A Stochastic Maximum Principle Approach for Reinforcement Learning with Parameterized Environment

Richard Archibald

Computer Science and Computational Science, Oak Ridge National Laboratory, Oak Ridge, TN, archibaldrk@ornl.gov

Feng Bao

Department of Mathematics, Florida State University, Tallahassee, FL, bao@math.fsu.edu

Jiongmin Yong

Department of Mathematics, University of Central Florida, Orlando, Florida, jiongmin.yong@ucf.edu

Abstract

In this work, we introduce a stochastic maximum principle (SMP) approach for solving the reinforcement learning problem with the assumption that the unknowns in the environment can be parameterized based on physics knowledge. For the development of numerical algorithms, we apply an eective online parameter estimation method as our exploration technique to estimate the environment parameter during the training procedure, and the exploitation for the optimal policy is achieved by an ecient backward action learning method for policy improvement under the SMP framework. Numerical experiments are presented to demonstrate that the SMP approach for reinforcement learning can produce reliable control policy, and the gradient descent type optimization in the SMP solver requires less training episodes compared with the standard dynamic programming principle based methods.

Keywords: Reinforcement learning, optimal control, stochastic maximum principle, parameter estimation, optimal ltering

1. Introduction

Reinforcement learning (RL) is an important research area in machine learning. Dierent from supervised learning and unsupervised learning, RL aims to nd how to map situations (state) to actions (control), and the goal of the RL problem is to let an agent learn how to take actions in an environment in order to minimize a performance cost or maximize a reward. As a major machine learning task, RL has been extensively studied, and it has application potentials to solve many real-life problems, e.g., robotic automation, natural language processing, health care, image processing, and trading in nancial market. In addition to its straightforward engineering style applications, RL has also drawn increasing attention from the science community. Some recent studies show that RL techniques can be used to solve scientic problems related to dynamic experimental design in physics and chemistry [16, 24].

The mathematical foundation of the standard approach for solving the RL problem is the dynamic programming principle (DPP) [26], which was introduced to solve the optimal control problem. The main idea of the DPP approach is to consider a family of local optimal control problems with dierent initial states and times and establish relationships among these sub-problems through the Hamilton-Jacobi-Bellman equation [28]. An important numerical method for implementing the DPP for solving the RL problem is temporal dierence (TD) learning [22], and a major breakthrough was the development of an o-policy TD control algorithm known as \Q-learning" [19, 27]. The Q-Learning method can carry out TD learning eciently, and it can learn how to take optimal actions without requiring an environment model. This makes Q-Learning applicable to solve many control problems in real-life. On the other hand, TD learning also has some limitations. For example, TD learning methods typically use gradient-free optimization to determine the optimal policy, hence even the Q-learning method suers from the eciency issue due to the low convergence rate of gradient-free optimization. Another notable disadvantage of TD learning is that it's a bootstrap procedure that analyzes how good is a guess from another guess. Hence the feedback in TD Learning is delayed and heavily corrupted by noise [10, 19]. As a result, decisions made by TD Learning are more reliable to optimize the agent's short-term performance. However, TD Learning can be misled when short-term gains disagree with the long-term goal. In this case, the agent can be attracted by carefully designed baits that lead it towards a trap. This is especially more challenging when using TD Learning to solve a continuous problem since approximation of the original problem can make predictions more complicated [13]. Such a drawback in TD Learning is mainly due to the nature of DPP, which solves the optimal control problem by combining a set of local sub-problems, and the fact that no overall physics model is considered to supervise the policy improvement in DPP.

In this work, we develop a fundamental methodology of solving the RL problem by using the stochastic maximum principle (SMP), which is a major alternative approach for solving the stochastic optimal control problem besides the DPP, and we will focus on the continuous time-space state model with noise perturbations. An important assumption we need for our SMP approach is that the environment is described by a physics model, and the unknown factors in the environment are characterized by model parameters. Although the capability of training an agent how to take actions without an explicit environment model is necessary for many RL applications, being able to take the rst principle into account is also important when applying RL to solve scientic problems with well-established physics knowledge.

As a fundamental mathematical eort, we will not discuss deep learning related deep reinforcement learning techniques (e.g. [25]), which might be more powerful but certainly will lead to several challenges caused by computational implementation of deep neural networks, such like the overtting issue, the representation capability, and the reliability in training. Although we don't consider the application of

deep learning in this work, appropriately designed deep learning methods can also be applied to our general computational framework and improve the eciency of our maximum principle approach.

