

Learning to Solve Stochastic Controls with Unknown Drifts and Running Rewards: Theory, Algorithms and Convergence

Jin Ma*, Gaozhan Wang[†] Jianfeng Zhang[‡] and Xun Yu Zhou[§]

September 13, 2026

Abstract

We study continuous-time and possibly high-dimensional stochastic control problems where drift coefficients and running reward functions are unknown. Due to these missing model primitives we take the exploratory, reinforcement learning (RL) framework of Wang, Zariphopoulou, and Zhou [34] with relaxed controls and entropy regularization. The objective is to develop theoretically grounded, efficient and scalable RL algorithms to learn both the optimal value functions (which also solves the exploratory HJB equation) and optimal exploratory feedback control policies. When the diffusion coefficients do not contain control, we employ probabilistic representations of both the optimal value function and its gradient based on an auxiliary state process depending only on the diffusion part of the original dynamics. With a delicate analysis on some properly defined mappings and their fixed points, this leads to the introduction of our RL algorithms and their convergence. We demonstrate the performance of our algorithms through various numerical examples. Finally, we study a special control-dependent diffusion case where probability representation of the Hessian is called for.

*Department of Mathematics, University of Southern California, Los Angeles, CA 90089; email: jinma@usc.edu. This author is supported in part by NSF grant #DMS-2510403.

[†]Department of Mathematics, University of Southern California, Los Angeles, CA 90089; email: gaozhanw@usc.edu.

[‡]Department of Mathematics, University of Southern California, Los Angeles, CA 90089; email: jianfenz@usc.edu. This author is supported in part by NSF grant #DMS-2510403.

[§]Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027; email: xz2574@columbia.edu. This author is supported in part by the Nie Center for Intelligent Asset Management at Columbia.

Keywords. Stochastic control, reinforcement learning, entropy regularization, exploratory HJB equation, model-free algorithms.

1 Introduction

Stochastic control problems are prevalent in everyday applications. Classical continuous-time stochastic control theory is *model-based*, namely, it specifies a system dynamics, typically a controlled stochastic differential equation (SDE), along with a given running reward function and, when the time horizon is finite, a terminal reward/payoff function. It has developed into a mature theory in the past 60 years or so, with its key pillars including maximum principle, dynamics programming, and linear-quadratic controls [9, 37].

There is, however, a fundamental flaw with this model-based approach. On one hand, some of the dynamics coefficients, especially the drifts, are notoriously hard or outright impossible to estimate to a workable accuracy. Moreover, in many applications one only observes a reward *signal* once an action is applied, instead of knowing the functional form of a running reward. On the other hand, optimal controls and value functions are often very sensitive to those model parameters/coefficients. The conflict between the intrinsic rigidity of a model and the inherent model uncertainty or inaccessibility leads to erroneous, irrelevant and even misleading solutions.

This is where reinforcement learning (RL) comes to the rescue. Model-free RL by-passes estimation of model parameters and seeks optimal controls *directly* based on observed, simulated, or generated data. Because the mid-step of model estimation is skipped, RL solutions are naturally more swift in responding to the (ever) changing environment and more robust.

For a very long time, RL study had almost exclusively focused on discrete-time Markov decision processes (MDPs) [30], even though most of the important real-life applications are continuous time (with continuous state/action spaces) by nature; e.g. autonomous driving, robot navigation and high-frequency trading. Wang, Zariphopoulou, and Zhou [34] are the first to present a continuous-time RL formulation, which employs (randomized) relaxed controls to characterize exploration and an entropy regularizer to capture the exploration-exploitation tradeoff central to RL, and to obtain a theoretically optimal control policy that is a Gibbs sampler via analyzing the associated exploratory Hamilton-Jacobi-Bellman (HJB) equation. Since then, there has been an upsurge of interest in this exploratory approach for continuous-time RL, including extensions to many types of stochastic control

problems (e.g. [4, 6, 28, 36]), as well as various other related problems (e.g. [1, 5, 7, 12, 27]). See also the survey papers by Hambly, Xu, and Yang [14], Hu and Laurière [16], and the references therein. Notably, the subsequent “trilogy” [23, 24, 25] by Jia and Zhou builds the theoretical foundation of model-free, data-driven continuous-time RL algorithms within the exploratory framework of [34].¹ The key technical thrust therein is a martingale theory that naturally leads to online/offline algorithms for policy evaluation, policy gradient, and q -learning (a continuous-time counterpart of the Q -learning for MDPs).

The above trilogy, however, does not address the important questions of convergence of the various algorithms devised and a regret analysis of the learned policies. Tang and Zhou [32] tackle these questions and establish model-free convergence and sublinear regret bounds for q -learning, based on the backward stochastic differential equation (BSDE) and stochastic approximation (SA) theories. A limitation of [32] is its strong and rather complex assumptions arising primarily from the SA technique employed. Meanwhile, [17, 18, 19] derive convergence and sublinear regret bounds for martingale-based policy gradient algorithms, in the more specific model-free settings of LQ controls and mean–variance portfolio selection, again leveraging the SA theory.

This paper studies stochastic controls with state dynamics governed by the Itô diffusion type SDEs. The objective is to design RL algorithms and prove their convergence when the drift coefficients and the running reward functions are unknown. The specific approach we develop requires the diffusion coefficients and terminal reward functions (in the case of a finite time horizon) to be known and given, which is relatively reasonable because the former are easier to estimate (compared to the drifts) and the latter are often specified in many applications (e.g. a payoff of a stock option).

Due to the unknown drifts and running rewards, we work in the exploratory realm of [34], and aim to learn the optimal critic–actor pair (u^*, π^*) , namely the optimal value function and optimal (randomized) feedback policies, both functions of the time–state pair (t, x) . Theoretically, this pair can be approximated through the so-called *policy iteration algorithm* (PIA). PIA is based on a very simple *coupled* relation between optimal critic and actor: the former is the conditional expectation of the total reward functional of the latter, and the latter is the maximizer (or soft maximizer) of the Hamiltonian which is a function of the gradient/Hessian of the former. This relation naturally implies an iterative scheme to approximate the two simultaneously. PIA type of algorithms along with their convergence have been extensively studied in the classical model-based setting (where exploration is not

¹Here and henceforth, by “model-free” we mean the model primitives are unknown, but there is still a basic underlying dynamics *structure* such as a Markov chain or an Itô SDE.

required), in both discrete-time and continuous-time; see the literature review in [26], the predecessor of this paper.

In the exploratory setting with entropy regularization, Wang and Zhou [35], for a mean-variance portfolio choice problem, show that a PIA converges in just two iterations. This is because the problem therein is essentially an LQ problem, for which the optimal Gibbs sampler reduces to Gaussian and thus highly tractable. See also Giegrich, Reisinger, and Zhang [11] for related convergence results in the LQ setting. Huang, Wang, and Zhou [20] study a general infinite horizon model with control-independent diffusion coefficients and establish convergence of their PIA. Tran, Wang, and Zhang [33] consider a similar problem and investigate two cases depending on whether or not control enters into the diffusion term. In both cases, with different additional assumptions, they establish the PIA convergence. The methods of these two papers [20, 33] are both partial differential equation (PDE)-based by analyzing the underlying exploratory HJB equations. By contrast, Ma, Wang and Zhang [26] employ purely probabilistic method, primarily probabilistic representation formulae for the value function and its derivatives, to design PIAs and derive their convergence. A recent work by Huang, Yu, and Zhang [21] extends the probabilistic method and proves the PIA convergence to a time-inconsistent problem with entropy regularization.

The aforementioned papers [20, 33, 26] are all model-based, assuming oracle access to all the model primitives. They share two essential pitfalls. First, policy iteration involves *derivatives* of the value function. While these papers establish convergence of the value function iterates, accurate numerical approximation of the value function alone does not guarantee accurate approximation of its derivatives. Second, the aforementioned model-based PIAs are not directly implementable when some of the coefficients are missing, as in a model-free setting. See more detailed discussions on these two issues at the end of § 2.1.

This paper endeavors to overcome the two pitfalls, in the setting of unknown drifts and running rewards, by stepping up the probabilistic analysis of [26]. The main part of the paper assumes that the diffusion coefficient does not depend on control, as in [26], in which case one needs only to consider the gradient (but not Hessian) of the value function in the analysis. Instead of the critic-actor pair (u^*, π^*) , we consider another pair (v^*, w^*) , where v^* is the gradient of u^* (in the spatial variable) and w^* the Gibbs exponent in the expression of π^* . Delicate probabilistic representation and analysis reveal that this new pair also satisfies a coupled relation that happens to be *independent* of the drift and running reward functions. This in turn leads to an error analysis, as well as algorithms with theoretical guarantee on the convergence of both the value function and its gradient.² The algorithms we develop

²We note that a related difficulty arises when estimating the spatial gradient of a conditioning function;

are data-driven, and we define precisely what “data” and “data-driven” mean; see Remark 2.1 and Assumption 3.3 for details.

As mentioned, the main results of the paper rely on the control-independence of the diffusion coefficients. For the very special case when the state space is one-dimensional, we extend our analysis and algorithm to include the case when control enters into diffusions in infinite time horizon. This calls for an analysis on the Hessian. Despite a very restrictive setting, to the best of our knowledge this is the first data-driven algorithm with a rigorous convergence analysis for continuous-time *nonlinear* stochastic controls with unknown drifts and known yet control-*dependent* diffusions.

We will also present a few numerical examples to demonstrate the efficiency and scalability of our learning algorithms. In particular, we provide an example to compare with a now classical benchmark of Han, Jentzen and E [15] for solving a class of high-dimensional PDEs based on deep BSDEs. The result shows that our algorithm achieves a similar performance in terms of accuracy of the learned solution to the exploratory HJB equation up to the state space dimension of 100 (which reaches the ceiling of our hardware capacity), even though [15] is model-based and ours is model-free.

Finally, this paper also provides a probabilistic numerical method to solve the exploratory HJB equations, which is a class of potentially high-dimensional, nonlinear parabolic or elliptic PDEs with certain unknown coefficients. Such PDEs with missing coefficients are termed *black-box* PDEs in a concurrent paper by Jia et al. [22], which contains a comprehensive literature review and motivations of studying black-box PDEs and/or dealing with the curse of dimensionality. That paper studies a fully nonlinear (i.e. up to the Hessian) parabolic PDE whose coefficients are all unknown and develops model-free, data-driven algorithms. Its key idea is to represent the gradient and Hessian via the so-called zeroth-order derivative estimators derived from perturbed Monte Carlo trajectories, which is fundamentally different from the representation in our paper. The PDE studied in [22] consists of a linear operator which is the generator of a diffusion process plus a nonhomogeneous source term which is fully nonlinear up to the Hessian. Although it includes in form the exploratory HJB equation as a special case, the assumption therein that the value of the nonlinear source term is known for each given input excludes our setting. On the other hand, our equation clearly does not cover the one in [22]. Therefore, the two papers are mutually exclusive and complementary to each other.

The rest of the paper is organized as follows. In §2 we introduce the problem and

 see Guo, Tang, and Xu [13].

highlight our main ideas. In §3 we carry out an analysis necessary for designing the main algorithm and its convergence when time duration is small. The analysis is extended to the general time horizon in §4. In §5 we present the algorithms along with numerical examples. In §6 we study a special case with control-dependent diffusion coefficients and one-dimensional state space, with a numerical example. Finally, §7 concludes. Some of the proofs are placed in Appendix.

2 Problem Formulation

Notations. Here we list a few notations that will be used frequently in the paper.

- For $x, \tilde{x} \in \mathbb{R}^n$, $x \cdot \tilde{x}$ denotes the inner product, and $|x|$ the Euclidean norm.
- For $M, \tilde{M} \in \mathbb{R}^{m \times n}$, $M^\top$ denotes the transpose of M , and $M : \tilde{M} := \text{tr}(M\tilde{M}^\top)$. Moreover, I_d denotes the $d \times d$ identity matrix.
- For a generic topological space $\mathbf{E}$, $C^0(\mathbf{E}; \mathbb{R}^n)$ denotes the space of continuous functions $\phi : \mathbf{E} \rightarrow \mathbb{R}^n$, and $C_b^0(\mathbf{E}; \mathbb{R}^n)$ the subspace of functions with finite uniform norms:

$$\|\phi\|_0 := \sup_{\mathbf{E}} |\phi|. \quad (2.1)$$

- For a measurable set A in some Euclidean space, $\mathcal{P}_0(A)$ denotes the space of probability density functions $\pi : A \rightarrow [0, \infty)$, namely $\pi(a) \geq 0$ a.e. and $\int_A \pi(a) da = 1$. Moreover, $\mathcal{H}(\pi)$ denotes the Shannon entropy of π :

$$\mathcal{H}(\pi) := - \int_A \pi(a) \ln \pi(a) da, \quad \pi \in \mathcal{P}_0(A). \quad (2.2)$$

- For a generic topological space $\mathbf{E}$, a measurable set A in some Euclidean space, and a measurable function $\phi : \mathbf{E} \times A \rightarrow \mathbb{R}^n$, denote

$$\tilde{\phi}(x, \pi) := \int_A \phi(x, a) \pi(a) da, \quad x \in \mathbf{E}, \pi \in \mathcal{P}_0(A). \quad (2.3)$$

2.1 Entropy-regularized stochastic control

Fix a finite time horizon $[0, T]$ and a filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{F})$ on which is defined a standard d -dimensional Brownian motion B , where $\mathbb{F}$ is the natural filtration generated by B . We are also given an action space A , which is a measurable set in some Euclidean space with a finite volume:

$$0 < |A| < \infty.$$

Our primal interest is the following classical, *target* control problem: for $(t, x) \in [0, T] \times \mathbb{R}^d$,

$$\begin{aligned} X_s^{t,x,\alpha} &= x + \int_t^s b(l, X_l^{t,x,\alpha}, \alpha_l) dl + \int_t^s \sigma(l, X_l^{t,x,\alpha}) dB_l, \quad s \in [t, T]; \\ u_0^*(t, x) &:= \sup_{\alpha} \mathbb{E} \left[g(X_T^{t,x,\alpha}) + \int_t^T r(s, X_s^{t,x,\alpha}, \alpha_s) ds \right]. \end{aligned} \quad (2.4)$$

In the above, an *admissible control* α is an A -valued and $\mathbb{F}$ -progressively measurable process, and b, σ, r, g are appropriate coefficients taking values in $\mathbb{R}^d$, $\mathbb{R}^{d \times d}$, $\mathbb{R}$, and $\mathbb{R}$, respectively.³ For the most part of the paper, we assume that σ is control-independent as in (2.4), but a special case of controlled σ will be investigated in §6.

The key feature considered in this paper is that the drift term b and the running reward function r are unknown. Therefore, classical approaches such as dynamic programming (via HJB equations) and maximum principle (via Hamiltonian and FBSDEs) fail to solve the above problem. We resort to the continuous-time reinforcement learning (RL) paradigm introduced by [34] and consider the following *exploratory*, entropy-regularized problem:

$$\begin{aligned} X_s^{t,x,\pi} &= x + \int_t^s \tilde{b}(l, X_l^{t,x,\pi}, \pi(l, X_l^{t,x,\pi})) dl + \int_t^s \sigma(l, X_l^{t,x,\pi}) dB_l, \quad s \in [t, T]; \\ J(t, x; \pi) &:= \mathbb{E} \left[g(X_T^{t,x,\pi}) + \int_t^T [\tilde{r}(s, X_s^{t,x,\pi}, \pi(s, X_s^{t,x,\pi})) + \lambda \mathcal{H}(\pi(s, X_s^{t,x,\pi}))] ds \right], \\ u^*(t, x) &:= u_{\lambda}^*(t, x) := \sup_{\pi \in \mathcal{A}_T} J(t, x; \pi), \quad (t, x) \in [0, T] \times \mathbb{R}^d. \end{aligned} \quad (2.5)$$

Here, $\mathcal{A}_T$ denotes the space of feedback type relaxed controls/policies $\pi : [0, T] \times \mathbb{R}^d \rightarrow \mathcal{P}_0(A)$, $\tilde{b}$ and $\tilde{r}$ are the “convexifications” of b and r respectively defined in (2.3), and $\lambda > 0$ is an exogenous “temperature parameter” capturing a balance between exploitation and exploration essential in the RL approach. It is shown in [34] that the value function u^* satisfies the *exploratory HJB equation*

$$\begin{aligned} u_t^* + \frac{1}{2} [\sigma \sigma^\top](t, x) : u_{xx}^* + H(t, x, u_x^*) &= 0, \quad u^*(T, x) = g(x), \quad \text{where} \\ H(t, x, z) &:= \sup_{\pi \in \mathcal{P}_0(A)} [\tilde{b}(t, x, \pi) \cdot z + \tilde{r}(t, x, \pi) + \lambda \mathcal{H}(\pi)], \end{aligned} \quad (2.6)$$

and the optimal relaxed control π^* takes the Gibbs form

$$\begin{aligned} \pi^*(t, x, a) &:= \Gamma(t, x, u_x^*(t, x), a), \quad (t, x, a) \in [0, T] \times \mathbb{R}^d \times A, \quad \text{where} \\ \Gamma(t, x, z, a) &:= \frac{\gamma(t, x, z, a)}{\int_A \gamma(t, x, z, a') da'}, \quad \text{with } \gamma(t, x, z, a) := \exp \left(\frac{1}{\lambda} [b(t, x, a) \cdot z + r(t, x, a)] \right). \end{aligned} \quad (2.7)$$

³In this paper we assume that the state and Brownian motion have the same dimension for ease of exposition. When they have different dimensions, the analysis is essentially the same except pseudo matrix inverse will need to be involved in certain places.

With the above notation, the Hamiltonian H in (2.6) can be rewritten as

$$H(t, x, z) = \lambda \ln \left(\int_A \gamma(t, x, z, a) da \right). \quad (2.8)$$

The above results show that one should use a Gibbs sampler in general to generate trial-and-error strategies to explore the environment when certain model coefficients are unknown. [31] further shows that the exploratory control problem converges to the original target control problem when the temperature parameter $\lambda \rightarrow 0$. In other words, the target problem (2.4) can be solved via the exploratory problem (2.5) in the RL setting. The goal of this paper is therefore to design efficient numerical algorithms with theoretical guarantees for computing the above (u^*, π^*) without knowing b and r . Note that besides numerically solving both the target and exploratory control problems, solving the exploratory HJB (2.6), which is a nonlinear parabolic PDE (and high dimensional in many applications), is interesting in its own right from a PDE perspective.⁴

To achieve the goal of this sort, the following *policy iteration algorithm* (PIA) has been popular in the literature. Given appropriate initialization u^0 , define (π^n, u^n) , $n \geq 1$, recursively:

$$\pi^n(t, x, a) := \Gamma(t, x, u_x^{n-1}(t, x), a), \quad u^n(t, x) := J(t, x; \pi^n). \quad (2.9)$$

In particular, under some technical conditions, it is shown in [26, Theorem 2.4] that the PIA converges with a super-exponential rate: for some $0 < \eta < 1$,

$$\|u^n - u^*\|_0 + \|\pi^n - \pi^*\|_0 \leq C\eta^{2^n}. \quad (2.10)$$

Despite the very strong theoretical convergence, the above algorithm has two fundamental drawbacks from the *numerical* and *learning* perspectives, which we now explain. First, the iterate π^n in (2.9) depends on u_x^{n-1} , whereas the above iterative scheme only returns the approximated value of u^n , say $\hat{u}^n$. It is well known that the convergence of a function (under, say, the uniform norm (2.1)), does not necessarily lead to the convergence of its derivatives. Therefore, even if $\hat{u}^{n-1}$ approximates u^{n-1} well, $\hat{u}_x^{n-1}$ may not be a desired approximation of u_x^{n-1} .⁵ Consequently $\hat{\pi}^n(t, x, a) := \Gamma(t, x, \hat{u}_x^{n-1}(t, x), a)$ may not approximate π^n satisfactorily, with errors propagating into subsequent iterates. Thus, in *actual*

⁴[31] establishes the well-posedness and regularity of the viscosity solution to the exploratory HJB equation, but does not address the problem of numerically computing the solution.

