\Bigg] \tag{22}$$

5: Update policy $\pi = \text{argmax}_\pi \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\theta(\cdot|s)}\Big[Q_\phi(s,a) - \alpha \log \pi_\theta(a|s)\Big]$
6: **end for**

B. Additional Ablation Study

We conduct a detailed ablation study across various environments in Mujoco and Antmaze to examine the impact of state consistency and action consistency, with results presented in Table 4.

Table 4. Ablation Study of first-order consistency in Mujoco and Antmaze.

	Ours	w/o Action Consistency	w/o State Consistency
Walker2d-medium	**87.0 ± 5.9**	84.7 ± 1.6	84.2 ± 3.1
Walker2d-medium-replay	**87.5 ± 8.2**	75.3 ± 11.9	83.9 ± 5.5
Hopper-medium	**102.0 ± 1.0**	**102.5 ± 0.2**	101.1 ± 2.4
Hopper-medium-replay	**101.3 ± 0.8**	101.0 ± 0.6	85.0 ± 14.9
Halfcheetah-medium	**66.9 ± 3.9**	65.0 ± 1.8	45.4 ± 0.1
Halfcheetah-medium-replay	**52.9 ± 0.5**	50.7 ± 0.8	40.2 ± 1.9
Mujoco Performance	**82.6**	**80.9**	**73.3**
Umaze	**98.0 ± 1.5**	97.4 ± 0.8	97.2 ± 1.4
Medium-play	**88.4 ± 4.3**	82.6 ± 5.2	84.8 ± 3.7
Large-play	**75.0 ± 5.0**	50.2 ± 20.7	62.2 ± 19.9
Umaze-diverse	**94.0 ± 2.2**	93.6 ± 1.9	80.4 ± 17.2
Medium-diverse	**88.2 ± 3.5**	77.2 ± 20.7	84.4 ± 4.5
Large-diverse	**70.4 ± 4.4**	66.2 ± 14.7	50.6 ± 20.7
Antmaze Performance	**85.7**	**77.8**	**76.6**

C. Hyperparameter Setting

For the general hyper-parameters of RL training, we follow ReBrac's setting (Tarasov et al., 2023) as shown in Table 5. For first-order consistency training, there are two additional hyper-parameters: first-order consistency weight λ_f and action gradient threhold g_a. The hyper-parameters are listed in Table 6.

Table 5. Hyperparameter Settings for the experiemtns in this paper.

	Antmaze	Adorit	Mujoco	Franka Kitchen
Optimizer	Adam	Adam	Adam	Adam
Batch Size	256	256	1024	256
Learning Rate	0.0003	0.0003	0.001	0.0003
Hidden Dimension	256	256	256	256
Num of Layers	3	3	3	3
Gamma	0.999	0.99	0.99	0.99
Nonlinearity	ReLU	ReLU	ReLU	ReLU

Table 6. Hyperparameter Settings for the experiments in this paper.

	Weight	ga
Antmaze		
Umaze	0.01	1
Medium-play	0.005	1
Large-play	0.002	0.5
Umaze-diverse	0.01	1
Medium-diverse	0.01	1
Large-diverse	0.01	1
Mujoco		
Walker2d-medium	0.001	1
Walker2d-medium-replay	0.001	1
Hopper-medium	0.001	1
Hopper-medium-replay	0.001	1
Halfcheetah-medium	0.001	0.5
Halfcheetah-medium-replay	0.002	1
Adroit		
Pen-cloned	0.0005	0.5
Pen-expert	0.002	0.2
Hammer-cloned	0.0005	0.5
Hammer-expert	0.001	0
Relocate-cloned	0.001	0.5
Relocate-expert	0.001	0
Door-cloned	0.01	0
Door-expert	0.001	0
Franka Kitchen		
Partial	0.0005	0.5
Complete	0.0001	0.8