For solving the classic stochastic optimal control problem, the SMP aims to nd the optimal control by optimizing a stochastic system called the Hamiltonian, and a gradient process with respect to control can be derived by applying the \Gâteaux derivative" to the adjoint process of the state dynamics in order to carry out gradient descent optimization [20]. The SMP approach has several advantages over the DPP approach. For example, it allows to have random coecients in the state equation and in the performance cost/reward, and it allows more state constraints { especially some nite dimensional terminal state constraints (see [28]). Moreover, the gradient process can give us a direction to improve the policy over the performance period. This is potentially more ecient compared with the DPP approach due to the application of gradient-based optimization. In addition, the gradient with respect to control derived under the SMP framework is based on the understanding of the \global" environment. In this way, the SMP designed policy can better balance between the short-term gains and the long-term goal. Therefore, the SMP approach can be more reliable than the DPP based methods, which typically rely on solutions of stacked local optimization problems.

On the other hand, the SMP approach has to face two challenges when solving the RL problem. First of all, although the gradient process with respect to control can be derived and formulated explicitly by using the adjoint of the state process, obtaining a numerical approximation for the adjoint state process, which is a backward stochastic dierential equation, is a challenging task. As a result, the optimization procedure for nding the optimal policy (i.e. the exploitation procedure) needs to evaluate the gradient repeatedly, which makes the SMP approach computationally expensive [11]. Secondly, the environment (i.e. the state model and the cost/reward model) in a RL problem is not completely known. Therefore, an environment estimation method for the purpose of exploration is needed while searching for the optimal policy.

To address the eciency issue in searching the optimal policy, we introduce a \backward action learning" (BAL) method for eciently solving the stochastic optimal control problem. The main theme of the BAL method is to apply a sample-wise numerical scheme to approximate the solution of the (backward) adjoint equation sample-by-sample and then adopt the methodology of stochastic approximation to carry out a stochastic gradient descent procedure to determine the optimal policy. In this way, we can avoid the high computational cost of solving backward stochastic dierential equations in the state space. At the same time, the mathematical expression of the gradient process, which contains solutions of the adjoint equation, can still be eectively utilized through the sample-wise approximator to search for the optimal policy.

To address the second issue and explore the environment, we assume that we have enough physics knowledge so that the environment can be formulated as a parameterized model. Then, exploring the

environment is equivalent to searching for the environment parameter, and therefore exploration can be achieved via parameter estimation. In this work, we apply the direct lter, which is an accurate and ecient online parameter estimation method [2], to estimate the environment parameter during the training procedure. Other online parameter estimation methods may also be used under our general SMP methodology.

Since an online parameter estimation method can dynamically provide feedbacks through training trials and generate real-time updates to improve the understanding of the environment, the estimated environment parameter can guide the BAL optimal control solver to exploit the optimal policy. To further explore the environment and balance between exploration and exploitation, we also perturb the policy by some articial noise and adopt an \-greedy" mechanism to encourage exploration [23].

The rest of this paper is organized as follows. In Section 2, we introduce the model-based RL problem with parameterized environment, and we shall provide a general methodology of using the SMP to solve the RL problem. In Section 3, we introduce the numerical algorithms that eciently solve the RL problem under the SMP framework. In Section 4, we present three numerical examples to demonstrate the eectiveness of our algorithm and the necessity of using the SMP approach to solve the RL problem in application scenarios. Some concluding remarks that summarize our research outcomes will be given in Section ??.

2. Problem setting and methodology

In this section, we rst introduce the formulation of model-based reinforcement learning (RL) with parameterized environment. Then, we shall introduce a direct lter method for learning the environment parameter and a stochastic maximum principle (SMP) type optimal control solver to nd the optimal policy.