⁵For example, when using deep neural networks, derivatives are typically computed via the so-called automatic differentiation (auto-diff) which are the exact derivatives of the network representation that approximates a target function.

implementation, the above PIA algorithm may not converge at all. Indeed, in §5 we will present a numerical example (Example 5.3) showing that a simple minded PIA actually diverges.⁶

The more serious issue arises from the unknown model parameters: the Gibbs function γ in (2.7) depends on the coefficients b and r ; so the algorithm is not implementable when they are unknown. Since the main purpose of the exploratory framework is to deal with such models with unknown parameters, we will make this a focal point in developing our new algorithms in this paper.

2.2 Plan of attack

Bearing the aforementioned two main issues in mind, in this paper we propose new implementable algorithms and analyze their convergence. In this subsection we highlight the main ideas for reader's convenience.

To overcome the first issue, we approximate u_x^* *directly* by utilizing the Bismut–Elworthy–Li representation formula [3, 8]. Specifically, denote

$$v^*(t, x) := u_x^*(t, x), \quad w^*(t, x, a) := v^*(t, x) \cdot b(t, x, a) + r(t, x, a). \quad (2.11)$$

Then, by (2.7) and (2.8), we may rewrite the exploratory HJB equation (2.6) as

$$u_t^* + \frac{1}{2}[\sigma\sigma^\top](t, x) : u_{xx}^* + \lambda \ln \int_A e^{-\frac{1}{\lambda} w^*(t, x, a)} da = 0; \quad u^*(T, x) = g(x). \quad (2.12)$$

It follows from the Feynman–Kac and Bismut–Elworthy–Li formulae that we have the following probabilistic presentations of u^* and v^* :

$$u^*(t, x) = \mathbb{E} \left\{ g(\mathcal{X}_T^{t, x}) + \int_t^T \lambda \ln \left[\int_A \exp \left(\frac{1}{\lambda} w^*(s, \mathcal{X}_s^{t, x}, a) \right) da \right] ds \right\}; \quad (2.13)$$

$$v^*(t, x) = \mathbb{E} \left\{ (\nabla \mathcal{X}_T^{t, x})^\top g_x(\mathcal{X}_T^{t, x}) + \int_t^T \lambda \ln \left[\int_A \exp \left(\frac{1}{\lambda} w^*(s, \mathcal{X}_s^{t, x}, a) \right) da \right] N_s^{t, x} ds \right\}. \quad (2.14)$$

In the above, $\mathcal{X}^{t, x}$ denotes the dynamics of the following *reference state*:

$$\mathcal{X}_s^{t, x} = x + \int_t^s \sigma(l, \mathcal{X}_l^{t, x}) dB_l, \quad s \in [t, T], \quad (2.15)$$

and $N^{t, x}$ is the Bismut–Elworthy–Li representation kernel defined by

$$N_s^{t, x} := \frac{1}{s - t} \int_t^s (\sigma^{-1}(l, \mathcal{X}_l^{t, x}) \nabla \mathcal{X}_l^{t, x})^\top dB_l, \quad s \in [t, T], \quad (2.16)$$

⁶The proof of [26, Theorem 2.4] also shows that $\|u_x^n - u_x^*\|_0$ has the same error bound of $C\eta^{2^n}$. However, it is a *theoretical* result assuming that in each iteration the derivative of the *oracle*, instead of an approximate (e.g. auto-diff), solution of the PDE involved is used. The numerical and algorithmic aspects of the problem are not studied in [26].

where $\nabla \mathcal{X}^{t,x}$ is the *variational process* of $\mathcal{X}^{t,x}$, which satisfies the following linear SDE:

$$\nabla \mathcal{X}_s^{t,x} = I_d + \sum_i \int_t^s \sigma_x^i(l, \mathcal{X}_l^{t,x}) \nabla \mathcal{X}_l^{t,x} dB_l^i, \quad s \in [t, T], \quad (2.17)$$

with σ^i denoting the i -th column of σ and B^i the i -th component of B . It is crucial to note that the representation system (2.13)–(2.17) depends on the functional forms of σ and g , but *not* those of b and r .

The remaining task is to find an effective and implementable way to learn the function w^* , again without involving b and r . Our idea is based on the following simple yet crucial observation: for $\Delta t > 0$ small and $a \in A$,

$$\begin{aligned} & \mathbb{E}[u^*(t, X_{t+\Delta t}^{t,x,a}) - u^*(t, x)] - \mathbb{E}[u^*(t, \mathcal{X}_{t+\Delta t}^{t,x}) - u^*(t, x)] \\ & \approx \mathbb{E}[u_x^*(t, x) \cdot (X_{t+\Delta t}^{t,x,a} - \mathcal{X}_{t+\Delta t}^{t,x})] \approx v^*(t, x) \cdot b(t, x, a) \Delta t, \end{aligned} \quad (2.18)$$

where $X^{t,x,a}$ is the state in (2.4) under a constant control $\alpha \equiv a$. Consequently, we have the following estimate of w^* :

$$w^*(t, x, a) \approx \frac{1}{\Delta t} \mathbb{E} \left[\int_x^{X_{t+\Delta t}^{t,x,a}} v^*(t, x') dx' \right] - \frac{1}{\Delta t} \mathbb{E} \left[\int_x^{\mathcal{X}_{t+\Delta t}^{t,x}} v^*(t, x') dx' \right] + r(t, x, a). \quad (2.19)$$

Our algorithms will be based on solving the approximate “fixed point” (v^*, w^*) in (2.14) and (2.19), which we develop in the following sections. Before we proceed, we would like to reiterate about what problem parameters are known and what are unknown, as they will be key in determining the implementability of our algorithm. We do this in the following remarks.

Remark 2.1. (i) Recall that the main feature of this paper is that the functional forms of b and r are unknown, and we aim to devise *data-driven* solutions. We now make the notions of “data” and “data-driven” precise in our setting. Even though b is unknown, we require the state process $X^{t,x,a}$ under any given control a is *observable* (i.e. the process constitutes *data*). This is a natural requirement. For example, in investment the state process is the wealth process of a portfolio (the control). Given a portfolio policy, we can certainly observe our corresponding wealth process even if we do not know anything about the price dynamics of the stocks involved in our portfolio. Likewise, while we do not know the running reward function r , we assume we will get the function *value* $r(t, x, a)$, a so-called *reward signal*, whenever a control a is applied at (t, x) . A data-driven RL algorithm makes use of these observed data along with other available or simulated data to learn final solutions. See e.g. [23, 24, 25] for more discussions on this issue.

(ii) There is a more subtle point regarding available data. For any given (t, x, a) , in calculating v^* based on (2.14) we need to compute $w^*(s, \mathcal{X}_s^{t,x}, a)$ for $s \in [t, T]$, which in view of (2.19) requires the data $X_{s+\Delta t}^{s, \mathcal{X}_s^{t,x}, a}$ and $\mathcal{X}_{s+\Delta t}^{s, \mathcal{X}_s^{t,x}}$ for all s and a . But since $\mathcal{X}_s^{t,x}$ is simulated, its trajectory could spread over the whole space. This in turn demands us to observe the state data process starting from (s, y) for *any* given y , which is not necessarily on the *original* state trajectory starting from (t, x, a) (i.e. in general $y \neq X_s^{t,x,a}$). We will formalize this type of data availability as an assumption; see Assumption 3.3-(i) along with discussions on this assumption in Remark 3.4 in the next section.

(iii) In this paper we assume that the function σ is known. This assumption is relatively benign because it is well understood that, comparing to the mean b , the volatility σ is much easier to estimate to very high accuracy. Also note that due to (2.17) we actually need access to σ_x as well. Given σ , we can *simulate* $\mathcal{X}$ and $\nabla \mathcal{X}$ at low costs. Similarly, we assume the terminal reward function g (and hence g_x) is known, which is reasonable in many applications where g is a known utility or payoff function (e.g. the payoff function of a financial derivative).

(iv) The introduction of the function w^* is central in our approach. Equation (2.14) by itself does not solve v^* , but coupled with (2.19) the pair (v^*, w^*) can be solved *simultaneously*. Moreover, u^* can also be determined by w^* . Now, w^* differs from the Hamiltonian of the *target* problem by a control-*independent* term $\frac{1}{2}[\sigma\sigma^\top](t, x) : u_{xx}^*(t, x)$. In view of the policy improvement theorem ([25, Theorem 2]) and the definition of the q -function ([25, Definition 4]), w^* is essentially the q -function in our particular setting where σ is independent of a . The general q -learning theory developed in [25] employs martingale conditions to design algorithms to learn the q -function without giving convergence results. [32] investigates the convergence and regret of q -learning based on BSDE and stochastic approximation, under technically complex and strong assumptions. By contrast, our paper takes a very different (and delicate) approach via representing the two key functions with each other: the gradient of the optimal value function and the (essentially) q -function, at the cost of having to assume σ and g to be known.

(v) In theory $\mathcal{X}$ and X can be driven by different Brownian motions, although we use the same notation B in (2.4) and (2.15). Indeed, in the numerical examples in §5 below, we will use independent Brownian motions to simulate $\mathcal{X}$ and generate the environmental data X respectively. However, the right side of (2.19) involves the difference of expectations, which are law invariant. Thus employing the *same* Brownian motion B does not cause any essential differences in theoretical analysis. ■

3 Small Time Horizons

In this section we carry out an analysis following the aforementioned idea, under the assumption that the time horizon T is sufficiently small. The case of a general time horizon will be studied in the next section.

In the remainder of the paper, we will impose the following *Standing Assumptions* for the finite horizon setting, without stating explicitly in the results. The first one is about the regularity of the problem coefficients (even though some of them are unknown).

Assumption 3.1. *There exist constants $L_0, L_1, L_* > 0$ such that the following hold true:*

- (i) b, σ, r are bounded by L_0 , and σ is uniformly non-degenerate: $\sigma \sigma^\top \geq \frac{1}{L_0} I_d$.
- (ii) b, r are measurable in a , and b, σ, r are uniformly Lipschitz continuous in x and uniformly $\frac{1}{2}$ -Hölder continuous in t , with a Lipschitz/Hölder constant L_1 .
- (iii) The PDE (2.6) has a unique classical solution u^* satisfying

$$\|u_x^*\|_0 + \|u_{xx}^*\|_0 + \|u_{xxx}^*\|_0 \leq L_*. \quad (3.1)$$

Note that assumptions on $g = u^*(T, \cdot)$ are implied by (iii). Throughout we will use a generic constant C (i.e. its value may change from line to line), which depends only on $d, m, \lambda, |A|$, and L_0, L_1 , but not on L_* or T . When the latter dependence occurs, we use the notation $C_{L_*, T}$.

Remark 3.2. Assumption 3.1 ensures that the state equation (2.4) has a unique strong solution for any given control α as well as a unique weak solution for any constant control a . On the other hand, since the PDE (2.6) is semi-linear, it is standard (see e.g. [10, Chapter 7]) to find sufficient conditions to ensure Assumption 3.1-(iii) by increasing the regularity of b, σ, r . However, because the key estimate for the convergence of our new algorithm, based on Lemma 3.8 below, relies only on L_0, L_1 but not on L_* , we prefer not to impose further regularity assumptions on b, σ, r so that we can focus on L_0, L_1 . Also, for our theoretical analysis, we require only the measurability in a . However, for the numerical examples in §5, we need to approximate integrations in a by Riemann sums, in which case certain regularity in a will further be needed. ■

The next assumption is on the data accessibility/availability from the environment.

Assumption 3.3. (i) *The environment returns a trajectory $\{X_s^{t,x,a} : t \leq s \leq T\}$, for each query with input $(t, x, a) \in [0, T] \times \mathbb{R}^d \times A$, which is the (unique) weak solution to (2.4) under the constant control a .*

- (ii) The environment returns a value $r(t, x, a)$ for each $(t, x, a) \in [0, T] \times \mathbb{R}^d \times A$.
- (iii) The functional forms of σ and g are known and given.

Remark 3.4. From a practical viewpoint it is a rather strong assumption to require the environment to return the state trajectory for *any* (t, x, a) . Imagine one controls a state process with a constant control a starting from initial time 0 and initial state x_0 . She will naturally observe the resulting data trajectory *as she goes*, even if she may not know the drift b . At time t , the state is at $X_t^{0, x_0, a}$ so she will continue to see the trajectory from $(t, X_t^{0, x_0, a}, a)$. However, Assumption 3.3-(i) requires her, at time t , to also see the data process starting from an *off-trajectory* state x , which may not be possible in general.

This being said, there are (important) cases where this assumption is satisfied. Consider the wealth equation in stock investment

$$dX_t = [rX_t + (\mu - r)a]dt + \sigma a dB_t,$$

where r is the risk-free rate (which is assumed to be known as it is typically the bank saving rate or the bond yield), μ and σ are the return rate (which is hard to estimate and hence unknown) and volatility rate respectively of a stock, a is the portfolio (the amount allocated to the stock), and X_t (which is observable) is the total wealth at time t . Here we assume there is only one stock but the following analysis extends trivially to multiple stocks. Because μ is unknown, the state equation above has an unknown drift. Denote by $X^{0, x, a}$ the solution to the equation with $X_0 = x$. Then it is immediate that

$$X_t^{0, y, a} = X_t^{0, x, a} + e^{rt}(y - x), \quad y \neq x.$$

In other words, the data process starting from a *different* state y can be *inferred* from the data process starting from the given x that can be observed. In this case, Assumption 3.3-(i) holds. So the results in this paper can be applied to a broad class of financial portfolio selection problems.

The assumption is also (approximately) satisfied in several other practical settings. For example, a rideshare platform can monitor vehicle trajectories starting from various locations within a region, assuming a sufficiently high vehicle density. A stock trading app with a huge number of users (such as Robinhood) can observe wealth trajectories of different accounts starting from various amounts.

Finally, if b (along with σ) is known then Assumption 3.3-(i) becomes automatic because one can simulate (and therefore observe) the state process starting from any point. ■

Fix $K \geq 1$ and consider the uniform time discretization $\mathbb{T}_K := \{t_k\}_{k=0}^K$ of $[0, T]$, with $\Delta t := \frac{T}{K}$ and $t_k = k\Delta t$, $k = 0, \dots, K$. Denote $\Delta B_{t_{k+1}} = B_{t_{k+1}} - B_{t_k}$. For each (t_k, x) , we first discretize the reference state (2.15) and its variational process (2.17):

$$\begin{aligned} \mathcal{X}_{t_k}^{t_k, x, \Delta t} &= x; \quad \mathcal{X}_{t_{i+1}}^{t_k, x, \Delta t} = \mathcal{X}_{t_i}^{t_k, x, \Delta t} + \sigma(t_i, \mathcal{X}_{t_i}^{t_k, x, \Delta t}) \Delta B_{t_{i+1}}; \\ \nabla \mathcal{X}_{t_k}^{t_k, x, \Delta t} &= I_d; \quad \nabla \mathcal{X}_{t_{i+1}}^{t_k, x, \Delta t} = \nabla \mathcal{X}_{t_i}^{t_k, x, \Delta t} + \sum_{j=1}^d \sigma_x^j(t_i, \mathcal{X}_{t_i}^{t_k, x, \Delta t}) \nabla \mathcal{X}_{t_i}^{t_k, x, \Delta t} \Delta B_{t_{i+1}}^j, \end{aligned} \quad (3.2)$$

for $i = k, \dots, K-1$. Moreover, we define the discretized kernel process from (2.16)

$$N_{t_i}^{t_k, x, \Delta t} = \frac{1}{t_i - t_k} \sum_{j=k}^{i-1} (\sigma^{-1}(t_j, \mathcal{X}_{t_j}^{t_k, x, \Delta t}) \nabla \mathcal{X}_{t_j}^{t_k, x, \Delta t})^\top \Delta B_{t_{j+1}}, \quad i = k+1, \dots, K. \quad (3.3)$$

Note that, due to the singularity of $N_s^{t, x}$ at $s = t$, the above iteration starts with $i = k+1$.

The following estimates are standard; see e.g. [38].

Lemma 3.5. *There exists a constant $C_T > 0$, depending only on d, λ, L_0, L_1 , and T but not on L_* , such that for any (t_k, x) and $s > t_k$, $i > k$,*

$$\begin{aligned} \mathbb{E}_{t_k, x} [|X_s^a - x|^4 + |\mathcal{X}_s - x|^4] &\leq C_T (s - t_k)^2, \quad \mathbb{E}_{t_k, x} [|\nabla \mathcal{X}_s|^4 + |\nabla \mathcal{X}_{t_i}^{\Delta t}|^4] \leq C_T, \\ \mathbb{E}_{t_k, x} [|N_s|^4] &\leq \frac{C_T}{(s - t_k)^2}, \quad \mathbb{E}_{t_k, x} [|N_{t_i}^{\Delta t}|^4] \leq \frac{C_T}{(t_i - t_k)^2}, \\ \mathbb{E}_{t_k, x} [|\mathcal{X}_{t_i} - \mathcal{X}_{t_i}^{\Delta t}|^4 + |\nabla \mathcal{X}_{t_i} - \nabla \mathcal{X}_{t_i}^{\Delta t}|^4] &\leq C_T (\Delta t)^2, \end{aligned} \quad (3.4)$$

where X^a is the state process starting from (t_k, x) under the constant control a . Moreover, the above C_T is increasing in T , and in particular $C_T \leq C_{\bar{T}}$ if $T \leq \bar{T}$. $\blacksquare$

Here and henceforth, for notational simplicity we take the convention that by using $\mathbb{E}_{t_k, x}$ we omit the superscripts t_k, x for the processes under the expectation. For example, in (3.4),

$$\mathcal{X} = \mathcal{X}^{t_k, x}, \quad \nabla \mathcal{X}^{\Delta t} = \nabla \mathcal{X}^{t_k, x, \Delta t}. \quad (3.5)$$

Next, inspired by the relations (2.14) and (2.19) we introduce two mappings:

$$\Phi_K : \mathcal{W}_K \mapsto \mathcal{V}_K, \quad \Psi_K : \mathcal{V}_K \mapsto \mathcal{W}_K, \quad (3.6)$$

$$\text{where } \mathcal{V}_K := C_b^0(\mathbb{T}_K \times \mathbb{R}^d; \mathbb{R}^d), \quad \mathcal{W}_K := C_b^0(\mathbb{T}_K \times \mathbb{R}^d \times A; \mathbb{R}),$$

such that, for any $(v, w) \in \mathcal{V}_K \times \mathcal{W}_K$ and $(t_k, x, a) \in \mathbb{T} \times \mathbb{R}^d \times A$, recalling (3.5),

$$\begin{aligned} \Phi_K(w)(t_k, x) &:= \mathbb{E}_{t_k, x} \left[(\nabla \mathcal{X}_T^{\Delta t})^\top g_x(\mathcal{X}_T^{\Delta t}) + \sum_{j=k+1}^{K-1} \lambda \ln \int_A e^{\frac{1}{\lambda} w(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da N_{t_j}^{\Delta t} \Delta t \right]; \\ \Psi_K(v)(t_k, x, a) &:= \frac{1}{\Delta t} \mathbb{E}_{t_k, x} \left[\int_x^{X_{t_{k+1}}^a} v(t_k, x') dx' - \int_x^{\mathcal{X}_{t_{k+1}}^{\Delta t}} v(t_k, x') dx' \right] + r(t_k, x, a). \end{aligned} \quad (3.7)$$

Remark 3.6. (i) In light of Remarks 2.1-(i), the mappings Φ_K and Ψ_K in (3.7) depend only on the coefficients σ and g which are assumed to be known, as well as the data processes X^a and $\mathcal{X}^{\Delta t}$ along with the reward signals $r(t, x, a)$, which can be either observed or simulated, but not on the functional forms of b and r . This ensures the implementability of any resulting algorithms built on these two mappings.