2.1. Problem setting

In this work, we consider the mode-based RL problem under the complete ltered probability space $(; F; F; P)$, on which a d-dimensional standard Brownian motion W is dened such that $F := fF_s g_{s0}$ is the natural ltration of W augmented by all the P-null sets in F. The dynamics of the agent is modeled by the following stochastic dierential equation (SDE)

$$dX_t^{a;} = b(t; X_t^{a;}; a_t;)dt + (t; X_t ; a_t;)dW_t; \qquad t 2 [0; T]; \tag{1}$$

where $X_t^{a;}$, valued in R^d, is the state of the agent, called state process; a_t, valued in R^m, stands for the agent action at time t, called control process; $b : [0; T]R^d R^m R^q ! R^d$ and $: [0; T]R^d R^m R^q ! R^{dd}$ are suitable maps called drift and diusion, respectively; the d-dimensional standard Brownian

motion $W := \{W_t\}_{t\in[0,T]}$ introduces uncertainty to the system. In the model-based RL problem, we use b and σ to model the dynamics of the agent, and they contain the "physics knowledge" of the environment that we already possess. Since we assume that there are also unknowns in the environment, we introduce a parameter θ, valued in $\mathbb{R}^q$, to represent physics-informed unknown factors that we need to learn in the environment. In the RL problem, the control process is also called the "policy". Denote

$$\mathcal{U}[0,T] := \{a : [0,T] \times \Omega \to U \subset \mathbb{R}^m \text{ a is } \mathcal{F}\text{-progressively measurable}\}$$

as the admissible control set, in which we can choose the control actions. Under some mild conditions [28], for every choice of parameter $\theta \in \mathbb{R}^q$ and control $a \in \mathcal{U}[0,T]$, SDE (1) admits a unique solution $X^{a;\theta}$.

The performance of the action a is measured by the following cost (or reward):

$$J(a) = \mathbb{E}\left[\int_0^T f(t, X_t^{a;\theta}, a_t)dt + h(X_T^{a;\theta})\right], \tag{2}$$

where f is the running cost, which may contain some unknown factors that reect the environment, and h measures the cost at the terminal time T.

In this work, we let the "environment" in the RL problem be mathematically formulated by b, σ, f and h. The unknowns in the environment are represented by parameter θ, and we want to re-emphasize that being able to incorporate physics knowledge into a RL task is necessary in many practical scientic machine learning problems.

The goal of the stochastic optimal control problem is to nd the "optimal control" a^* that minimizes the cost J [1], i.e.

$$J(a^*) = \inf_{a\in\mathcal{U}[0,T]} J(a). \tag{3}$$

In the RL language, we aim to nd the optimal policy a^* that minimizes the "penalty". Similar framework can also search for the optimal policy that maximizes the "reward" by switching the minimization problem to maximization problem in Eq. (3).

When parameter θ is given, equations (1) - (3) formulate a classic stochastic optimal control problem. The major dierence that distinguishes the RL problem from the stochastic optimal control problem is the unknown environment, which is modeled by the physics-informed unknown parameter θ in this work.

As a numerical approach for solving the RL problem, our method will consist an exploration procedure and an exploitation procedure. Since the environment in this work is parameterized, the exploration procedure, which aims to determine the environment during learning, is equivalent to implementing online parameter estimation for θ. On the other hand, the exploitation procedure, which nds the

[1] In the case of J is a reward, we maximize the reward functional.

optimal policy, is equivalent to solving the stochastic optimal control problem. In what follows, we shall introduce a SMP based RL solver with online parameter estimation for learning the environment.

2.2. A stochastic maximum principle framework for the reinforcement learning problem

Since the major dierence between the RL problem and the classic optimal control problem is the exploration of the environment, nding the parameter that represents the environment while learning the optimal policy is a key challenge. In this work, we formulate the dynamical estimation of the environment parameter as an optimal ltering problem.

Exploration: Learning the environment by using the direct lter method

The main idea of the \optimal ltering" approach for dynamically estimating parameters is to \project" the data of the agent state to the parameter space and use the conditional probability density function (PDF) of the parameter, i.e. $p(_k jX_k)$, to calculate the estimate for at the k-th training episode, where -algebra $X_{tk} := X^{a\,|\,;\,|}; f^{\,|}(t; X_t; a_t); 0 \;\; t \;\; T; l = 0; 1; \; ; k$ contains the information of the state of the agent as well as the parameterized cost in the previous training episodes, and we assume an initial guess $_0$ for the environment. In this work, we apply the direct lter method [2] to estimate the environment parameter in the online manner.

To proceed, we use a sequence of random variables $f_k g_{k0}$ to represent our estimates for the unknown parameter , where $_k$ is the estimate corresponding to the k-th training episode. Assuming that we have $_k$ that follows the conditional PDF $p(_k jX_k)$, we generate a proposal parameter random variable, i.e. $_{k+1}$, through the following pseudo-dynamics:

$$_{k+1} = \; _k + \; _k; \qquad k = 0; 1; 2; \; ; \tag{4}$$

where $_k \;\; N(0; (_k)^2)$ is a standard Brownian motion with pre-chosen covariance constants $f_k g_{k0}$, and $_k$ gives articial noise that allows to explore other possible values of the environment parameter.

For a given conditional PDF $p(_k jX_k)$, which describes the estimated parameter at the k-th episode, we apply the following Bayesian inference to obtain the posterior PDF $p(_{k+1} jX_{k+1})$ that \optimally" describes the estimated parameter $_{k+1}$ corresponding to the state information of the $k + 1$-th episode as follows

$$p(_{k+1} jX_{k+1}) \;\; p(_{k+1} jX_k) p(X^{a;}; f j_{k+1}); \tag{5}$$

where $p(_{k+1} jX_k)$ is the prior conditional PDF of $_{k+1}$ derived from the pseudo-dynamics (4) and the parameter distribution $p(_k jX_k)$ at the training episode k, and $p(X^{a;}; f j_{k+1})$ is the likelihood function that compares the simulated agent state and its corresponding cost derived from the prior parameter variable $_{k+1}$ with the real agent state $X^{a;}$ and the real cost f produced by the true environment parameter .

Exploitation: The stochastic maximum principle approach for searching the optimal policy

In the case that the complete knowledge of the environment (or the environment parameter is known), and the optimal strategy a is in the interior of U [2], we can deduce by using the Gâteaux derivative of a and the maximum principle that the gradient process of the cost functional J with respect to the control process over time interval $t \in [0, T]$ has the following form [11, 28]

$$r J_a(a_t) = E\left[b_a(t, X_t^{a,}; a_t,)^\top Y_t^{a,} + {}_a(t, X^{a,}; a_t,)^\top Z_t^{a,} + f_a(t, X^{a,}; a_t)^\top \right]_t \tag{6}$$

where subscripts are used to denote partial derivatives of functions. The stochastic processes Y and Z are adapted solutions of the following forward backward stochastic dierential equations (FBSDE) system

$$\begin{aligned} dX_t^{a,} &= b(t, X_t^{a,}; a_t,)dt + (t, X^{a,}; a_t,)dW_t, && \text{(forward SDE)} \\ dY_t^{a,} &= -b_x(t, X_t^{a,}; a_t,)^\top Y_t^{a,} - {}_x(t, X^{a,}; a_t,)^\top Z_t^{a,} - f_x(t, X_t^{a,}; a_t)^\top dt \\ &\quad + Z_t^{a,} dW_t, \qquad Y_T^{a,} = h_x(X_T^{a,})^\top, && \text{(BSDE)} \end{aligned} \tag{7}$$

where the rst equation in (7) is a standard forward stochastic dierential equation (SDE) with the same expression as the state equation (1) for the agent, and the second equation is a backward stochastic dierential equation (BSDE), which is also call the \adjoint equation" of the state equation. The solution Z of the BSDE is the martingale representation of Y with respect to the Brownian motion W.

With gradient $r J_a$ introduced in Eq. (6) and solutions X, Y, and Z introduced in the FBSDE system (7), the SMP approach for solving the stochastic optimal control problem will carry out the following gradient descent optimization procedure to solve for the optimal control, i.e. the optimal policy in RL,

$$a^{k+1} = a^k - {}_k r J_a(a^k), \qquad k = 0, 1, 2, \ldots, K - 1, \tag{8}$$

where we have an initial guess policy a^0, ${}_k$ is the learning rate, K is a pre-determined total number of training episodes, and we let a^K be our estimated optimal policy.

The general methodology of solving the RL problem via the SMP

In what follows, we briey discuss the general methodology about how to solve the RL problem under the SMP framework. At this moment, we ignore computational implementation issues, and the numerical algorithms will be introduced in the next section.

The main theme of the SMP solver for RL is to use the iterative scheme (8) to improve the policy

[2] In many practical RL scenarios, people often let U be the real space.

and use the direct lter to explore the environment. Assume that with k episodes of training, we have an estimated environment parameter $_k$ (with its conditional PDF $p(_k jX_k)$) and a sub-optimal policy a^k based on the understanding of the environment corresponding to $_k$.