(ii) Note that, in the above we discretize $\mathcal{X}$ but not X because the former can only be simulated while the latter is observed upon query. However, in the numerical examples in §5 below, we will use simulated data of X , and thus will have to discretize X too. Nevertheless, similarly to the estimates in Lemma 3.5, such a difference has no impact on the final convergence rate. $\blacksquare$

Denote by $v_{\mathbb{T}_K}^* := v^*|_{\mathbb{T}_K} \in \mathcal{V}_K$ and $w_{\mathbb{T}_K}^* := w^*|_{\mathbb{T}_K} \in \mathcal{W}_K$ the restriction of v^*, w^* on $\mathbb{T}_K$. The following result, whose proof is postponed to Appendix, rigorously justifies our construction of (Φ_K, Ψ_K) as a machinery to find the *approximate* fixed point (v^*, w^*) in (2.14) and (2.19) and, moreover, reveals that the approximation is of the order of the square root of the time step size Δt .

Proposition 3.7. *There exists a constant $C_{L_*, T}$, which may depend on L_*, T but not on K or Δt , such that*

$$\|\Phi_K(w_{\mathbb{T}_K}^*) - v_{\mathbb{T}_K}^*\|_0 + \|\Psi_K(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*\|_0 \leq C_{L_*, T} \sqrt{\Delta t}. \quad (3.8)$$

The next estimate is crucial for building our algorithm.

Lemma 3.8. *Let $T \leq \bar{T}$ for some $\bar{T} < \infty$. There exists a constant $C > 0$, depending only on $d, \lambda, L_0, L_1, \bar{T}$, but not on L_*, T, K , such that, for any $(v, w), (v', w') \in \mathcal{V}_K \times \mathcal{W}_K$,*

$$\|\Phi_K(w) - \Phi_K(w')\|_0 \leq C\sqrt{T} \|w - w'\|_0; \quad \|\Psi_K(v) - \Psi_K(v')\|_0 \leq C \|v - v'\|_0. \quad (3.9)$$

Proof Fix $T \leq \bar{T}$ and $K \geq 1$. Recall Remarks 2.1-(v) that we can use the same Brownian motion B in (2.5) and (2.15), and Lemma 3.5 that we can use a generic constant $C = C_{\bar{T}}$ independent of $T \leq \bar{T}$ for all the estimates therein. Fix an arbitrary triplet (t_k, x, a) , and assume without loss of generality that $k = 0$. For any j ,

$$\begin{aligned} \lambda \ln \int_A e^{\frac{1}{\lambda} w(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da &\leq \lambda \ln \int_A e^{\frac{1}{\lambda} [w'(t_j, \mathcal{X}_{t_j}^{\Delta t}, a) + \|w - w'\|_0]} da \\ &= \lambda \ln \int_A e^{\frac{1}{\lambda} w'(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da + \|w - w'\|_0. \end{aligned}$$

Similarly one can prove the symmetric inequality, leading to

$$\left| \lambda \ln \int_A e^{\frac{1}{\lambda} w(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da - \lambda \ln \int_A e^{\frac{1}{\lambda} w'(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da \right| \leq \|w - w'\|_0. \quad (3.10)$$

Then by (3.7), (3.2), and Lemma 3.5, we have

$$\begin{aligned} |(\Phi_K(w) - \Phi_K(w'))(t_0, x)| &\leq C \|w - w'\|_0 \sum_{j=1}^{K-1} \mathbb{E}_{0,x}[|N_{t_j}^{\Delta t}|] \Delta t \\ &\leq C \|w - w'\|_0 \sum_{j=1}^{K-1} \frac{\Delta t}{\sqrt{t_j}} \leq C \|w - w'\|_0 \sqrt{K\Delta t} = C\sqrt{T} \|w - w'\|_0. \end{aligned}$$

Next, note that

$$\begin{aligned} |(\Psi_K(v) - \Psi_K(v'))(t_0, x, a)| &= \frac{1}{\Delta t} \left| \mathbb{E}_{0,x} \left[\int_{\mathcal{X}_{t_1}^{\Delta t}}^{X_{t_1}^a} (v - v')(x') dx' \right] \right| \\ &\leq \frac{1}{\Delta t} \|v - v'\|_0 \mathbb{E}_{0,x}[|X_{t_1}^a - \mathcal{X}_{t_1}^{\Delta t}|] \\ &\leq \frac{1}{\Delta t} \|v - v'\|_0 \mathbb{E}_{0,x} \left[\int_{t_0}^{t_1} |b(s, X_s^a, a)| ds + \left| \int_{t_0}^{t_1} (\sigma(s, X_s^a) - \sigma(t_0, x)) dB_s \right| \right]. \end{aligned}$$

Then, it follows from Assumption 3.1 and Lemma 3.5 that

$$\begin{aligned} |(\Psi_K(v) - \Psi_K(v'))(t_0, x, a)| &\leq \frac{C}{\Delta t} \|v - v'\|_0 \left[\Delta t + \left(\mathbb{E}_{0,x} \left[\int_{t_0}^{t_1} |\sigma(s, X_s^a) - \sigma(t_0, x)|^2 ds \right] \right)^{\frac{1}{2}} \right] \\ &\leq \frac{C}{\Delta t} \|v - v'\|_0 \left[\Delta t + \left(\mathbb{E}_{0,x} \left[\int_{t_0}^{t_1} (s + |X_s^a - x|^2) ds \right] \right)^{\frac{1}{2}} \right] \\ &\leq \frac{C}{\Delta t} \|v - v'\|_0 \left[\Delta t + \left(\int_{t_0}^{t_1} s ds \right)^{\frac{1}{2}} \right] \leq C \|v - v'\|_0. \end{aligned}$$

By the arbitrariness of (t_k, x, a) we complete the proof. $\blacksquare$

The analysis so far holds true for arbitrary T . However, the result below will be valid only for small T , and we will extend it to general T in the next section.

Fix K and introduce the loss function:

$$J_K(v, w) := \|\Phi_K(w) - v\|_0 + \|\Psi_K(v) - w\|_0, \quad (v, w) \in \mathcal{V}_K \times \mathcal{W}_K. \quad (3.11)$$

Proposition 3.7 implies that

$$J_K(v_{\mathbb{T}_K}^*, w_{\mathbb{T}_K}^*) \leq C_{L^*, T} \sqrt{\Delta t}, \quad \text{and hence} \quad \inf_{(v, w) \in \mathcal{V}_K \times \mathcal{W}_K} J_K(v, w) \leq C_{L^*, T} \sqrt{\Delta t}. \quad (3.12)$$

The problem now boils down to finding a minimizing sequence for J_K . We have the following main result.

Theorem 3.9. *There exist constants $\delta, C > 0$, depending only on $d, \lambda, |A|$, and L_0, L_1 but not on L^* or T , such that for any K and $T \leq \delta$,*

$$\|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0 + \|\Psi_K(v) - w_{\mathbb{T}_K}^*\|_0 \leq C J_K(v, w) + C_{L^*} \sqrt{\Delta t}, \quad (v, w) \in \mathcal{V}_K \times \mathcal{W}_K. \quad (3.13)$$

Proof Fix $\delta > 0$ which will be specified later, and assume without loss of generality that $\delta \leq 1$. Fix K and assume $T \leq \delta$. Applying Proposition 3.7 and Lemma 3.8 we have

$$\begin{aligned}
\|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0 &\leq \|\Phi_K(w) - \Phi_K(w_{\mathbb{T}_K}^*)\|_0 + \|\Phi_K(w_{\mathbb{T}_K}^*) - v_{\mathbb{T}_K}^*\|_0 \\
&\leq C\sqrt{\delta} \|w - w_{\mathbb{T}_K}^*\|_0 + C_{L_*}\sqrt{\Delta t} \\
&\leq C\sqrt{\delta} [\|w - \Psi_K(v)\|_0 + \|v - \Phi_K(w)\|_0 + \|\Psi_K(v) - w_{\mathbb{T}_K}^*\|_0] + C_{L_*}\sqrt{\Delta t} \\
&= C\sqrt{\delta} [J_K(v, w) + \|\Psi_K(v) - w_{\mathbb{T}_K}^*\|_0] + C_{L_*}\sqrt{\Delta t}. \tag{3.14}
\end{aligned}$$

Here we used the obvious fact that the constant $C_{L_*, T}$ in (3.8) is increasing in T and thus we may set $C_{L_*} = C_{L_*, 1}$. Similarly, we also have

$$\begin{aligned}
\|\Psi_K(v) - w_{\mathbb{T}_K}^*\|_0 &\leq \|\Psi_K(v) - \Psi_K(v_{\mathbb{T}_K}^*)\|_0 + \|\Psi_K(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*\|_0 \\
&\leq C \|v - v_{\mathbb{T}_K}^*\|_0 + C_{L_*}\sqrt{\Delta t} \\
&\leq C [J_K(v, w) + \|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0] + C_{L_*}\sqrt{\Delta t}. \tag{3.15}
\end{aligned}$$

Plugging (3.15) into (3.14), we have, for a constant C_0 depending only on the model parameters $d, \lambda, |A|$, and L_0, L_1 in Assumption 3.1,

$$\|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0 \leq C_0\sqrt{\delta} [J_K(v, w) + \|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0] + C_{L_*}\sqrt{\Delta t}. \tag{3.16}$$

Set $\delta := \frac{1}{4C_0^2} \wedge 1$ for the above C_0 . Then,

$$\|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0 \leq \frac{1}{2} [J_K(v, w) + \|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0] + C_{L_*}\sqrt{\Delta t}.$$

Thus

$$\frac{1}{2} \|\Phi_K(w) - v_{\mathbb{T}_K}^*\|_0 \leq \frac{1}{2} J_K(v, w) + C_{L_*}\sqrt{\Delta t}.$$

Substituting this into (3.15) we obtain (3.13) and hence complete the proof. $\blacksquare$

The following is the convergence result.

Theorem 3.10. *Let $T \leq \delta$ for $\delta > 0$ from Theorem 3.9. Then for any $\varepsilon > 0$, there is K (or equivalently Δt) such that J_K has an ε -minimizer $(v^\varepsilon, w^\varepsilon) \in \mathcal{V}_K \times \mathcal{W}_K$ that satisfies*

$$\|\bar{v}^\varepsilon - v_{\mathbb{T}_K}^*\|_0 + \|\bar{w}^\varepsilon - w_{\mathbb{T}_K}^*\|_0 \leq C_{L_*}\varepsilon, \quad \text{where } \bar{v}^\varepsilon := \Psi_K(w^\varepsilon), \bar{w}^\varepsilon := \Phi_K(v^\varepsilon). \tag{3.17}$$

Moreover, for $k = 0, \dots, K$, define

$$\bar{u}^\varepsilon(t_k, x) := \mathbb{E}_{t_k, x} \left[g(\mathcal{X}_T^{\Delta t}) + \sum_{j=k}^{K-1} \lambda \ln \int_A e^{\frac{1}{\lambda} \bar{w}^\varepsilon(t_j, \mathcal{X}_{t_j}^{\Delta t}, a)} da \Delta t \right], \tag{3.18}$$

$$\pi^\varepsilon(t_k, x, a) := \Gamma(t_k, x, \bar{v}^\varepsilon(t_k, x), a).$$

Then, restricting u^*, π^* to $\mathbb{T}_K$, we have

$$\|\bar{u}^\varepsilon - u_{\mathbb{T}_K}^*\|_0 + \|\pi^\varepsilon - \pi_{\mathbb{T}_K}^*\|_0 \leq C_{L^*} \varepsilon. \quad (3.19)$$

Proof Fix $\varepsilon > 0$. Let $\Delta t > 0$ be small enough such that $C_{L^*, T} \sqrt{\Delta t} \leq \frac{\varepsilon}{2}$ for the constant $C_{L^*, T}$ in (3.12). Then $\inf_{(v, w) \in \mathcal{V}_K \times \mathcal{W}_K} J_K(v, w) \leq \frac{\varepsilon}{2}$, and thus J_K has ε -minimizer. Now (3.17) follows directly from (3.13), and (3.19) follows from rather standard estimates. ■

Remark 3.11. In implementation one can use neural networks to approximate the functions v and w , and then apply any specific iterative scheme to find a minimizing sequence of J_K , such as Picard's iteration or the stochastic gradient descent (SGD). In §5.1, we will present an SGD based algorithm. A benefit of the estimate (3.13) is that we can evaluate $J_K(v, w)$ along the iterative steps, which enables us to gauge the error of the current solution to decide when to stop the algorithm. ■

To conclude this section, we briefly discuss the infinite horizon case with time homogeneous coefficients b, σ, r . In this case, the exploratory HJB equation (2.6) becomes elliptic with a certain discount factor $\rho > 0$ (cf. [26]):

$$\rho u^*(x) = \frac{1}{2}[\sigma \sigma^\top](x) : u_{xx}^*(x) + H(x, u_x^*), \quad x \in \mathbb{R}^d. \quad (3.20)$$

With straightforward modifications, our analysis remains valid when ρ is large, which essentially corresponds to a small T . Indeed, by omitting time discretization, we may modify the mappings and loss function with the parameter Δt as follows:

$$\begin{aligned} \Phi(w)(x) &:= \mathbb{E}_{0,x} \left[\int_0^\infty e^{-\rho t} \ln \int_A e^{w(\mathcal{X}_t, a)} da N_t dt \right], \\ \Psi_{\Delta t}(v)(x, a) &:= \frac{1}{\Delta t} \mathbb{E} \left[\int_x^{X_{\Delta t}^a} v(x') dx' - \int_x^{\mathcal{X}_{\Delta t}} v(x') dx' \right] + r(x, a), \end{aligned} \quad (3.21)$$

$$J_{\Delta t}(v, w) := \|\Phi(w) - v\|_0 + \|\Psi_{\Delta t}(v) - w\|_0, \quad v \in C_b^0(\mathbb{R}^d; \mathbb{R}^d), w \in C_b^0(\mathbb{R}^d \times A; \mathbb{R}).$$

Then one can similarly show that,

$$\begin{aligned} J_{\Delta t}(v^*, w^*) &\leq C_\rho \sqrt{\Delta t}, \\ \|\Phi(w) - \Phi(w')\|_0 &\leq \frac{C}{\sqrt{\rho}} \|w - w'\|_0; \quad \|\Psi_{\Delta t}(v) - \Psi_{\Delta t}(v')\|_0 \leq C \|v - v'\|_0, \end{aligned} \quad (3.22)$$

and, for ρ sufficiently large,

$$\|\Phi(w) - v^*\|_0 + \|\Psi_{\Delta t}(v) - w^*\|_0 \leq C J_{\Delta t}(v, w) + C \sqrt{\Delta t}. \quad (3.23)$$

Then, again, the problem boils down to solving the minimization problem $\inf_{v,w} J_{\Delta t}(v, w)$. We leave the details to interested readers. Note that, in the infinite horizon setting, we are able to solve a special case of the problem when the diffusion coefficient depends on control, namely $\sigma = \sigma(x, a)$; see §6.

4 General Time Horizon

In this section we extend the results in the previous section to the case of a general T . For arbitrary $T > 0$, let $\delta > 0$ be determined in Theorem 3.9, which is independent of T . Consider a uniform partition: $0 = T_0 < T_1 < \dots < T_S = T$, with $\Delta T := \frac{T}{S} \leq \delta$ and $T_i := i\Delta T$, $i = 0, \dots, S$. Now, for each sub-interval $[T_i, T_{i+1}]$, we introduce a further uniform time discretization $\mathbb{T}_K^i = \{t_k^i\}_{k=0, \dots, K}$, with $\Delta t := \frac{\Delta T}{K} = \frac{T}{SK}$ and $t_k^i := T_i + k\Delta t$. As in (3.6), denote

$$\mathcal{V}_K^i := C_b^0(\mathbb{T}_K^i \times \mathbb{R}^d; \mathbb{R}^d), \quad \mathcal{W}_K^i := C_b^0(\mathbb{T}_K^i \times \mathbb{R}^d \times A; \mathbb{R}), \quad i = 0, \dots, m-1.$$

Our main idea of treating the arbitrary time horizon is to apply the previous analysis/results to a small period recursively on each sub-interval $[T_i, T_{i+1}]$ in a *backward* manner, starting from $[T_{S-1}, T_S] = [T_{S-1}, T]$. To be precise, denote

$$\bar{v}_S^*(x) := g_S(x), \quad C_S^* := 0, \quad \text{which is equivalent to} \quad \|\bar{v}_S^* - v^*(T_S, \cdot)\|_0 \leq C_S^* \sqrt{\Delta t}, \quad (4.1)$$

where v^* is the true solution. For $i = S-1, \dots, 0$, assume we have constructed $\bar{v}_{i+1}^* \in C_b^0(\mathbb{R}^d; \mathbb{R}^d)$ and there exists a constant $C_{i+1}^* \geq 0$ such that

$$\|\bar{v}_{i+1}^* - v^*(T_{i+1}, \cdot)\|_0 \leq C_{i+1}^* \sqrt{\Delta t}. \quad (4.2)$$

Now as in (3.6) and (3.7) we construct mappings $\Phi_K^i : \mathcal{W}_K^i \mapsto \mathcal{V}_K^i$ and $\Psi_K^i : \mathcal{V}_K^i \mapsto \mathcal{W}_K^i$ by

$$\begin{aligned} \Phi_K^i(w)(t_k^i, x) &:= \mathbb{E}_{t_k^i, x} \left[(\nabla \mathcal{X}_{T_{i+1}}^{\Delta t})^\top \bar{v}_{i+1}^*(\mathcal{X}_{T_{i+1}}^{\Delta t}) + \sum_{j=k+1}^{K-1} \lambda \ln \int_A e^{\frac{1}{\lambda} w(t_j^i, \mathcal{X}_{t_j^i}^{\Delta t}, a)} da N_{t_j^i}^{\Delta t} \Delta t \right]; \\ \Psi_K^i(v)(t_k^i, x, a) &:= \frac{1}{\Delta t} \mathbb{E}_{t_k^i, x} \left[\int_x^{X_{t_{k+1}^i}^a} v(t_k^i, x') dx' - \int_x^{\mathcal{X}_{t_{k+1}^i}^{\Delta t}} v(t_k^i, x') dx' \right] + r(t_k^i, x, a), \end{aligned} \quad (4.3)$$

and define the loss function as in (3.11) by:

$$J_K^i(v, w) := \|\Phi_K^i(w) - v\|_0 + \|\Psi_K^i(v) - w\|_0, \quad (v, w) \in \mathcal{V}_K^i \times \mathcal{W}_K^i. \quad (4.4)$$

Then we have the following result.