For the k+1-th episode, we use scheme (4) to generate a prior conditional PDF $p(_{k+1}jX_k)$ that characterizes the proposal parameter. Then, we let the agent follow the current policy a^k and interact with the real environment to obtain a state process $X^{a^k;}$, which can be used to compare with simulated state processes to generate the likelihood $p(X_t^{a^k;}j_{k+1})$, and we can obtain the posterior PDF $p(_{k+1}jX_{k+1})$ for the estimated parameter variable $_{k+1}$ via the Bayesian inference (5). Note that the agent trial state process, i.e. $X^{a^k;}$, provides the feedback from the real environment for exploration. With an updated estimate for the environment at the learning episode k+1, we apply the SMP method to nd the optimal policy. Specically, we compute the solutions of the FBSDE system (7) with the estimated policy a^k and the environment parameter $_{k+1}$ to obtain approximations for $X^{a^k;_{k+1}}$, $Y^{a^k;_{k+1}}$ and $Z^{a^k;_{k+1}}$. Then, the gradient rJ_a introduced in Eq. (6) can be calculated by using the simulated solutions of the FBSDE system (7). As a result, the approximated gradient would give us a direction to improve the current policy, and we can obtain the improved policy a^{k+1} via the gradient descent scheme (8). To encourage exploration, we also adopt the \-greedy" method by perturbing the sub-optimal policy with some articial noise.

3. Numerical algorithms

In this section, we derive numerical algorithms to implement the above SMP approach for the RL problem. For convenience of presentation, we assume that the diusion coecient in the state dynamics (1) is a deterministic time-dependent process, denoted by $_t$. Algorithms for more general cases can be obtained under our methodology with more tedious derivation, and the model with the simplied diusion coecient can already cover wide range of application problems since the physics knowledge of a stochastic model is often incorporated into the drift term. In what follows, we shall rst discuss numerical implementation of the direct lter method for exploration (Section 3.1), and we will provide a backward action learning (BAL) method for solving the classic stochastic optimal control problem under the SMP framework for the purpose of exploitation (Section 3.2). Then, we combine exploration with exploitation and introduce an overarching algorithm to solve the RL problem (Section 3.3). A pseudo-algorithm that summarizes our SMP approach for the RL problem will be provided at the end of this section (Section 3.4).

3.1. Particle implementation of the direct lter method for exploration

In this paper, we adopt the numerical recipe of the \particle lter" [6, 12], which is also known as a \sequential Monte Carlo method", to implement the direct lter for exploring the environment in the

RL problem.

Assume that after k training episodes, we have a (suboptimal) policy a^k and a set of Q samples (called "particles" in the particle filter), denoted by $\{\theta_k^{(q)}\}_{q=1}^{Q}$, that follow the conditional PDF $p(\theta_k|X_k)$ for the estimated parameter θ_k. To carry out the exploration procedure in the $k+1$-th episode, we first generate a set of particles based on the pseudo-dynamics introduced in Eq. (4) to generate a set of proposal particles, $\{\tilde{\theta}_{k+1}^{(q)}\}_{q=1}^{Q}$, as follows:

$$\tilde{\theta}_{k+1}^{(q)} = \theta_k^{(q)} + \varepsilon_k^{(q)}, \qquad q = 1, 2, \ldots, Q, \tag{9}$$

where $\varepsilon_k^{(q)} \sim N(0, \sigma_k^2)$, and the particle set $\{\theta_{k+1}^{(q)}\}_{q=1}^{Q}$ formulates the empirical distribution, i.e. $\hat{p}(\theta_{k+1}|X_k) := \frac{1}{Q}\sum_{q=1}^{Q} \delta_{\theta_{k+1}^{(q)}}(\theta_{k+1})$, for the prior PDF $p(\theta_{k+1}|X_k)$.

To incorporate the trial agent state at the $k+1$-th training episode and learn the environment, we update the proposal particles through Bayesian inference. Specifically, for each proposal particle $\tilde{\theta}_{k+1}^{(q)}$, we generate a simulated state trajectory $\tilde{X}^{a^k;\tilde{\theta}_{k+1}^{(q)}} := \{X_t^{a^k;\tilde{\theta}_{k+1}^{(q)}}\}_{0\le t\le T}$ based on the current policy a^k and the proposal particle $\tilde{\theta}_{k+1}^{(q)}$. On the other hand, the agent that follows policy a^k interacts with the real environment, and it generates the real state trajectory $X^{a^k;\theta}$ that reflects the true environment parameter θ. Then, by comparing each simulated state sample trajectory $X^{a^k;\tilde{\theta}_{k+1}^{(q)}}$ with the real agent state $X^{a^k;\theta}$, we have the following (unnormalized) likelihood for the parameter particle $\tilde{\theta}_{k+1}^{(q)}$ as

$$p(X^{a^k;\theta}; f_{k+1}^{(q)}) \propto \exp\left\{ \int_0^T \left[\int_0^T f_{\tilde{\theta}_{k+1}^{(q)}}(t, X_t^{a^k;\tilde{\theta}_{k+1}^{(q)}}; a^k)\, dt - \int_0^T f(t, X_t^{a^k;\theta}; a^k)\, dt \right]^2 / 2\sigma_k^2 \right\}, \tag{10}$$

where σ_k is the standard deviation of the artificial noise in the pseudo parameter process that encourages exploration.

Then, by combining the prior with the above likelihood through Bayesian inference, we have

$$p_{\theta_{k+1}}(\tilde{\theta}_{k+1}^{(q)}|X_{k+1}) = \frac{\hat{p}(\theta_{k+1}|X_k)\, p(X^{a^k;\theta}; f^{(q)})}{C} \tilde{\theta}_{k+1}, \qquad q = 1, 2, \ldots, Q, \tag{11}$$

where C is a normalization factor.

Since the empirical prior distribution $\hat{p}(\theta_{k+1}|X_k)$ is described by a set of (unweighted) prediction particles, the weighted particle pairs $\{(\tilde{\theta}_{k+1}^{(q)}, \omega_{k+1}^{(q)})\}_{q=1}^{Q}$ obtained in Eq. (11) can describe the (weighted) empirical distribution for $p_{\theta_{k+1}|X_{k+1}}$, where the weight $\omega_{k+1}^{(q)} := p(X^{a^k;\theta}; f_{\tilde{\theta}_{k+1}^{(q)}})/C$ is the likelihood of each parameter particle. To improve the stability and address the degeneracy of the particles, i.e. only a very small number of particles have significant likelihood weights [1, 8, 17], we also introduce a

classic bootstrap resampling procedure by using the importance sampling method [12] for the weighted particle pairs $\{\tilde{x}_{k+1}^{(q)}, \omega_{k+1}^{(q)}\}_{q=1}^{Q}$ and obtain a set of equally weighted particles $\{x_{k+1}^{(q)}\}_{q=1}^{Q}$, which follow the conditional PDF $p(x_{k+1}|\mathcal{X}_{k+1})$ as needed for the next exploration procedure in the next training episode.

Remark 3.1. Note that the simulated state trajectories $\{\mathcal{X}_{k+1}^{(q)}\}_{q=1}^{Q}$ need to be calculated on discrete temporal points with appropriate numerical schemes. Since the state of the agent coincides with the forward SDE in the FBSDE system, we shall postpone our discussion on the numerical method for the state dynamics in the next subsection when we introduce the numerical method for solving FBSDEs.

Remark 3.2. In the case that the environment parameter is state-dependent, we let be a vector corresponding to agent states and carry out the direct lter method to estimate the parameter if the agent enters the correspondent state block. Numerical experiments that demonstrate this scenario will be presented in Section 4.

3.2. Backward action learning for exploitation

To introduce the numerical algorithm for exploitation, which is equivalent to solving a stochastic optimal control problem, we rst assume that we have complete knowledge of the environment, i.e. the environment parameter is known. Then, we shall combine the direct lter method for exploration with the backward action learning method for exploitation in the next subsection.

The computational framework of our backward action learning method is to carry out a gradient descent optimization procedure with iterative scheme (8) to improve the policy, and the gradient with respect to the policy, which is introduced in Eq. (6), is derived based on the SMP with usage of the Gâteaux derivative. Since the gradient is composed of solutions of the FBSDE system, numerical methods for solving FBSDEs are needed.

Numerical solvers for both forward SDEs and backward SDEs have been well studied [7, 15, 31]. In what follows, we introduce the standard numerical schemes for solving SDEs and BSDEs.