Theorem 4.1. *Consider the above setting where (4.2) holds. Then*

$$\|\Phi_K^i(w_{\mathbb{T}_K}^*) - v_{\mathbb{T}_K}^*\|_0 + \|\Psi_K^i(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*\|_0 \leq C_{L_*} [1 + C_{i+1}^*] \sqrt{\Delta t}, \quad (4.5)$$

$$\left\| \Phi_K^i(w) - v_{\mathbb{T}_K}^* \right\|_0 + \left\| \Psi_K^i(v) - w_{\mathbb{T}_K}^* \right\|_0 \leq C J_K^i(v, w) + C_{L_*} [1 + C_{i+1}^*] \sqrt{\Delta t}, \quad (4.6)$$

for all i and all $(v, w) \in \mathcal{V}_K^i \times \mathcal{W}_K^i$.

Proof Denote

$$\tilde{\Phi}_K^i(w)(t_k^i, x) := \mathbb{E}_{t_k^i, x} \left[(\nabla \mathcal{X}_{T_{i+1}}^{\Delta t})^\top v^*(T_{i+1}, \mathcal{X}_{T_{i+1}}^{\Delta t}) + \sum_{j=k+1}^{K-1} \lambda \ln \int_A e^{\frac{1}{\lambda} w(t_j^i, \mathcal{X}_{t_j^i}^{\Delta t}, a)} da N_{t_j^i}^{\Delta t} \Delta t \right].$$

Then, since $\Delta T \leq \delta \leq 1$, it follows from Proposition 3.7 that

$$\|\tilde{\Phi}_K^i(w_{\mathbb{T}_K}^*) - v_{\mathbb{T}_K}^*\|_0 + \|\Psi_K^i(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*\|_0 \leq C_{L_*} \sqrt{\Delta t}.$$

Moreover, for any (t_k^i, x) , by Lemma 3.5 and (4.2) we have

$$|\tilde{\Phi}_K^i(w)(t_k^i, x) - \Phi_K^i(w)(t_k^i, x)| \leq \mathbb{E}_{t_k^i, x} [|\nabla \mathcal{X}_{T_{i+1}}^{\Delta t}|] \|\bar{v}_{i+1}^* - v^*(T_{i+1}, \cdot)\|_0 \leq C_{L_*} C_{i+1}^* \sqrt{\Delta t}.$$

Combining the above two estimates, we obtain (4.5) immediately.

Next, Lemma 3.8 remains true for the mappings Φ_K^i, Ψ_K^i . That is,

$$\|\Phi_K^i(w) - \Phi_K^i(w')\|_0 \leq C\sqrt{\delta} \|w - w'\|_0, \quad \|\Psi_K^i(v) - \Psi_K^i(v')\|_0 \leq C \|v - v'\|_0,$$

for all $(v, w), (v', w') \in \mathcal{V}_K^i \times \mathcal{W}_K^i$. Combining this with (4.5), we derive (4.6) following the same arguments as in Theorem 3.9. $\blacksquare$

Now set $\varepsilon := 2C_{L_*} [1 + C_{i+1}^*] \sqrt{\Delta t}$ with the constant C_{L_*} in (4.5). Applying the arguments in proving Theorem 3.10, we obtain an ε -minimizer $(v_i^\varepsilon, w_i^\varepsilon) \in \mathcal{V}_K^i \times \mathcal{W}_K^i$ of J_K^i . Denoting $\bar{v}_i^\varepsilon := \Phi_K^i(w_i^\varepsilon)$, $\bar{w}_i^\varepsilon := \Psi_K^i(v_i^\varepsilon)$, we have

$$\|\bar{v}_i^\varepsilon - v_{\mathbb{T}_K}^*\|_0 + \|\bar{w}_i^\varepsilon - w_{\mathbb{T}_K}^*\|_0 \leq C_{L_*} \varepsilon = 2|C_{L_*}|^2 [1 + C_{i+1}^*] \sqrt{\Delta t}. \quad (4.7)$$

Set

$$\bar{v}_i^*(x) := \bar{v}_i^\varepsilon(t_0^i, x) = \bar{v}^\varepsilon(T_i, x), \quad C_i^* := 2|C_{L_*}|^2 [1 + C_{i+1}^*]. \quad (4.8)$$

We conclude that $\bar{v}_i^\varepsilon$ and C_i^* satisfy (4.2) at i , thus completing the induction step.

We are now ready to state the main result of this section.

Theorem 4.2. Let $\delta, T_i, \mathbb{T}_K^i, \varepsilon$ be given and $v_i^\varepsilon, w_i^\varepsilon, \bar{v}_i^\varepsilon, \bar{w}_i^\varepsilon, v_i^*, C_i^*, J_K^i$ be constructed as above, where in particular $(v_i^\varepsilon, w_i^\varepsilon)$ is an ε -minimizer of J_K^i . Moreover, define $\bar{u}_S^*(x) := g(x)$, and for $i = S-1, \dots, 0$, and for all k ,

$$\bar{u}_i^\varepsilon(t_k^i, x) := \mathbb{E}_{t_k^i, x} \left[\bar{u}_{i+1}^*(\mathcal{X}_{T_{i+1}}^{\Delta t}) + \sum_{j=k}^{K-1} \lambda \ln \int_A e^{\frac{1}{\lambda} \bar{w}_i^\varepsilon(t_j^i, \mathcal{X}_{t_j^i}^{\Delta t}, a)} da \Delta t \right], \quad \bar{u}_i^*(x) := \bar{u}_i^\varepsilon(t_0^i, x); \quad (4.9)$$

$$\pi_i^\varepsilon(t_k^i, x, a) := \Gamma(t_k^i, x, \bar{v}_i^\varepsilon(t_k^i, x), a).$$

Then, there exists $C_{L^*} > 0$ such that, for $i = 0, \dots, m-1$,

$$\begin{aligned} \|\bar{v}_i^\varepsilon - v_{\mathbb{T}_K^i}^*\|_0 + \|\bar{w}_i^\varepsilon - w_{\mathbb{T}_K^i}^*\|_0 &\leq C_i^* \sqrt{\Delta t}; \\ \|\bar{u}_i^\varepsilon - u_{\mathbb{T}_K^i}^*\|_0 &\leq 2C_i^* \sqrt{\Delta t}, \quad \|\pi_i^\varepsilon - \pi_{\mathbb{T}_K^i}^*\|_0 \leq C_{L^*} C_i^* \sqrt{\Delta t}. \end{aligned} \quad (4.10)$$

Proof The estimates for $\bar{v}_i^\varepsilon, \bar{w}_i^\varepsilon$ are already given in (4.7). To estimate $\bar{u}_i^\varepsilon$, first note that $\|\bar{u}_n^* - u^*(T_S, \cdot)\|_0 = 0 = 2C_S^* \sqrt{\Delta t}$. Assume we have proved $\|\bar{u}_{i+1}^* - u^*(T_{i+1}, \cdot)\|_0 \leq 2C_{i+1}^* \sqrt{\Delta t}$. Recall (2.13) and note that, for $k = 0, \dots, K-1$,

$$u^*(t_k^i, x) = \mathbb{E}_{t_k^i, x} \left\{ u^*(T_{i+1}, \mathcal{X}_{T_{i+1}}) + \int_{t_k^i}^{T_{i+1}} \lambda \ln \left[\int_A \exp \left(\frac{1}{\lambda} w^*(s, \mathcal{X}_s, a) \right) da \right] ds \right\}.$$

Then, by (4.9) and (3.10) we have, for the function H^* defined in (8.20) in Appendix,

$$\begin{aligned} |\bar{u}_i^\varepsilon(t_k^i, x) - u^*(t_k^i, x)| &\leq \|\bar{u}_{i+1}^* - u^*(T_{i+1}, \cdot)\|_0 + \sum_{j=k}^{K-1} \|\bar{w}_i^\varepsilon - w_{\mathbb{T}_K^i}^*\|_0 \Delta t + \sum_{j=k}^{K-1} \int_{t_j^i}^{t_{j+1}^i} I_s^j ds, \\ \text{where } I_s^j &:= \mathbb{E}_{t_k^i, x} [|H^*(s, \mathcal{X}_s) - H^*(t_j^i, \mathcal{X}_{t_j^i}^{\Delta t})|]. \end{aligned}$$

It follows from the regularity of H^* ; see (8.22) in Appendix, along with Lemma 3.5, that

$$|I_s^j| \leq C_{L^*} \mathbb{E}_{t_k^i, x} [\sqrt{s - t_j^i} + |\mathcal{X}_s - \mathcal{X}_{t_j^i}| + |\mathcal{X}_{t_j^i} - \mathcal{X}_{t_j^i}^{\Delta t}|] \leq C_{L^*} \sqrt{\Delta t}.$$

Then, noting $K\Delta t = \Delta T \leq \delta \leq 1$, we get

$$|\bar{u}_i^\varepsilon(t_k^i, x) - u^*(t_k^i, x)| \leq 2C_{i+1}^* \sqrt{\Delta t} + [C_i^* \sqrt{\Delta t} + C_{L^*} \sqrt{\Delta t}] K\Delta t \leq 2C_i^* \sqrt{\Delta t}.$$

In particular,

$$\|\bar{u}_i^* - u^*(T_i, \cdot)\|_0 = \sup_x |\bar{u}_i^\varepsilon(t_0^i, x) - u^*(t_0^i, x)| \leq 2C_i^* \sqrt{\Delta t}.$$

This completes the induction step and thus we obtain the estimate for $\bar{u}_i^\varepsilon$.

Finally, recall (2.7) and note that

$$\begin{aligned} \partial_z \gamma(t, x, z, a) &:= \frac{1}{\lambda} b(t, x, a) \gamma(t, x, z, a), \\ \partial_z \Gamma(t, x, z, a) &:= \frac{\frac{1}{\lambda} b(t, x, a) \gamma(t, x, z, a)}{\int_A \gamma(t, x, z, a') da'} - \frac{\gamma(t, x, z, a) \int_A \frac{1}{\lambda} b(t, x, a') \gamma(t, x, z, a') da'}{(\int_A \gamma(t, x, z, a') da')^2}. \end{aligned}$$

Then,

$$|\partial_z \Gamma(t, x, z, a)| \leq \frac{C\gamma(t, x, z, a)}{\int_A \gamma(t, x, z, a') da'} + \frac{C\gamma(t, x, z, a) \int_A \gamma(t, x, z, a') da'}{(\int_A \gamma(t, x, z, a') da')^2} = C\Gamma(t, x, z, a).$$

Moreover, noting $|v^*| \leq C_{L_*}$ as well as the estimate of $\bar{v}_i^\varepsilon$ we have

$$|\bar{v}_i^\varepsilon(t_k^i, x)| \leq |v^*(t_k^i, x)| + C_i^* \sqrt{\Delta t} \leq C_{L_*} + 1.$$

Here we assume without loss of generality that Δt is small enough so that $C_i^* \sqrt{\Delta t} \leq 1$. Then by Assumption 3.1 and (2.7) we can easily see that $\Gamma(t, x, z, a) \leq C_{L_*}$ for all z between $\bar{v}_i^\varepsilon(t_k^i, x)$ and $v^*(t_k^i, x)$. Thus

$$\begin{aligned} |\pi_i^\varepsilon(t_k^i, x, a) - \pi^*(t_k^i, x, a)| &= \left| \Gamma(t_k^i, x, \bar{v}_i^\varepsilon(t_k^i, x), a) - \Gamma(t_k^i, x, v^*(t_k^i, x), a) \right| \\ &\leq C_{L_*} |\bar{v}_i^\varepsilon(t_k^i, x) - v^*(t_k^i, x)| \leq C_{L_*} C_i^* \sqrt{\Delta t}. \end{aligned}$$

This completes the proof. ■

Remark 4.3. Recall that the constant $\delta > 0$ does not depend on T or L_* . For a general time horizon $T > 0$, the partition number S is determined by $\frac{T}{\delta}$. So when T is very large and/or when δ is very small (when L_0, L_1 are large), S will be large as well, so will C_i^* . Indeed, from (4.8) it follows that $C_0^* \sim (2|C_{L_*}|^2)^S \sim (2|C_{L_*}|^2)^{\frac{T}{\delta}}$. In other words, theoretically our scheme may become inefficient when T is really large. Nevertheless, in the next section we will show numerically that for reasonably large T or S this multi-step procedure is indeed efficient while the one-step method in the previous section simply fails. ■

5 Numerical Experiments

In this section, we first present our numerical algorithms, including a single-step algorithm for a small time horizon T and a multi-time-period algorithm for a general T , along with numerical examples in both cases, including high-dimensional ones. Finally, we give comparisons with some well-known existing algorithms.

5.1 Algorithm for small time horizon

Before presenting the full numerical scheme in Algorithm 1, we clarify several key points.

First, the ultimate goal of the algorithm is to learn the optimal value function u^* (which solves the exploratory HJB) and the optimal exploratory policy π^* . In view of (2.13) and

(2.7), it suffices to learn the functions v^* and w^* . In the algorithm, we use two neural networks, v^ψ and w^ϕ to approximate v^* and w^* respectively.

Next, the algorithm takes a batch of M state/reference trajectories, each starting from the initial time $t_0 = 0$ and an initial state x_0 randomly drawn from a distribution $\text{Law}(X_0)$, chosen based on the specific objective of the task at hand. If the objective is to solve the exploratory HJB and/or obtain the optimal feedback policy, then a typical choice of $\text{Law}(X_0)$ is a uniform distribution supported on a bounded region (that is likely to contain the most states of interest). If one aims to just learn the optimal value $u^*(0, x_0)$ with respect to a specific, given initial state x_0 , then the distribution can be just the Dirac measure centered at that initial state x_0 .

In the procedure we need to apply a constant control a drawn uniformly from the action space A . In general when A is an arbitrary measurable set with a non-zero and finite volume (which is assumed in this paper), there is a so-called hit-and-run algorithm that can sample uniformly from A ; see [29, 2]. Alternatively, especially when A has a more regular shape (e.g. an orthotope), we can use a sufficiently fine uniform grid $\{a_j\}_{j=1}^{N_a}$ to approximate A , where N_a denotes the number of discrete points. Then we can sample actions uniformly just from these grid points. This method, which we will employ in our subsequent numerical experiments, has an added benefit: it also provides the quadrature nodes required to numerically evaluate the integral in the Φ mapping. Aligning the sampling grid with the integration grid ensures consistency across iterations and simplifies the evaluation of the policy.

In view of Remark 2.1-(v), we need to carefully manage the independent Brownian increments used to evaluate the mappings Φ and Ψ . The mapping Φ in (4.3) is evaluated along an *entire* reference path $\mathcal{X}^{(m)}$ driven by the increments $B_k^{(m)}$; see Algorithm 1, line 10-13. By contrast, the mapping Ψ , at each node $\mathcal{X}_k^{(m)}$, requires *one-step* transitions of both the controlled process $X_{k+1}^{a,(m,n)}$ and the reference path $\mathcal{X}_{k+1}^{(m,n)}$. In practice, the former is observed from the environment upon a query (see Remark 2.1-(i),(ii)) while the latter is simulated using $\tilde{B}_k^{(m,n)}$ that is independent from $B_k^{(m)}$; see Algorithm 1, lines 17-20. In our numerical experiments below, we simulate the environment using Brownian increments independent of those driving the controlled state or reference processes. We note that, in practical applications, the overall cost of the algorithm also includes that of acquiring the controlled-transition data $X_{k+1}^{a,(m,n)}$.

Finally, we formalize the function approximation and the empirical loss. Recall we need to learn two networks v^ψ and w^ϕ . The theoretical objective relies on the supremum norm, which is well-suited for convergence analysis but impractical for gradient-based optimiza-

tion due to its non-smoothness. Therefore, we approximate the supremum norm using a differentiable softmax surrogate. Given a parameter $\tau > 0$, for any vector l of length K , we define:

$$\text{softmax}_\tau(l) = \tau \log \sum_{i=1}^K \exp(l_i/\tau).$$

It is a standard result that $\text{softmax}_\tau(l)$ converges to the supremum norm of l as $\tau \rightarrow 0$.⁷

To construct the empirical loss function, we first compute the residuals on each trajectory (m) at each time step t_k :

$$\begin{aligned} l_k^{v,(m)} &= |v^\psi(t_k, \mathcal{X}_k^{(m)}) - \Phi_k^{(m)}|, \\ l_k^{w,(m)} &= |w^\phi(t_k, \mathcal{X}_k^{(m)}, a^{(m)}) - \Psi_k^{(m)}|. \end{aligned}$$

Applying the softmax approximation, we aggregate these residuals over the time grid (from $k = 0$ to $K - 2$) to obtain the trajectory-level losses:

$$L_v^{(m)} = \text{softmax}_\tau \left(\{l_k^{v,(m)}\}_{k=0}^{K-2} \right), \quad L_w^{(m)} = \text{softmax}_\tau \left(\{l_k^{w,(m)}\}_{k=0}^{K-2} \right).$$

Averaging across the batch of size M , the total empirical loss J , which proxies the theoretical supremum norm objective, is given by:

$$J = \frac{1}{M} \sum_{m=1}^M \left(L_v^{(m)} + L_w^{(m)} \right).$$

We update the parameters ψ and ϕ to minimize J via stochastic gradient descent with learning rates η_v and η_w respectively.

We now present our first algorithm, Algorithm 1. For ease of presentation, we write the pseudocode for $d = 1$. Modifications are straightforward for $d > 1$. Take the expression of $\nabla \mathcal{X}$ for example, it will be a matrix-valued Jacobian initialized at I_d . Its scalar exponential update will be replaced by a discretization of the corresponding matrix variational SDE. Moreover, the integrand involved will become $(\sigma^{-1} \nabla \mathcal{X})^\top dB$, while the terminal derivative term will be $(\nabla \mathcal{X})^\top \nabla g$.

Algorithm 1 (Single-time-period algorithm)

- 1: **Input:** Fixed initial time $t_0 = 0$, terminal time T , number of grid points K , batch size M , nested path size N , and epochs E ; action space A , temperature parameter $\lambda > 0$, softmax parameter τ , and learning rates η_v, η_w ; distribution of the initial state $\text{Law}(X_0)$.
-

⁷Another natural approximation of the max norm is the $\mathbb{L}^p$ norm with a relatively large p . However, when p becomes large, computations become noticeably heavier in both time and memory, as observed in our preliminary experiments (not reported in the paper). Specifically, $p \geq 8$ introduces significant slowdowns even in the one-dimensional case.

2: **Output:** Trained networks (v^ψ, w^ϕ) and the resulting estimated optimal value $\hat{u}$ and policy $\hat{\pi}$.

3: **Initialize:**

Set

$$\Delta t \leftarrow \frac{T - t_0}{K - 1}, \quad t_k \leftarrow t_0 + k\Delta t, \quad k = 0, \dots, K - 1,$$

and randomly initialize the network weights (ψ, ϕ) for $v^\psi(t, x)$ and $w^\phi(t, x, a)$.

4: **for** epoch = 1, ..., E **do**

5: **for** $m = 1, \dots, M$ **do**

▷ Batch generation

6: Draw $\mathcal{X}_0^{(m)} \sim \text{Law}(X_0)$.

7: Draw primary Brownian increments

$$\{\Delta B_k^{(m)}\}_{k=0}^{K-2} \sim \mathcal{N}(0, \Delta t).$$

8: Set

$$\nabla \mathcal{X}_0^{(m)} \leftarrow 1, \quad N_0^{(m)} \leftarrow 0.$$

9: **for** $k = 0$ **to** $K - 2$ **do**

▷ Forward simulation of the primary reference path

10: $\mathcal{X}_{k+1}^{(m)} \leftarrow \mathcal{X}_k^{(m)} + \sigma(t_k, \mathcal{X}_k^{(m)})\Delta B_k^{(m)}$.

11: $\nabla \mathcal{X}_{k+1}^{(m)} \leftarrow \nabla \mathcal{X}_k^{(m)} \exp\left(\sigma_x(t_k, \mathcal{X}_k^{(m)})\Delta B_k^{(m)} - \frac{1}{2}\sigma_x^2(t_k, \mathcal{X}_k^{(m)})\Delta t\right)$.

12: $N_{k+1}^{(m)} \leftarrow N_k^{(m)} + \nabla \mathcal{X}_k^{(m)} \sigma^{-1}(t_k, \mathcal{X}_k^{(m)})\Delta B_k^{(m)}$.

13: **for** $k = 0$ **to** $K - 2$ **do**

▷ Compute Φ - and Ψ -residuals

14: Compute $\Phi_k^{(m)}$ using the primary-path data $\left(k, \{\mathcal{X}_i^{(m)}\}, \{N_i^{(m)}\}, \{\nabla \mathcal{X}_i^{(m)}\}; \phi\right)$.

15: $l_k^{v, (m)} \leftarrow \left|v^\psi(t_k, \mathcal{X}_k^{(m)}) - \Phi_k^{(m)}\right|$.

16: **for** $n = 1, \dots, N$ **do**

▷ Nested paths from $(t_k, \mathcal{X}_k^{(m)})$

17: Draw $\Delta \tilde{B}_k^{(m, n)} \sim \mathcal{N}(0, \Delta t)$.

18: Simulate the nested reference transition

$$\mathcal{X}_{k+1}^{(m, n)} \leftarrow \mathcal{X}_k^{(m)} + \sigma(t_k, \mathcal{X}_k^{(m)})\Delta \tilde{B}_k^{(m, n)}.$$

19: Sample $a^{(n)} \sim \text{Unif}(A)$.

20: Apply the fixed action $a^{(n)}$ at $(t_k, \mathcal{X}_k^{(m)})$ and observe $X_{k+1}^{a^{(n)}, (m, n)}$.

21: Compute $\Psi_k^{(m)}$ using the nested data $\left(k, \mathcal{X}_k^{(m)}, a^{(n)}, \{X_{k+1}^{a^{(n)}, (m, n)}\}, \{\mathcal{X}_{k+1}^{(m, n)}\}; \psi\right)$.

22: $l_k^{w, (m)} \leftarrow \left|w^\phi(t_k, \mathcal{X}_k^{(m)}, a^{(n)}) - \Psi_k^{(m)}\right|$.

23: $L_v^{(m)} \leftarrow \text{softmax}_\tau\left(\{l_k^{v, (m)}\}_{k=0}^{K-2}\right)$.

24: $L_w^{(m)} \leftarrow \text{softmax}_\tau\left(\{l_k^{w, (m)}\}_{k=0}^{K-2}\right)$.

25: Compute the total empirical loss

$$J \leftarrow \frac{1}{M} \sum_{m=1}^M \left(L_v^{(m)} + L_w^{(m)}\right).$$

26: Update

$$\psi \leftarrow \psi - \eta_v \nabla_\psi J, \quad \phi \leftarrow \phi - \eta_w \nabla_\phi J.$$

- 27: Obtain $\hat{\pi}(t, x, a)$ from $w^\phi(t, x, a)$ through (2.7).
 28: Recover $\hat{u}$ from $w^\phi(t, x, a)$ using independent paths through (2.13).
 29: **Return:** $(v^\psi, w^\phi, \hat{u}, \hat{\pi})$.
-

We now present numerical results for applying Algorithm 1. Unless otherwise stated, all the test results reported are based on 50 independent runs on a local machine with an NVIDIA GeForce RTX 3080 Ti GPU (12 GB GDDR6X, 384-bit memory interface). For detailed neural network specifications, hyperparameters, and the complete codebase, please refer to our GitHub repository <https://github.com/GaozhanWang/MWZZ-RL-2026>.

We start with two one-dimensional stochastic control examples.

Example 5.1. Consider a control problem with the following coefficients and control set:

$$b(t, x, a) = x + a, \quad \sigma(t, x) = 1, \quad a \in [0, 1],$$

along with the reward functions

$$r(t, x, a) = (2t + 2ax)e^{-(t^2+x^2+1)}, \quad g(x) = e^{-(T^2+x^2-1)}.$$

Given those coefficients, setting $\lambda = 1$, we can solve the exploratory HJB equation (2.6) analytically to get the *ground-truth* optimal value function as well as the corresponding optimal policy

$$u^*(t, x) = e^{-(t^2+x^2+1)}; \quad \pi^*(t, x, a) = \frac{1}{|A|} = 1, \quad a \in [0, 1].$$

Now we pretend that we do not know the analytical forms of b and r , and apply Algorithm 1 to solve the problem numerically. First consider the problem starting from $(t_0, x_0) = (0, 0.1)$, in which case we choose $\text{Law}(X_0) = \text{Dirac}(x_0)$. In implementation, we set $T = 0.4, \Delta t = 0.02$. Denote the relative error to be

$$RE = \left| \frac{u^*(t_0, x_0) - \hat{u}(t_0, x_0)}{u^*(t_0, x_0)} \right|,$$

where $u^*(t_0, x_0), \hat{u}(t_0, x_0)$ are the true (initial) value and the learned one respectively. This error is motivated by a scenario where one is concerned with only the optimal value corresponding to the particular initial pair (t_0, x_0) .

Time horizon	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
$T=0.4$	0.3642	0.3546	2.6397%	0.4459%	38.12

Table 1: Numerical results with Algorithm 1 on Example 5.1.

Applying Algorithm 1 with the choice of $\text{Law}(X_0) = \text{Dirac}(x_0)$, we not only can learn the particular value $\hat{u}(t_0, x_0)$ but also the *entire* optimal value function $\hat{u}$ along with the optimal policy $\hat{\pi}$. At below we provide visualizations of the thus learned optimal value function and optimal policy versus the oracle ones in the following two figures. To be more precise, Figure 1 shows the learned value function surface and its relative error with respect to the oracle solution.⁸ The three plots in Figure 2 compare the learned policy $\hat{\pi}(t, \cdot, \cdot)$ with the true optimal policy $\pi^*(t, \cdot, \cdot)$ at the time slices $t = 0, 0.2, 0.4$, respectively.

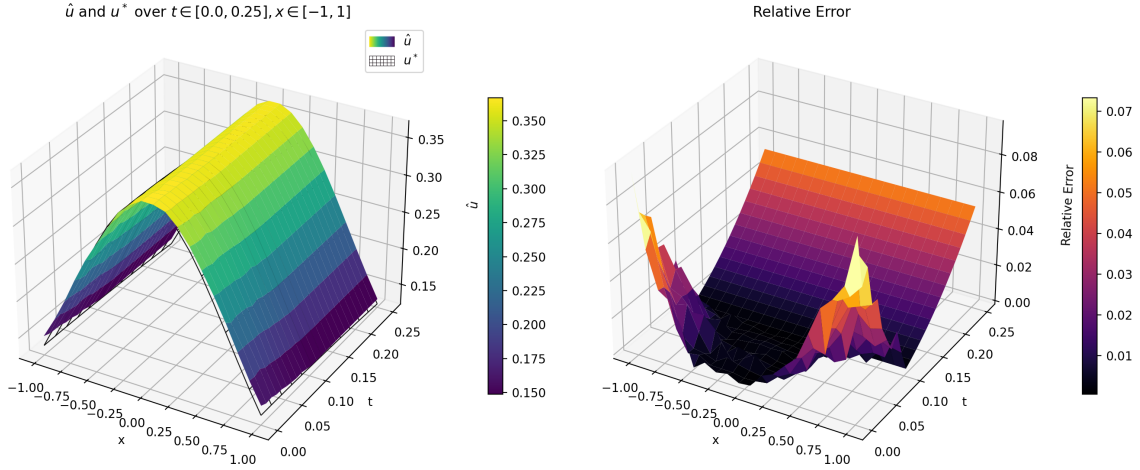


Figure 1: $\hat{u}$ vs u^*

⁸Due to our GPU limit, we display only the portion of the surface over $t \in [0, 0.3]$ for illustration. All the figures in this paper are provided in vector format; hence although scaled down to fit the page layout, they can be enlarged without loss of resolution to inspect finer details.

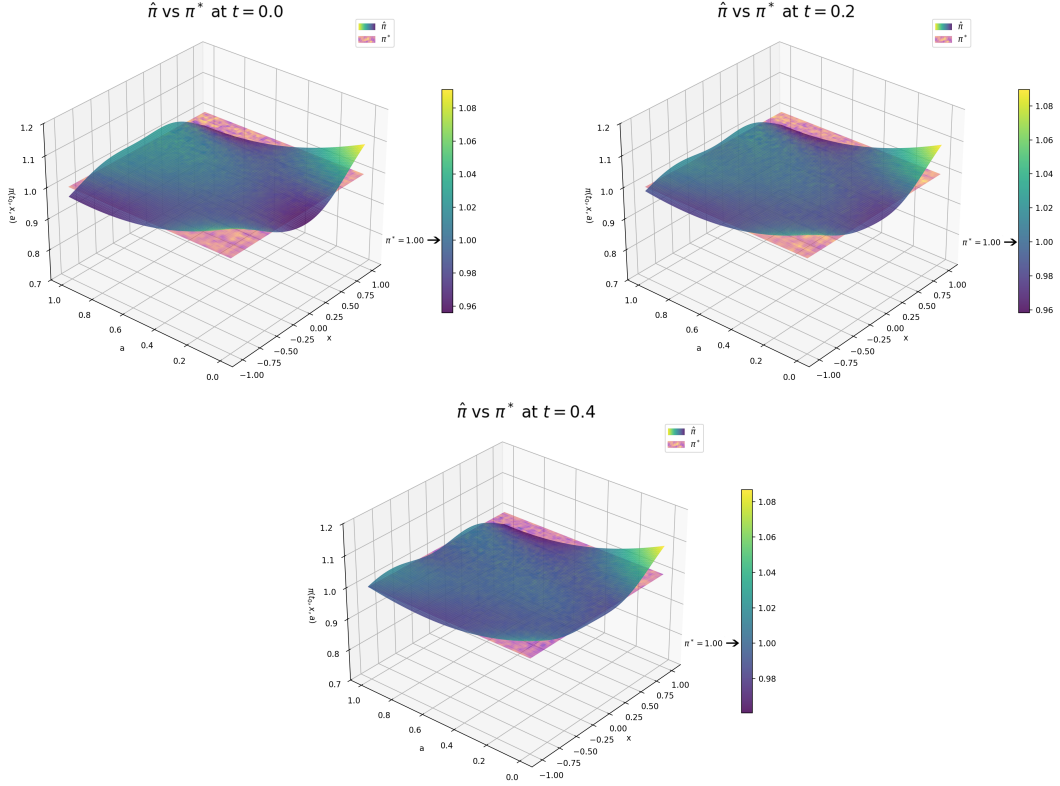


Figure 2: $\hat{\pi}$ vs π^*

We observe in Figure 1 that the relative errors in the optimal value function remain small in a neighborhood of the given initial point, $(t_0, x_0) = (0.0, 0.1)$. As state variable x moves away from this initial condition x_0 , the errors gradually increase. This behavior is natural because in this example $\hat{u}$ is learned based on state/reference trajectories all initialized at $(t_0, x_0) = (0.0, 0.1)$. Consequently, the training data are concentrated along trajectories emanating from this point, leading to higher accuracy in its vicinity and weaker generalization in more distant regions. For applications where the control problem is strictly tied to a specific initial time and state, this localized sampling approach provides a highly computationally efficient solution and avoids the unnecessary overhead of exploring the broader state space.

If the goal of the learning is to obtain the *global* optimal value function (or equivalently to solve the expository HJB) and/or the optimal feedback policy, then we need to randomize the initial state x_0 starting from $t_0 = 0$ according to some non-Dirac distribution $\text{Law}(X_0)$ for generating training trajectories/data as discussed earlier. For illustration and simplicity, we take $\text{Law}(X_0)$ to be a uniform distribution over an interval $[x_0^{\min}, x_0^{\max}] = [-1, 1]$ for the

current example. The corresponding test results (run with a new set of independent random seeds) are presented in Figure 3.

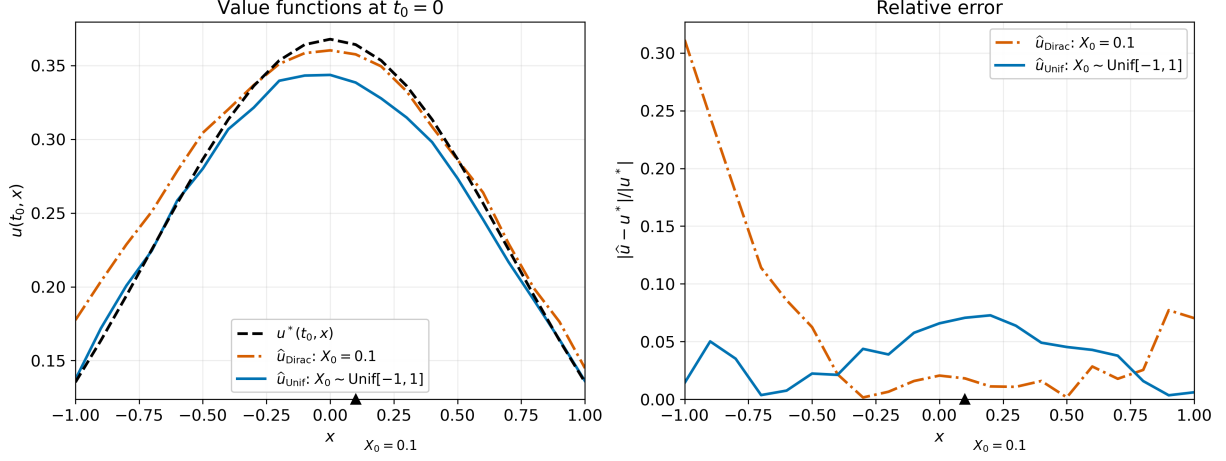


Figure 3: $\hat{u}(0, \cdot)$ vs $u^*(0, \cdot)$

While randomizing initial state naturally demands a higher computational cost during training, it successfully mitigates the local concentration of trajectories. Indeed, Figure 3 shows that the uniform accuracy of the approximation is now significantly improved across the target state space.

Example 5.2. The next example involves trigonometric type functions with a non-constant volatility. Specifically,

$$b(t, x, a) = -2 \cos(t + x) - \frac{1}{4} \sin(2t + 2x) - 1 + a, \quad \sigma(t, x) = 2 + \sin(t + x), \quad a \in [0, 1],$$

with

$$r(t, x, a) = a \sin(t + x) + 2 \cos(t + x), \quad g(x) = \cos(T + x).$$

Setting $\lambda = 1$, the theoretical optimal value function is $u^*(t, x) = \cos(t + x)$. We set $(t_0, x_0) = (0, 0)$, $T = 0.1$, $\Delta t = 0.02$, and obtain the results below:

Time horizon	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
$T=0.1$	1.0000	0.9737	2.6222%	0.0423%	29.15

Table 2: Numerical results with Algorithm 1 on Example 5.2.

Note that in this example as both the model parameters and the truth value functions are trigonometric, we deliberately avoid using `tanh` as the neural network activation function—even though it is common—in order to test the generality of our algorithm’s behavior.

We now use the previous two examples to compare Algorithm 1 with the classical model-based method which assumes all the model parameters to be known and applies PIA to directly approximate the optimal value function. Importantly, as discussed earlier PIA takes auto-diff to directly calculate the gradient of the approximated value function across iterations.

Example 5.3. (PIA Comparison) We find that the model-based PIA is numerically unstable: with GPU acceleration, it either terminates quickly in a few seconds but stops at a poor estimation, or it fails to converge within a reasonable runtime. In the tables below, we only report results with “converged” values and, in the latter case, we cap the policy iteration count at 1000 to prevent endless runs. As a result, we do not report runtimes here.

Tables 3 and 4 indicate the poor performance of the PIA scheme, which generally does not converge to the true solution.

Time Horizon	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE
T=0.4	0.3642	2.191	495.6%	6.5301%

Table 3: Numerical results with PIA for Example 5.1.

Time Horizon	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE
T=0.1	1.000	1.466	46.58%	1.4452%

Table 4: Numerical results with PIA for Example 5.2.

Example 5.4. We now extend our numerical experiment to multi-dimensional cases. Take a scaled version of Example 5.1, with the dynamics and the true value function given as

$$b_i(t, \vec{x}, a) = \vec{x}_i + a_i, \quad \sigma(t, x) = \mathbb{I}_d, \quad a \in [0, 1]^d, \quad i = 1, \dots, d,$$

$$u^*(t, \vec{x}) = e^{-(t^2 + \|\vec{x}\|_2^2/d+1)}, \quad \vec{x} \in \mathbb{R}^d,$$

along with the reward functions

$$r(t, \vec{x}, a) = (2t + 2a^\top \vec{x})e^{-(t^2 + \|\vec{x}\|_2^2/d+1)}, \quad g(\vec{x}) = e^{-(T^2 + \|\vec{x}\|_2^2/d-1)}.$$

We test Algorithm 1 on dimensions $d = 1, 5, 10, 20, 50, 100$ with $\vec{x}_0 = (0.1, 0.1, \dots, 0.1)^\top \in \mathbb{R}^d$, $T = 0.1$, and report the results in Table 5.⁹

⁹Algorithm 1 is written for $d = 1$, but see the remarks immediately before the algorithm pseudocode

Dimension	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
$d=1$	0.3642	0.3678	1.0021%	0.1476%	10.41
$d=5$	0.3642	0.3351	7.9830%	0.0731%	10.75
$d=10$	0.3642	0.3309	9.1297%	0.2205%	18.29
$d=20$	0.3642	0.3285	9.8023%	0.1234%	36.88
$d=50$	0.3642	0.3274	10.1823%	0.0828%	103.31
$d=100$	0.3642	0.3266	10.3283%	0.0057%	17201.05

Table 5: Numerical results for Algorithm 1 on Example 5.4.

We now compare the above result with a state-of-the-art benchmark – a *model-based* method for high-dimensional PDEs in [15]. Under the same setting of Example 5.4 but assuming oracle knowledge of the model parameters, we apply the algorithm in [15] to obtain results in the following table.

Dimension	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
$d=5$	0.3642	0.3443	5.4619%	0.0314%	4.84
$d=10$	0.3642	0.3317	8.9098%	0.0371%	6.36
$d=20$	0.3642	0.3308	9.1290%	0.0138%	19.75
$d=50$	0.3642	0.3295	9.5215%	0.0149%	40.55
$d=100$	0.3642	0.3280	9.9925%	0.0880%	64.66

Table 6: Numerical results with algorithm in [15] for Example 5.4.

A comparison of the results in Tables 5 and 6 is in order. First of all, due to the missing key model coefficients our algorithm needs to carry out a significantly larger number of sampling and computations that inevitably increase the runtime. However, in relatively lower dimensions ($d \leq 50$), our algorithm, aided by GPU acceleration, achieves near-identical accuracy to the model-based approach within a reasonable computational timeframe (about 2-3 times longer). Crucially, unlike CPU execution, where computational time scales roughly linearly with sample size and dimension, GPU parallelization allows for sub-linear time scaling, maintaining high speed until hardware limits are reached. As the dimension increases to $d = 100$, we encounter exactly this hardware bottleneck under our current local computation for an explanation on multi-dimensional cases. Meanwhile, the dramatic runtime jump from dimension 50 to 100 is chiefly a hardware effect—our local GPU saturates its parallel resources (memory-bandwidth limits)—rather than a sudden increase in statistical or algorithmic complexity. Due to the heavy runtime, the results for the 100-dimensional case are based on only 2 independent runs.

ing power. While we anticipate that our algorithm can maintain comparable accuracy and execution speed given abundant GPU resources, it is important to explicitly acknowledge its inherent computational demands. Specifically, for any fixed dimension, our approach requires a significantly larger GPU memory footprint than the model-based baseline.

5.2 Algorithm for general time horizon

We now extend Algorithm 1 to the general setting when T is arbitrary. For all the previously reported examples, we will show that the performance of Algorithm 1 deteriorates rapidly as T increases (see Example 5.5 below). In particular, it happens when $T > 0.4$ in Example 5.1 and $T > 0.1$ in Example 5.2.

We first present the following (multi-period) algorithm based on our analysis in §4 for a general time horizon T .

Algorithm 2 (Multi-time-period algorithm)

1: **Input:** Fixed initial time $t_0 = 0$ and segment partition

$$t_0 = T_0 < T_1 < \dots < T_S = T;$$

for each segment j , number of grid points K_j , epochs E_j , and learning rates $\eta_{v,j}, \eta_{w,j}$; batch size M , nested path size N , action space A , temperature parameter $\lambda > 0$, and softmax parameter τ ; distribution of the initial state $\text{Law}(X_0)$.

2: **Output:** Trained segment networks $\{(v^{\psi_j}, w^{\phi_j})\}_{j=0}^{S-1}$ and the resulting estimated optimal value $\hat{u}$ and policy $\hat{\pi}$.

3: **Initialize:** For each $j = 0, \dots, S-1$, set

$$\Delta t_j \leftarrow \frac{T_{j+1} - T_j}{K_j - 1}, \quad t_{j,k} \leftarrow T_j + k\Delta t_j, \quad k = 0, \dots, K_j - 1,$$

and randomly initialize the segment-network weights (ψ_j, ϕ_j) .

4: **for** $j = S-1, \dots, 0$ **do**

▷ Backward induction over segments

5: Define the terminal derivative for segment j by

$$g_x^{(j)}(x) = \begin{cases} g_x(x), & j = S-1, \\ v^{\psi_{j+1}}(T_{j+1}, x), & j < S-1. \end{cases}$$

6: **for** epoch $= 1, \dots, E_j$ **do**

7: **Sample initial states for the current segment:**

8: **for** $m = 1, \dots, M$ **do**

9: **if** $j = 0$ **then**

10: Draw $\mathcal{X}_{j,0}^{(m)} \sim \text{Law}(X_0)$.

11: **else**

- 12: Draw $\mathcal{X}_{j,0}^{(m)} \sim \text{Law}(\mathcal{X}_{T_j}^{t_0, X_0})$. ▷ Empirical law obtained from forward simulation
- 13: Execute the primary-path simulation, nested-path simulation, and residual computation as in Algorithm 1, restricted to the interval $[T_j, T_{j+1}]$ with time step Δt_j . Replace the terminal derivative target in Algorithm 1 by $g_x^{(j)}(\mathcal{X}_{j, K_j-1}^{(m)})$.
- 14: Aggregate the batch residuals to obtain the segment loss J_j .
- 15: Update
- $$\psi_j \leftarrow \psi_j - \eta_{v,j} \nabla_{\psi_j} J_j, \quad \phi_j \leftarrow \phi_j - \eta_{w,j} \nabla_{\phi_j} J_j.$$
- 16: Store the trained segment networks (v^{ψ_j}, w^{ϕ_j}) .
- 17: Construct $\hat{\pi}(t, x, a)$ by stitching $\{w^{\phi_j}\}_{j=0}^{S-1}$ through (2.7).
- 18: Recover $\hat{u}$ using independent paths and $\{w^{\phi_j}\}_{j=0}^{S-1}$ through (2.13).
- 19: **Return:** $(\{v^{\psi_j}, w^{\phi_j}\}_{j=0}^{S-1}, \hat{u}, \hat{\pi})$.
-

There is a key design choice that distinguishes the multi-period scheme from the single-period one, Algorithm 1, where the initial state distribution, $\text{Law}(X_0)$, is typically *fixed* as, e.g., a Dirac or uniform distribution. In the multi-period setting, however, one needs to utilize the learned value function of period j as the terminal condition for period $j - 1$. To incorporate this backward coupling, at the starting point of period j , we draw initial states from the empirical law of the forward reference paths $\mathcal{X}_{t_j}^{t_0, x_0}$ (see line 12 of Algorithm 2). This mechanism is both natural and canonical, so as to match the distributional weighting inherent to the Feynman–Kac representation (2.13), thereby strictly aligning the training objective of the algorithm with the underlying theory.

Example 5.5. We now compare Algorithm 1 with Algorithm 2 using Examples 5.1 and 5.2 with different T ’s while keeping all other parameters (including Δt and the neural network hyperparameters) unchanged. Algorithm 1 is trained on a single period, while Algorithm 2 is trained on multiple periods dividing the given single period. The following two tables show that Algorithm 2 clearly outperforms Algorithm 1 in terms of learning accuracy. Also, the former reduces runtime substantially for larger T .

Case I	Case II
Example 5.1	Example 5.1
$T = 0.6$	$T = 0.8$
Single time segment: $[0, 0.6]$	Single time segment: $[0, 0.8]$
Multi time segments: $[0, 0.2], [0.2, 0.4], [0.4, 0.6]$	Multi time segments: $[0, 0.2], \dots, [0.6, 0.8]$

Time segments	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
Case I Single	0.3642	0.3170	12.9657%	0.8409%	72.35
Case I Multi	0.3642	0.3542	2.7890%	1.1365%	121.98
Case II Single	0.3642	0.2717	25.3914%	0.7311%	609.40
Case II Multi	0.3642	0.3676	1.3194%	2.4641%	147.29

Table 7: Comparison of two algorithms on Example 5.1.

Case III Example 5.2 $T = 0.2$ Single time segment: $[0, 0.2]$ Multiple time segments: $[0, 0.1], [0.1, 0.2]$	Case IV Example 5.2 $T = 0.3$ Single time segment: $[0, 0.3]$ Multiple time segments: $[0, 0.1], [0.1, 0.2], [0.2, 0.3]$
--	---

Time segment	$u^*(t_0, x_0)$	Mean $\hat{u}(t_0, x_0)$	Mean RE	STD RE	Mean runtime (s)
Case III Single	1.0000	0.8771	12.2906%	0.0642%	54.60
Case III Multi	1.0000	0.9673	3.2677%	0.9841%	87.19
Case IV Single	1.0000	0.7755	22.4452%	1.161%	456.89
Case IV Multi	1.0000	1.0267	2.8463%	1.2735%	135.78

Table 8: Comparison of two algorithms on Example 5.2.

6 A Special Case with Controlled Diffusion Coefficients

The general problem considered so far excludes the case when control enters into the diffusion coefficients of the dynamics, for which the approach we develop fails. In this section, we discuss a special case with controllable diffusion coefficients and one-dimensional state space in infinite time horizon and offer a numerical example. The treatment is based on the main idea of [26].

Recall the infinite horizon problem formulated at the end of §3 and consider only the one-dimensional case, i.e., $d = 1$. The entropy-regularized control problem with control-

dependent diffusion is

$$\begin{aligned}
dX_t^\pi &= \tilde{b}(X_t^\pi, \pi(t, X_t^\pi))dt + \sqrt{\widetilde{\sigma}^2(X_t^\pi, \pi(t, X_t^\pi))}dW_t; \\
J(x, \pi) &:= \mathbb{E} \left[\int_0^\infty e^{-\rho t} [\tilde{r}(X_t^\pi, \pi(t, X_t^\pi)) + \lambda \mathcal{H}(\pi(t, X_t^\pi))] dt \mid X_0^\pi = x \right]; \\
u^*(x) &:= \sup_{\pi \in \mathcal{A}} J(x, \pi).
\end{aligned} \tag{6.1}$$

The corresponding exploratory HJB is

$$\begin{aligned}
\rho u^* &= H(x, u_x^*, u_{xx}^*), \quad x \in \mathbb{R}, \quad \text{where} \\
H(x, z, q) &:= \sup_{\pi \in \mathcal{P}_0(A)} \left[\frac{1}{2} \widetilde{\sigma}^2(x, \pi) q + \tilde{b}(x, \pi) z + \tilde{r}(x, \pi) + \lambda \mathcal{H}(\pi) \right].
\end{aligned} \tag{6.2}$$

The optimal feedback policy has the Gibbs form:

$$\begin{aligned}
\pi^*(x, a) &= \Gamma(x, u_x^*, u_{xx}^*, a), \quad \text{where } \Gamma(x, z, q, a) := \frac{\gamma(x, z, q, a)}{\int_A \gamma(x, z, q, a') da'}, \\
\gamma(x, z, q, a) &:= \exp \left(\frac{1}{\lambda} \left[\frac{1}{2} \sigma^2(x, a) q + b(x, a) z + r(x, a) \right] \right).
\end{aligned} \tag{6.3}$$

One can consequently rewrite (6.2) as

$$\rho u^*(x) = \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma^2(x, a) \partial_{xx} u^*(x) + b(x, a) \partial_x u^*(x) + r(x, a) \right] \right\} da.$$

We make the following strengthened standing assumptions, including in particular the so-called *smallness* assumption on the dependence of the diffusion coefficient σ on control which is used also in e.g. [33].

Assumption 6.1. *There exist constants $L_*, \varepsilon > 0$ such that the followings hold true:*

- (i) $d = 1; b, r$ satisfy Assumption 3.1;
- (ii) $\sigma^2(x, a) = \sigma_0^2(x) + \sigma_1^2(x, a)$, where σ_0 satisfies Assumption 3.1 and $|\sigma_1^2| \leq \varepsilon$;
- (iii) The true solution u^* of the HJB equation (6.2) satisfies

$$\|u^*\|_0 + \|u_x^*\|_0 + \|u_{xx}^*\|_0 \leq L_*.$$

Throughout this section, we omit the time discretization analysis as it is completely parallel to the one in previous sections, focusing on the key ideas for designing the algorithm. As a result, in contrast to Assumption 3.1 we do not require the boundedness of $\|u_{xxx}^*\|_0$ here, because the convergence analysis presented here does not involve time discretization errors. Moreover, the boundedness of $\|u^*\|_0$ is not an additional assumption, as it follows directly from the boundedness of the reward function r .

Under Assumption 6.1, the corresponding HJB equation becomes

$$\rho u^*(x) = \frac{1}{2}\sigma_0^2(x)u_{xx}^*(x) + \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2}\sigma_1^2(x, a)u_{xx}^*(x) + b(x, a)u_x^*(x) + r(x, a) \right] \right\} da \quad (6.4)$$

Denote

$$v^*(x) := u_x^*(x), \quad \theta^*(x) := u_{xx}^*(x), \quad w^*(x, a) := b(x, a)u_x^*(x) + r(x, a).$$

Following the same idea as in §2 by applying the Feynman–Kac formula and Bismut–Elworthy–Li representation, we first have

$$u^*(x) = \mathbb{E} \left[\int_0^\infty e^{-\rho t} \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2}\sigma_1^2(\mathcal{X}_t^x, a)\theta^*(\mathcal{X}_t^x) + w^*(\mathcal{X}_t^x, a) \right] \right\} dadt \right], \quad (6.5)$$

$$v^*(x) = \mathbb{E} \left[\int_0^\infty e^{-\rho t} N_t^x \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2}\sigma_1^2(\mathcal{X}_t^x, a)\theta^*(\mathcal{X}_t^x) + w^*(\mathcal{X}_t^x, a) \right] \right\} dadt \right], \quad (6.6)$$

where $\mathcal{X}^x$ is the dynamics of the reference state in infinite horizon:

$$\mathcal{X}_t^x := x + \int_0^t \sigma_0(\mathcal{X}_s^x) dB_s, \quad t > 0. \quad (6.7)$$

The kernel term in the representation of v^* in (6.6) becomes

$$N_t^x = \frac{1}{t} \int_0^t \sigma_0^{-1}(\mathcal{X}_s^x) \nabla \mathcal{X}_s^x dB_s, \quad \nabla \mathcal{X}_t^x = 1 + \int_0^t \partial_x \sigma_0(\mathcal{X}_s^x) \nabla \mathcal{X}_s^x dB_s. \quad (6.8)$$

Mimicking the estimate (2.19) in §2, we have

$$w^*(x, a) \approx \frac{1}{\Delta t} \mathbb{E} \left[\left(\int_x^{X_{\Delta t}^{x,a}} - \int_x^{\mathcal{X}_{\Delta t}^{x,a}} \right) v^*(x') dx' + r(x, a) \Delta t \right], \quad (6.9)$$

where $\mathcal{X}^{x,a}$ is a new, control-dependent reference process defined as

$$\mathcal{X}_t^{x,a} := x + \int_0^t \sigma(\mathcal{X}_s^{x,a}, a) dB_s, \quad t > 0, \quad (6.10)$$

while the controlled state dynamics is

$$X_t^{x,a} := x + \int_0^t b(X_s^{x,a}, a) ds + \int_0^t \sigma(X_s^{x,a}, a) dB_s, \quad t > 0. \quad (6.11)$$

Finally, thanks to the special structure of σ^2 and the simplicity in the one-dimensional case, we have the following representation of the second-order derivative term,

$$\theta^*(x) = \frac{2\rho}{\sigma_0^2(x)} u(x) - \frac{2}{\sigma_0^2(x)} \left[\lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2}\sigma_1^2(x, a)\theta^*(x) + w^*(x, a) \right] \right\} da \right]. \quad (6.12)$$

Denote

$$\mathcal{U} := C_b^0(\mathbb{R}; \mathbb{R}), \quad \mathcal{V} := C_b^0(\mathbb{R}; \mathbb{R}), \quad \mathcal{W} := C_b^0(\mathbb{R} \times A; \mathbb{R}), \quad \Theta := C_b^0(\mathbb{R}; \mathbb{R}).$$

In light of the relations given by (6.5), (6.6), (6.9), and (6.12), we introduce four mappings

$$\Lambda : \mathcal{W} \times \Theta \mapsto \mathcal{U}, \quad \Phi : \mathcal{W} \times \Theta \mapsto \mathcal{V}, \quad \Psi : \mathcal{V} \mapsto \mathcal{W}, \quad \Pi : \mathcal{U} \times \mathcal{W} \times \Theta \mapsto \Theta, \quad (6.13)$$

as follows: for any $(u, v, w, \theta) \in \mathcal{U} \times \mathcal{V} \times \mathcal{W} \times \Theta$ and $(x, a) \in [0, \infty) \times A$,

$$\begin{aligned} \Lambda(w, \theta)(x) &:= \mathbb{E} \left[\int_0^\infty e^{-\rho t} \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta(\mathcal{X}_t^x) + w(\mathcal{X}_t^x, a) \right] \right\} da dt \right], \\ \Phi(w, \theta)(x) &:= \mathbb{E} \left[\int_0^\infty e^{-\rho t} N_t^x \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta(\mathcal{X}_t^x) + w(\mathcal{X}_t^x, a) \right] \right\} da dt \right], \\ \Psi(v)(x, a) &:= \frac{1}{\Delta t} \mathbb{E} \left[\left(\int_x^{X_{\Delta t}^{x,a}} - \int_x^{\mathcal{X}_{\Delta t}^{x,a}} \right) v(x') dx' + r(x, a) \Delta t \right], \\ \Pi(u, w, \theta)(x) &:= \frac{2\rho}{\sigma_0^2(x)} u(x) - \frac{2}{\sigma_0^2(x)} \left[\lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(x, a) \theta(x) + w(x, a) \right] \right\} da \right]. \end{aligned} \quad (6.14)$$

Note the above mappings only use the coefficient σ which is assumed to be known (including its components σ_0 and σ_1), as well as $\mathcal{X}^x, \mathcal{X}^{x,a}$ which can be simulated and $X^{x,a}, r$ which are observed data processes upon queries.

Our algorithm is now built on the iteration of (u, v, w, θ) . Similar to (3.11), introduce the new loss function

$$J(u, v, w, \theta) := \|\Lambda(w, \theta) - u\|_0 + \|\Phi(w, \theta) - v\|_0 + \|\Psi(v) - w\|_0 + \|\Pi(u, w, \theta) - \theta\|_0. \quad (6.15)$$

Proposition 6.2. *There exists a constant $C_{L_*} > 0$, depending on L_0, L_1, L_* , such that*

$$J(u^*, v^*, w^*, \theta^*) \leq C_{L_*} \Delta t. \quad (6.16)$$

All the proofs in this section become technically easy without considering the time discretization error and are postponed to Appendix. The following estimates are needed to show the convergence.

Recall the number ε in Assumption 6.1-(ii).

Lemma 6.3. *There exists a constant $C > 0$, depending only on m, λ, L_0, L_1 such that, for any $(u, v, w, \theta), (u', v', w', \theta') \in \mathcal{U} \times \mathcal{V} \times \mathcal{W} \times \Theta$ and $(x, a) \in [0, \infty) \times A$, we have*

$$\begin{aligned} \|\Delta \Phi\|_0 + \|\Delta \Psi\|_0 &\leq \frac{C}{\sqrt{\rho}} \left\{ \varepsilon \|\Delta \theta\|_0 + \|\Delta w\|_0 \right\}, \\ \|\Delta \Lambda\|_0 &\leq \frac{C}{\rho} \left\{ \varepsilon \|\Delta \theta\|_0 + \|\Delta w\|_0 \right\}, \\ \|\Delta \Pi\|_0 &\leq C \left\{ \varepsilon \|\Delta \theta\|_0 + \frac{1}{\sqrt{\rho}} \|\Delta w\|_0 \right\}, \end{aligned} \quad (6.17)$$

where $\Delta \Phi := \Phi(w, \theta) - \Phi(w', \theta')$, $\Delta \theta := \theta - \theta'$, and similarly for $\Delta \Psi, \Delta \Lambda, \Delta \Pi$ and Δw .

We finally arrive at the main result of the section.

Theorem 6.4. *There exist constants C, ρ_0, ε_0 depending on $m, \lambda, |A|$ and L_0, L_1 but not on L_* , such that for all $\rho > \rho_0$ and $\varepsilon < \varepsilon_0$,*

$$\begin{aligned} & \|\Lambda(w, \theta) - u^*\|_0 + \|\Phi(w, \theta) - v^*\|_0 + \|\Psi(v) - w^*\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \\ \leq & \quad C J(u, v, w, \theta) + C_{L_*} \Delta t. \end{aligned} \quad (6.18)$$

Thus again, the problem boils down to solving the minimization problem

$$\inf_{u, v, w, \theta} J(u, v, w, \theta),$$

leading to the following algorithm.

Algorithm 3 (Algorithm for problems with controlled diffusion coefficients)

1: **Input:** Fixed initial time $t_0 = 0$, truncation terminal time T , number of grid points K , batch size M , nested path size N , and epochs E ; discount $\rho > 0$, action space A , temperature parameter $\lambda > 0$, softmax parameter τ , and learning rates $\eta_u, \eta_v, \eta_w, \eta_\theta$; distribution of the initial state $\text{Law}(X_0)$.

2: **Output:** Trained networks $(u^\psi, v^\gamma, w^\phi, \theta^\xi)$ and the resulting estimated optimal value $\hat{u}$ and policy $\hat{\pi}$.

3: **Initialize:** Set

$$\Delta t \leftarrow \frac{T - t_0}{K - 1}, \quad t_k \leftarrow t_0 + k \Delta t, \quad k = 0, \dots, K - 1,$$

and randomly initialize the weights $(\psi, \gamma, \phi, \xi)$ for $u^\psi(x)$, $v^\gamma(x)$, $w^\phi(x, a)$, and $\theta^\xi(x)$.

4: **for** epoch = 1, ..., E **do**

5: **for** $m = 1, \dots, M$ **do** ▷ Batch generation

6: Draw $\mathcal{X}_0^{(m)} \sim \text{Law}(X_0)$.

7: Draw primary Brownian increments $\{\Delta B_k^{(m)}\}_{k=0}^{K-2} \sim \mathcal{N}(0, \Delta t)$.

8: Set

$$\nabla \mathcal{X}_0^{(m)} \leftarrow 1, \quad N_0^{(m)} \leftarrow 0.$$

9: **for** $k = 0$ **to** $K - 2$ **do** ▷ Forward simulation of the primary reference path

10:

$$\mathcal{X}_{k+1}^{(m)} \leftarrow \mathcal{X}_k^{(m)} + \sigma_0(\mathcal{X}_k^{(m)}) \Delta B_k^{(m)}.$$

11:

$$\nabla \mathcal{X}_{k+1}^{(m)} \leftarrow \nabla \mathcal{X}_k^{(m)} \exp \left(\sigma'_0(\mathcal{X}_k^{(m)}) \Delta B_k^{(m)} - \frac{1}{2} \left| \sigma'_0(\mathcal{X}_k^{(m)}) \right|^2 \Delta t \right).$$

12:

$$N_{k+1}^{(m)} \leftarrow \frac{t_k - t_0}{t_{k+1} - t_0} N_k^{(m)} + \frac{1}{t_{k+1} - t_0} \nabla \mathcal{X}_k^{(m)} \sigma_0^{-1}(\mathcal{X}_k^{(m)}) \Delta B_k^{(m)}.$$

13: For $k = 0, \dots, K-1$, compute

$$Q_k^{(m)} \leftarrow \lambda \log \int_A \exp \left\{ \frac{1}{\lambda} \left(\frac{1}{2} \sigma_1^2(\mathcal{X}_k^{(m)}, a) \theta^\xi(\mathcal{X}_k^{(m)}) + w^\phi(\mathcal{X}_k^{(m)}, a) \right) \right\} da.$$

14: Compute the path-level targets at $\mathcal{X}_0^{(m)}$:

$$\begin{aligned} \tilde{u}^{(m)} &\leftarrow \sum_{k=1}^{K-1} e^{-\rho(t_k - t_0)} Q_k^{(m)} \Delta t, \\ \tilde{v}^{(m)} &\leftarrow \sum_{k=1}^{K-1} e^{-\rho(t_k - t_0)} N_k^{(m)} Q_k^{(m)} \Delta t, \end{aligned}$$

and

$$\tilde{\theta}^{(m)} \leftarrow \frac{2\rho}{\sigma_0^2(\mathcal{X}_0^{(m)})} u^\psi(\mathcal{X}_0^{(m)}) - \frac{2}{\sigma_0^2(\mathcal{X}_0^{(m)})} Q_0^{(m)}.$$

15: **for** $k = 0$ **to** $K-2$ **do**

▷ Compute w -targets along the path

16: **for** $n = 1, \dots, N$ **do**

▷ Nested transitions conditional on $a^{(n)}$

17: Draw $\Delta \hat{B}_k^{(m,n)} \sim \mathcal{N}(0, \Delta t)$.

18: Sample $a^{(n)} \sim \text{Unif}(A)$

19: Apply the fixed action $a^{(n)}$ at $\mathcal{X}_k^{(m)}$ and observe $X_{k+1}^{a^{(n)}, (m,n)}$.

20: Simulate the nested reference transition

$$\mathcal{X}_{k+1}^{(m,n)} \leftarrow \mathcal{X}_k^{(m)} + \sigma(\mathcal{X}_k^{(m)}, a^{(n)}) \Delta \hat{B}_k^{(m,n)}.$$

21: Compute

$$W_k^{(m,n)} \leftarrow \int_{\mathcal{X}_k^{(m)}}^{X_{k+1}^{a^{(n)}, (m,n)}} v^\gamma(x) dx - \int_{\mathcal{X}_k^{(m)}}^{\mathcal{X}_{k+1}^{(m,n)}} v^\gamma(x) dx + r(\mathcal{X}_k^{(m)}, a^{(n)}) \Delta t.$$

22:

$$\tilde{w}_k^{(m)} \leftarrow \frac{1}{N \Delta t} \sum_{n=1}^N W_k^{(m,n)}.$$

23:

$$\ell_{w,k}^{(m)} \leftarrow \left| w^\phi(\mathcal{X}_k^{(m)}, a^{(n)}) - \tilde{w}_k^{(m)} \right|.$$

24: Compute

$$\begin{aligned} \ell_u^{(m)} &\leftarrow \left| u^\psi(\mathcal{X}_0^{(m)}) - \tilde{u}^{(m)} \right|, & \ell_v^{(m)} &\leftarrow \left| v^\gamma(\mathcal{X}_0^{(m)}) - \tilde{v}^{(m)} \right|, \\ \ell_\theta^{(m)} &\leftarrow \left| \theta^\xi(\mathcal{X}_0^{(m)}) - \tilde{\theta}^{(m)} \right|, & L_w^{(m)} &\leftarrow \text{softmax}_\tau \left(\{ \ell_{w,k}^{(m)} \}_{k=0}^{K-2} \right). \end{aligned}$$

25: Aggregate the batch losses:

$$\begin{aligned} L_u &\leftarrow \frac{1}{M} \sum_{m=1}^M \ell_u^{(m)}, & L_v &\leftarrow \frac{1}{M} \sum_{m=1}^M \ell_v^{(m)}, \\ L_\theta &\leftarrow \frac{1}{M} \sum_{m=1}^M \ell_\theta^{(m)}, & L_w &\leftarrow \frac{1}{M} \sum_{m=1}^M L_w^{(m)}. \end{aligned}$$

26: Set

$$J \leftarrow L_u + L_v + L_w + L_\theta.$$

27: Update

$$\begin{aligned}\psi &\leftarrow \psi - \eta_u \nabla_\psi J, & \gamma &\leftarrow \gamma - \eta_v \nabla_\gamma J, \\ \phi &\leftarrow \phi - \eta_w \nabla_\phi J, & \xi &\leftarrow \xi - \eta_\theta \nabla_\xi J.\end{aligned}$$

28: Set $\hat{u} \leftarrow u^\psi$ and obtain $\hat{\pi}$ from (w^ϕ, θ^ξ) through the entropy-regularized optimizer.

29: **Return:** $(u^\psi, v^\gamma, w^\phi, \theta^\xi, \hat{u}, \hat{\pi})$.

Example 6.5. Consider an example where the dynamics coefficients are

$$b(x, a) = xe^{-2(x^2+a)}, \quad \sigma_0^2(x) = 1, \quad \sigma_1^2(x, a) = e^{-(x^2+a)},$$

and the running reward is

$$r(x, a) = (e^{-2(x^2+a)} - 2x^2 + \rho + 1)e^{-x^2}.$$

Setting $\lambda = 1$, the HJB (6.4) has the oracle solution along with its derivatives

$$u^*(x) = e^{-x^2}, \quad v^*(x) = -2xu^*(x), \quad \theta^*(x) = (4x^2 - 2)u^*(x).$$

Although the problem is in infinite horizon, for numerical implementation we terminate the time at $T = 0.2$. We also set $x_0 = 0$, $\rho = 50$, $\Delta t = 0.02$ in implementation. The result is given in the following table, which shows a somewhat similar, if not slightly less, accuracy compared to the average accuracy in the previously reported examples for drift control problems.

$u^*(x_0)$	Mean $\hat{u}(x_0)$	Mean RE	STD RE	Mean runtime (s)
1.0000	0.9330	6.700%	4.827%	58.06

Table 9: Numerical results for Example 6.5.

7 Conclusions

This paper studies potentially high-dimensional stochastic control problems with missing key primitives, and develops data-driven algorithms that learn the optimal value functions and optimal randomized policies with theoretical guarantees on convergence and convergence rates. The method also provides a scalable way to numerically solve certain black-box PDEs. It is distinctive from and complementary to the other approaches developed

to solve similar model-free/black-box problems, such as stochastic approximation [32] and zeroth-order derivative [22].

There are clearly outstanding open questions going forward. One is to allow diffusion terms to depend on control in the general setting. Control entering into diffusion is important not only because this feature is inherent in many applications (e.g. portfolio selection), but also because this is the very feature that fundamentally distinguishes stochastic controls from determinist ones (see the many discussions on this point in Yong and Zhou [37]). Technically, this feature demands an additional (and most likely delicate) analysis on Hessian, which we already had a glimpse in §6 for a very special case.

The other open question is when diffusion coefficients and/or terminal rewards are also unknown, which may arise in certain applications. While such a case would invalidate our current approach right away, it is interesting to investigate a way around, namely to keep the same big idea of probabilistic representations of the derivatives but find a different mapping for the fix-point argument to work.

Finally, as discussed in Remark 3.4, the assumption on the (unknown) environment in terms of how data are observed, Assumption 3.3-(i), can be strong for many applications. It is important to verify and make sense of this assumption in a given application. What is more interesting, however, is to investigate whether and how generative AI might help when the assumption fails.

8 Appendix

8.1 Proof of Proposition 3.7

Proof We first analyze Ψ_K . For any (t_k, x, a) , by (2.11) and (3.7) we have

$$\begin{aligned}
& (\Psi_K(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*)(t_k, x, a) \\
&= \frac{1}{\Delta t} \mathbb{E}_{t_k, x} \left[\int_x^{X_{t_k+1}^a} v^*(t_k, x') dx' - \int_x^{\mathcal{X}_{t_k+1}} v^*(t_k, x') dx' \right] - v^*(t_k, x) \cdot b(t_k, x, a) \\
&= \frac{1}{\Delta t} \mathbb{E}_{t_k, x} \left[[u^*(t_k, X_{t_k+1}^a) - u^*(t_k, x)] - [u^*(t_k, \mathcal{X}_{t_k+1}) - u^*(t_k, x)] \right] - v^*(t_k, x) \cdot b(t_k, x, a) \\
&= \frac{1}{\Delta t} \mathbb{E}_{t_k, x} \left[\int_{t_k}^{t_{k+1}} \left[\frac{1}{2} \sigma \sigma^\top : u_{xx}^*(t_k, X_s^a) + b(t_k, X_s^a, a) \cdot v^*(t_k, X_s^a) \right. \right. \\
&\quad \left. \left. - \frac{1}{2} \sigma \sigma^\top : u_{xx}^*(t_k, \mathcal{X}_s) \right] ds \right] - v^*(t_k, x) \cdot b(t_k, x, a) \\
&= \frac{1}{2\Delta t} \mathbb{E}_{t_k, x} \left[\int_{t_k}^{t_{k+1}} [\sigma \sigma^\top : u_{xx}^*(t_k, X_s^a) - \sigma \sigma^\top : u_{xx}^*(t_k, x)] ds \right] \\
&\quad - \frac{1}{2\Delta t} \mathbb{E}_{t_k, x} \left[\int_{t_k}^{t_{k+1}} [\sigma \sigma^\top : u_{xx}^*(t_k, \mathcal{X}_s) - \sigma \sigma^\top : u_{xx}^*(t_k, x)] ds \right] \\
&\quad + \frac{1}{\Delta t} \mathbb{E}_{t_k, x} \left[\int_{t_k}^{t_{k+1}} [v^*(t_k, X_s^a) \cdot b(t_k, X_s^a, a) - v^*(t_k, x) \cdot b(t_k, x, a)] ds \right].
\end{aligned}$$

Then, by Assumption 3.1 and Lemma 3.5,

$$\left| (\Psi_K(v_{\mathbb{T}_K}^*) - w_{\mathbb{T}_K}^*) \right| \leq C_{L^*} \sup_{t_k \leq s \leq t_{k+1}} \mathbb{E} \left[|X_s^a - x| + |\mathcal{X}_s - x| \right] ds \leq C_{L^*, T} \sqrt{\Delta t}. \quad (8.19)$$

We next analyze Φ_K . Recall (2.8) and denote

$$H^*(t, x) := H(t, x, u_x^*(t, x)). \quad (8.20)$$

For any (t_k, x) , and assuming without loss of generality that $k = 0$, by (2.14) and (3.7) we have

$$\begin{aligned}
v^*(0, x) - \Phi_K(w_{\mathbb{T}_K}^*)(0, x) &= \varepsilon_K^{\Delta t} + \varepsilon_0^{\Delta t} + \sum_{j=1}^{K-1} [\varepsilon_j^{1, \Delta t} + \varepsilon_j^{2, \Delta t}], \quad \text{where} \quad (8.21) \\
\varepsilon_K^{\Delta t} &:= \mathbb{E}_{0, x} \left[(\nabla \mathcal{X}_T)^\top g_x(\mathcal{X}_T) - (\nabla \mathcal{X}_T^{\Delta t})^\top g_x(\mathcal{X}_T^{\Delta t}) \right], \\
\varepsilon_0^{\Delta t} &:= \mathbb{E}_{0, x} \left[\int_{t_k}^{t_{k+1}} H^*(s, \mathcal{X}_s) N_s ds \right], \\
\varepsilon_j^{1, \Delta t} &:= \mathbb{E}_{0, x} \left[\int_{t_j}^{t_{j+1}} [H^*(s, \mathcal{X}_s) - H^*(t_j, \mathcal{X}_{t_j}^{\Delta t})] N_s ds \right], \\
\varepsilon_j^{2, \Delta t} &:= \mathbb{E}_{0, x} \left[\int_{t_j}^{t_{j+1}} H^*(t_j, \mathcal{X}_{t_j}^{\Delta t}) [N_s - N_{t_j}^{\Delta t}] ds \right],
\end{aligned}$$

for $j = 1, \dots, K-1$. By (2.8) one can easily verify that

$$\|H^*\|_0 \leq C_{L*}, \quad |H^*(t, x) - H^*(t', x')| \leq C_{L*} [\sqrt{|t - t'|} + |x - x'|]. \quad (8.22)$$

Then, applying Lemma 3.5, we have

$$\begin{aligned} |\varepsilon_K^{\Delta t}| &\leq C_{L*} \mathbb{E}_{0,x} \left[|\nabla \mathcal{X}_T - \nabla \mathcal{X}_T^{\Delta t}| + |\nabla \mathcal{X}_T| |\mathcal{X}_T - \mathcal{X}_T^{\Delta t}| \right] \leq C_{L*,T} \sqrt{\Delta t}, \\ |\varepsilon_0^{\Delta t}| &\leq C_{L*,T} \mathbb{E} \left[\int_{t_0}^{t_1} |N_s| ds \right] \leq C_{L*,T} \int_{t_0}^{t_1} \frac{1}{\sqrt{s}} ds = C_{L*,T} \sqrt{\Delta t}. \end{aligned} \quad (8.23)$$

Moreover, note that

$$\begin{aligned} |\varepsilon_j^{1,\Delta t}| &\leq C_{L*} \mathbb{E}_{0,x} \left[\int_{t_j}^{t_{j+1}} [\sqrt{s - t_j} + |\mathcal{X}_s - \mathcal{X}_{t_j}| + |\mathcal{X}_{t_j} - \mathcal{X}_{t_j}^{\Delta t}|] |N_s| ds \right] \\ &\leq C_{L*} \int_{t_j}^{t_{j+1}} \left(\mathbb{E}_{0,x} [s - t_j + |\mathcal{X}_s - \mathcal{X}_{t_j}|^2 + |\mathcal{X}_{t_j} - \mathcal{X}_{t_j}^{\Delta t}|^2] \right)^{\frac{1}{2}} \left(\mathbb{E}_{0,x} [|N_s|^2] \right)^{\frac{1}{2}} ds \\ &\leq C_{L*,T} \sqrt{\Delta t} \int_{t_j}^{t_{j+1}} \frac{1}{\sqrt{s}} ds. \end{aligned}$$

Then

$$\sum_{j=1}^{K-1} |\varepsilon_j^{1,\Delta t}| \leq \sum_{j=1}^{K-1} C_{L*,T} \sqrt{\Delta t} \int_{t_j}^{t_{j+1}} \frac{1}{\sqrt{s}} ds = C_{L*,T} \sqrt{\Delta t} \int_{t_1}^T \frac{1}{\sqrt{s}} ds \leq C_{L*,T} \sqrt{\Delta t}. \quad (8.24)$$

It remains to estimate $\varepsilon_j^{2,\Delta t}$. Denote

$$I_l := (\sigma^{-1}(l, \mathcal{X}_l) \nabla \mathcal{X}_l)^\top, \quad I_l^{\Delta t} := \sum_{i=0}^{K-1} \mathbf{1}_{[t_i, t_{i+1})}(l) (\sigma^{-1}(t_i, \mathcal{X}_{t_i}^{\Delta t}) \nabla \mathcal{X}_{t_i}^{\Delta t})^\top.$$

Then, noting that $\frac{\Delta t}{t_j} = \frac{1}{j}$, we have

$$\begin{aligned} \varepsilon_j^{2,\Delta t} &= \mathbb{E}_{0,x} \left[\int_{t_j}^{t_{j+1}} H^*(t_j, \mathcal{X}_{t_j}^{\Delta t}) [\mathbb{E}_{0,x} [N_s | \mathcal{F}_{t_j}] - N_{t_j}^{\Delta t}] ds \right] \\ &= \mathbb{E}_{0,x} \left[\int_{t_j}^{t_{j+1}} H^*(t_j, \mathcal{X}_{t_j}^{\Delta t}) \left[\frac{1}{s} \int_0^{t_j} I_l dB_l - \frac{\Delta t}{t_j} \int_0^{t_j} I_l^{\Delta t} dB_l \right] ds \right] \\ &= \mathbb{E}_{0,x} \left[H^*(t_j, \mathcal{X}_{t_j}^{\Delta t}) \left[\ln \frac{t_{j+1}}{t_j} \int_0^{t_j} I_l dB_l - \frac{\Delta t}{t_j} \int_0^{t_j} I_l^{\Delta t} dB_l \right] \right] \\ &= \mathbb{E}_{0,x} \left[H^*(t_j, \mathcal{X}_{t_j}^{\Delta t}) \left[\frac{1}{j} \int_0^{t_j} (I_l - I_l^{\Delta t}) dB_l + c_j \int_0^{t_j} I_l dB_l \right] \right], \end{aligned}$$

where $c_j := \ln \frac{t_{j+1}}{t_j} - \frac{1}{j} = \ln(1 + \frac{1}{j}) - \frac{1}{j}$ satisfies $|c_j| \leq \frac{C}{j^2}$. Accordingly, by (8.22),

$$\begin{aligned} |\varepsilon_j^{2,\Delta t}|^2 &\leq \frac{C_{L*}}{j^2} \mathbb{E}_{0,x} \left[\int_0^{t_j} |I_l - I_l^{\Delta t}|^2 dl \right] + \frac{C}{j^4} \mathbb{E}_{0,x} \left[\int_0^{t_j} |I_l|^2 dl \right] \\ &\leq \frac{C_{L*}}{j^2} \sum_{i=0}^{j-1} \int_{t_i}^{t_{i+1}} \mathbb{E}_{0,x} [|I_l - I_l^{\Delta t}|^2] dl + \frac{C t_j}{j^4}. \end{aligned}$$

For $l \in [t_i, t_{i+1}]$, it follows easily from Assumption 3.1 and Lemma 3.5 that

$$\mathbb{E}_{0,x} [|I_l - I_l^{\Delta t}|^2] \leq C \mathbb{E}_{0,x} [|I_l - I_{t_i}|^2 + |I_{t_i} - I_{t_i}^{\Delta t}|^2] \leq C_{L^*,T} \Delta t.$$

Then

$$|\varepsilon_j^{2,\Delta t}|^2 \leq \frac{C_{L^*,T}}{j^2} \sum_{i=0}^{j-1} |\Delta t|^2 + \frac{C \Delta t}{j^3} \leq \frac{C_{L^*,T} |\Delta t|^2}{j}.$$

Thus

$$\sum_{j=1}^{K-1} |\varepsilon_j^{2,\Delta t}| \leq C_{L^*,T} \Delta t \sum_{j=1}^{K-1} \frac{1}{\sqrt{j}} \leq C_{L^*,T} \Delta t \sqrt{K} = C_{L^*,T} \sqrt{T \Delta t}. \quad (8.25)$$

Plugging (8.23), (8.24) and (8.25) into (8.21), we see that

$$|v^*(0, x) - \Phi_K(w_{\mathbb{T}_K}^*)(0, x)| \leq C_{L^*,T} \sqrt{\Delta t}.$$

This, together with (8.19) and the arbitrariness of (t_k, x, a) , proves (3.8). $\blacksquare$

8.2 Proof of Proposition 6.2

Proof The proof is much simplified once we omit the time discretization. We can directly read from the definition (6.14) that for any x ,

$$\Phi(w^*, \theta^*)(x) = v^*(x), \quad \Lambda(w^*, \theta^*)(x) = u^*(x), \quad \Pi(u^*, w^*, \theta^*) = \theta^*(x).$$

For arbitrary (x, a) ,

$$\left| (\Psi(v^*) - w^*)(x, a) \right| = \left| \frac{1}{\Delta t} \mathbb{E} \left[\left(\int_0^{\Delta t} b(X_t^{x,a}, a) v^*(X_t^{x,a}) - b(x, a) v^*(x) dt \right) \right] \right|.$$

Thus by Assumption 6.1,

$$\left| (\Psi(v^*) - w^*)(x, a) \right| \leq C_{L^*} \sup_{0 < t < \Delta t} \mathbb{E} \left[|X_t^{x,a} - x| \right] \leq C_{L^*} \Delta t.$$

The above two estimates, together with the arbitrariness of (x, a) , complete the proof. $\blacksquare$

8.3 Proof of Lemma 6.3

Proof For arbitrary (x, a) , we first have

$$\begin{aligned} & \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta(\mathcal{X}_t^x) + w(\mathcal{X}_t^x, a) \right] \right\} da \\ & \leq \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) (\theta'(\mathcal{X}_t^x) + \|\theta - \theta'\|_0) + w'(\mathcal{X}_t^x, a) + \|w - w'\|_0 \right] \right\} da \\ & = \lambda \ln \int_A \exp \left\{ \frac{1}{\lambda} \left[\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta'(\mathcal{X}_t^x) + w'(\mathcal{X}_t^x, a) \right] \right\} da + \frac{\varepsilon}{2} \|\theta - \theta'\| + \|w - w'\|_0. \end{aligned}$$

This leads to

$$\begin{aligned} & \left| \lambda \ln \int_A e^{\frac{1}{\lambda} [\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta(\mathcal{X}_t^x) + w(\mathcal{X}_t^x, a)]} da - \lambda \ln \int_A e^{\frac{1}{\lambda} [\frac{1}{2} \sigma_1^2(\mathcal{X}_t^x, a) \theta(\mathcal{X}_t^x) + w(\mathcal{X}_t^x, a)]} da \right| \\ & \leq \frac{\varepsilon}{2} \|\theta - \theta'\| + \|w - w'\|_0. \end{aligned} \quad (8.26)$$

It follows from (6.14) and Lemma (3.4) that

$$\begin{aligned} |\Delta \Phi(x)| & \leq C \left\{ \varepsilon \|\theta - \theta'\| + \|w - w'\|_0 \right\} \mathbb{E} \left[\int_0^\infty e^{-\rho t} |N_t^x| dt \right] \\ & \leq \frac{C}{\sqrt{\rho}} \left\{ \varepsilon \|\theta - \theta'\|_0 + \|w - w'\|_0 \right\}. \end{aligned} \quad (8.27)$$

Then, by (8.27), (6.14) and the arbitrariness of (x, a) , we have

$$\|\Delta \Psi\|_0 \leq C \|\Delta \Phi\|_0 \leq \frac{C}{\sqrt{\rho}} \left\{ \varepsilon \|\theta - \theta'\|_0 + \|w - w'\|_0 \right\}. \quad (8.28)$$

Next, by (6.14) and (8.26), we obtain similarly

$$\begin{aligned} |\Delta \Lambda(x)| & \leq C \left\{ \varepsilon \|\theta - \theta'\| + \|w - w'\|_0 \right\} \mathbb{E} \left[\int_0^\infty e^{-\rho t} dt \right] \\ & \leq \frac{C}{\rho} \left\{ \varepsilon \|\theta - \theta'\|_0 + \|w - w'\|_0 \right\}. \end{aligned} \quad (8.29)$$

Combining (6.14), (8.28) and (8.29), together with the arbitrariness of (x, a) , we obtain

$$\begin{aligned} \|\Delta \Pi\|_0 & \leq C \left\{ \rho \|\Delta \Lambda\|_0 + \varepsilon \|\Delta \Pi\|_0 + \|\Delta \Psi\|_0 \right\} \\ & \leq C \left\{ \varepsilon \|\theta - \theta'\|_0 + \|\Delta \Psi\|_0 \right\} \\ & \leq C \left\{ \varepsilon \|\theta - \theta'\|_0 + \frac{1}{\sqrt{\rho}} \|w - w'\|_0 \right\}. \end{aligned}$$

■

8.4 Proof of Theorem 6.4

Proof Fix ρ_0, ε_0 which will be specified later, and assume without loss of generality that $\rho_0 > 1$ and $\varepsilon_0 < 1$. Applying Proposition 6.2 and Lemma 6.3 we have

$$\begin{aligned} \|\Lambda(w, \theta) - u^*\|_0 & \leq \|\Lambda(w, \theta) - \Lambda(w^*, \theta^*)\|_0 + \|\Lambda(w^*, \theta^*) - u^*\|_0 \\ & \leq \frac{C}{\rho_0} \left\{ \varepsilon_0 \|\theta - \theta^*\|_0 + \|w - w^*\|_0 \right\} \\ & \leq C \left\{ \varepsilon_0 \left(\|\theta - \Pi(u, w, \theta)\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\ & \quad \left. + \frac{1}{\sqrt{\rho_0}} \left(\|w - \Psi(w, \theta)\|_0 + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} \\ & \leq C \left\{ \varepsilon_0 \left(J(u, v, w, \theta) + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\ & \quad \left. + \frac{1}{\sqrt{\rho_0}} \left(J(u, v, w, \theta) + \|\Psi(w, \theta) - w^*\|_0 \right) \right\}. \end{aligned} \quad (8.30)$$

Similarly, we have the following estimates for the other three terms

$$\begin{aligned}
\|\Phi(w, \theta) - v^*\|_0 &\leq \|\Phi(w, \theta) - \Phi(w^*, \theta^*)\|_0 + \|\Phi(w^*, \theta^*) - v^*\|_0 \\
&\leq \frac{C}{\sqrt{\rho_0}} \left\{ \varepsilon_0 \|\theta - \theta^*\|_0 + \|w - w^*\|_0 \right\} \\
&\leq C \left\{ \varepsilon_0 \left(\|\theta - \Pi(u, w, \theta)\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(\|w - \Psi(w, \theta)\|_0 + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} \\
&\leq C \left\{ \varepsilon_0 \left(J(u, v, w, \theta) + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(J(u, v, w, \theta) + \|\Psi(w, \theta) - w^*\|_0 \right) \right\}, \tag{8.31}
\end{aligned}$$

$$\begin{aligned}
\|\Pi(u, w, \theta) - \theta^*\|_0 &\leq \|\Pi(u, w, \theta) - \Pi(u^*, w^*, \theta^*)\|_0 + \|\Pi(u^*, w^*, \theta^*) - \theta^*\|_0 \\
&\leq C \left\{ \varepsilon_0 \|\theta - \theta^*\|_0 + \frac{1}{\sqrt{\rho_0}} \|w - w^*\|_0 \right\} \\
&\leq C \left\{ \varepsilon_0 \left(\|\theta - \Pi(u, w, \theta)\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(\|w - \Psi(w, \theta)\|_0 + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} \\
&\leq C \left\{ \varepsilon_0 \left(J(u, v, w, \theta) + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(J(u, v, w, \theta) + \|\Psi(w, \theta) - w^*\|_0 \right) \right\}, \tag{8.32}
\end{aligned}$$

and

$$\begin{aligned}
\|\Psi(v) - w^*\|_0 &\leq \|\Psi(v) - \Psi(v^*)\|_0 + \|\Psi(v^*) - w^*\|_0 \\
&\leq \frac{C}{\sqrt{\rho_0}} \left\{ \varepsilon_0 \|\theta - \theta^*\|_0 + \|w - w^*\|_0 \right\} + C_{L_*} \Delta t \\
&\leq C \left\{ \varepsilon_0 \left(\|\theta - \Pi(u, w, \theta)\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(\|w - \Psi(w, \theta)\|_0 + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} + C_{L_*} \Delta t \\
&\leq C \left\{ \varepsilon_0 \left(J(u, v, w, \theta) + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) \right. \\
&\quad \left. + \frac{1}{\sqrt{\rho_0}} \left(J(u, v, w, \theta) + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} + C_{L_*} \Delta t. \tag{8.33}
\end{aligned}$$

Combining (8.30), (8.31), (8.32) and (8.33), similar to the derivation of (3.16), we have

$$\begin{aligned}
&\|\Lambda(w, \theta) - u^*\|_0 + \|\Phi(w, \theta) - v^*\|_0 + \|\Psi(v) - w^*\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \\
&\leq C \left\{ \varepsilon_0 \left(J(u, v, w, \theta) + \|\Pi(u, w, \theta) - \theta^*\|_0 \right) + \frac{1}{\sqrt{\rho_0}} \left(J(u, v, w, \theta) + \|\Psi(w, \theta) - w^*\|_0 \right) \right\} + C_{L_*} \Delta t.
\end{aligned}$$

Set $\varepsilon_0 := \frac{1}{4C} \wedge 1$ and $\rho_0 := \frac{1}{16C^2} \vee 1$ for the above C . Then

$$\begin{aligned} & \|\Lambda(w, \theta) - u^*\|_0 + \|\Phi(w, \theta) - v^*\|_0 + \|\Psi(w, \theta) - w^*\|_0 + \|\Pi(u, w, \theta) - \theta^*\|_0 \\ \leq & \frac{C}{2} J(u, v, w, \theta) + \frac{1}{4} \left(\|\Pi(u, w, \theta) - \theta^*\|_0 + \|\Psi(w, \theta) - w^*\|_0 \right) + C_{L_*} \Delta t. \end{aligned}$$

This implies the desired result immediately. ■

References

- [1] E. Bayraktar, M. Hernandez, Q. Yan, and Y. Zhu. Policy gradient for continuous-time mean-field control. arXiv preprint, 2026. arXiv:2605.20718.
- [2] C. J. P. Bélisle, H. E. Romeijn, and R. L. Smith. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.
- [3] J.-M. Bismut. *Large Deviations and the Malliavin Calculus*, volume 45 of *Progress in Mathematics*. Birkhäuser, Boston, MA, 1984.
- [4] H. Cao, Y. Dong, and Z. Yang. A two-fold randomization framework for impulse control problems. *SIAM Journal on Control and Optimization*, 2026. To appear; arXiv:2509.12018.
- [5] Z. Cheng, X. Guo, and Y. Zhang. Deterministic policy gradient for reinforcement learning with continuous time and state. arXiv preprint, 2025. arXiv:2509.23711.
- [6] M. Dai, Y. Dong, and L. Li. Reinforcement learning for arbitrage strategies in stock index futures. SSRN working paper, 2025. SSRN: 5403455.
- [7] J. Dianetti, G. Ferrari, and R. Xu. Exploratory optimal stopping: A singular control formulation. arXiv preprint, 2024. arXiv:2408.09335; revised 11 March 2026.
- [8] K. D. Elworthy and X.-M. Li. Formulae for the derivatives of heat semigroups. *Journal of Functional Analysis*, 125(1):252–286, 1994.
- [9] W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions*, volume 25 of *Stochastic Modelling and Applied Probability*. Springer, New York, NY, second edition, 2006.
- [10] A. Friedman. *Partial Differential Equations of Parabolic Type*. Prentice-Hall, Englewood Cliffs, NJ, 1964.

- [11] M. Giegrich, C. Reisinger, and Y. Zhang. Convergence of policy gradient methods for finite-horizon exploratory linear–quadratic control problems. *SIAM Journal on Control and Optimization*, 62(2):1060–1092, 2024.
- [12] X. Guo, Y. Huang, and X. Yu. Deterministic policy gradient for learning equilibrium in time-inconsistent control problems. arXiv preprint, 2026. arXiv:2606.11798.
- [13] Z. Guo, W. Tang, and R. Xu. Conditional diffusion guidance under hard constraint: A stochastic analysis approach. arXiv preprint, 2026. arXiv:2602.05533 [cs.AI].
- [14] B. Hambly, R. Xu, and H. Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 2023.
- [15] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, 115(34):8505–8510, 2018.
- [16] R. Hu and M. Laurière. Recent developments in machine learning methods for stochastic control and games. *Numerical Algebra, Control and Optimization*, 14(3):435–525, 2024.
- [17] Y. Huang, Y. Jia, and X. Y. Zhou. Mean–variance portfolio selection by continuous-time reinforcement learning: Algorithms, regret analysis, and empirical study. arXiv preprint, 2024. arXiv:2412.16175.
- [18] Y. Huang, Y. Jia, and X. Y. Zhou. Sublinear regret for a class of continuous-time linear–quadratic reinforcement learning problems. *SIAM Journal on Control and Optimization*, 63(5):3452–3474, 2025.
- [19] Y. Huang and X. Y. Zhou. Data-driven exploration for a class of continuous-time indefinite linear–quadratic reinforcement learning problems. *IEEE Transactions on Automatic Control*, 2025. To appear; arXiv:2507.00358.
- [20] Y.-J. Huang, Z. Wang, and Z. Zhou. Convergence of policy iteration for entropy-regularized stochastic control problems. *SIAM Journal on Control and Optimization*, 63(2):752–777, 2025.
- [21] Y.-J. Huang, X. Yu, and K. Zhang. Policy iteration achieves regularized equilibrium under time inconsistency. arXiv preprint, 2026. arXiv:2603.06145.

- [22] Y. Jia, D. Ouyang, H. Pham, and X. Y. Zhou. A zeroth-order deep learning method for fully nonlinear parabolic partial differential equations with unknown coefficients. arXiv preprint, 2026. arXiv:2606.24999.
- [23] Y. Jia and X. Y. Zhou. Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research*, 23(154):1–55, 2022.
- [24] Y. Jia and X. Y. Zhou. Policy gradient and actor–critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(275):1–50, 2022.
- [25] Y. Jia and X. Y. Zhou. q -learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.
- [26] J. Ma, G. Wang, and J. Zhang. Convergence analysis for entropy-regularized control problems: A probabilistic approach. *SIAM Journal on Control and Optimization*, 64(2):816–842, 2026.
- [27] H. Pham, Y. P. Zhang, and Y. Zhu. Discretization error from regularized reinforcement learning to continuous-time stochastic control. arXiv preprint, 2026. arXiv:2604.21179.
- [28] D. Sethi, D. Šiška, and Y. Zhang. Entropy annealing for policy mirror descent in continuous time and space. *SIAM Journal on Control and Optimization*, 63(4):3006–3041, 2025.
- [29] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.
- [30] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, 1998.
- [31] W. Tang, Y. P. Zhang, and X. Y. Zhou. Exploratory HJB equations and their convergence. *SIAM Journal on Control and Optimization*, 60(6):3191–3216, 2022.
- [32] W. Tang and X. Y. Zhou. Regret of exploratory policy improvement and q -learning. arXiv preprint, 2024. arXiv:2411.01302.
- [33] H. V. Tran, Z. Wang, and Y. P. Zhang. Policy iteration for exploratory Hamilton–Jacobi–Bellman equations. *Applied Mathematics & Optimization*, 91(2):50, 2025.

- [34] H. Wang, T. Zariphopoulou, and X. Y. Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198):1–34, 2020.
- [35] H. Wang and X. Y. Zhou. Continuous-time mean–variance portfolio selection: A reinforcement learning framework. *Mathematical Finance*, 30(4):1273–1308, 2020.
- [36] X. Wei and X. Yu. Continuous time q -learning for mean-field control problems. *Applied Mathematics & Optimization*, 91(1):10, 2025.
- [37] J. Yong and X. Y. Zhou. *Stochastic Controls: Hamiltonian Systems and HJB Equations*, volume 43 of *Stochastic Modelling and Applied Probability*. Springer, New York, NY, 1999.
- [38] J. Zhang. *Backward Stochastic Differential Equations: From Linear to Fully Nonlinear Theory*, volume 86 of *Probability Theory and Stochastic Modelling*. Springer, New York, NY, 2017.