To proceed, we introduce a temporal partition

$$\pi_{N_T} := \{t_n : 0 = t_0 < t_1 < \cdots < t_n < \cdots < t_{N_T} = T\};$$

and we consider the following FBSDE system (7) over the time interval $[t_n; t_{n+1}]$ with a policy a^k and a given environment parameter , which needs to be estimated by the direct lter based exploration

procedure in the RL problem,

$$
\begin{aligned}
X^{a^k;}_{t_{n+1}} &= X^{a^k;}_{t_n} + \int_{t_n}^{t_{n+1}} b(t, X^{a^k;}_t, a^k;)\,dt + \int_{t_n}^{t_{n+1}} {}_t\,dW_t; && \text{(forward SDE)} \\
Y^{a^k;}_{t_n} &= Y^{a^k;}_{t_{n+1}} + \int_{t_n}^{t_{n+1}} b_x(t, X^{a^k;}_t, a^k;)^{>} Y^{a^k;}_t + f(t, X^{a^k;}_t, a^k)^{>}\,dt \\
&\quad - \int_{t_n}^{t_{n+1}} Z^{a^k;}_t\,dW_t: && \text{(BSDE)}
\end{aligned}
\tag{12}
$$

For a random variable $X^{a^k;}_{t_n}$ that represents the solution of the forward SDE at time t_n, we can approximate $X^{a^k;}_{t_{n+1}}$ by using the following standard Euler-Maruyama scheme

$$
X^{a^k;}_{t_{n+1}} \; X^{a^k;}_{t_n} + b(t_n, X^{a^k;}_{t_n}, a^k_{t_n};)t_n + {}_{t_n} W_{t_n}; \tag{13}
$$

where $t_n := t_{n+1} \quad t_n$ and $W_{t_n} := W_{t_{n+1}} \quad W_{t_n}$.

To solve the BSDE and obtain a numerical solution for Y, we take conditional expectation $E^{X^k}_n[] := E[j X^{a^k;}_{t_n}]$ on both sides of the BSDE in Eq. (12). Since the BSDE is the adjoint equation of the forward state equation, which is backward in time, we assume that a random variable $Y^{a^k;}_{t_{n+1}}$ representing the solution of the BSDE at time t_{n+1} is given, and we use the right-point formula to approximate the deterministic integral on the right hand side of the BSDE. Then, we obtain the following approximation scheme for Y

$$
Y^{a^k;}_{t_n} \; E^{X^k}_n Y^{a^k;}_{t_{n+1}} + E^{X^k}_n \Big[b_x(t_{n+1}, X^{a^k;}_{t_{n+1}}, a_{t_{n+1}};)^{>} Y^{a^k;}_{t_{n+1}} + f(t_{n+1}, X^{a^k;}_{t_{n+1}}, a_{t_{n+1}})^{>} t_n; \Big]_k \tag{14}
$$

where we have used the martingale property of Itô type stochastic integrals to get $E^{X^k}_n \Big[\int_{t_n}^{t_{n+1}} Z^{a^k;}_t\,dW_t \Big] = 0$, and note that $Y^{a^k;}_{t_n} = E^{X^k}_n [Y^{a^k;}_{t_n}]$ due to the adaptedness of Y with respect to X.

By using approximation equations (13)-(14) as a guideline, we introduce the following (temporal-discretized) scheme for solving the FBSDE system (see [4, 29]):

$$
\begin{aligned}
X^{a^k;}_{n+1} &= X^{a^k;}_n + b(t_n, X^{a^k;}_n, a^k;)t_n + {}_{t_n} W_{t_n}; \\
Y^{a^k;}_n &= E^{X^k}_n Y^{a^k;}_{n+1} + E^{X^k}_n \Big[b_x(t_{n+1}, Y^{a^k;}_{n+1}, a_{t_{n+1}};)^{>} X^{a^k;}_{n+1} + f(t_{n+1}, X^{a^k;}_{n+1}, a_{t_{n+1}})^{>} t_n; \Big]_k
\end{aligned}
\tag{15}
$$

where $X^{a^k;}_{n+1}$ and $Y^{a^k;}_n$ are approximations for $X^{a^k;}_{t_{n+1}}$ and $Y^{a^k;}_{t_n}$ with an estimated policy a^k and an environment parameter . The side condition of the BSDE is $Y^{a^k;}_{N_T} = h_x$, where h is the terminal cost (i.e. penalty), and X is initialized with the initial state of the agent. With scheme (15), we use numerical solutions $X^{a^k;}_n$ and $Y^{a^k;}_n$ to approximate X and Y in the gradient process and get the

following approximation scheme for ∇J_a:

$$\nabla J_a(a^k_{t_n}) \approx \nabla J_a(\hat{a}^k_{t_n}) = E\left[b_a(t_n, X^{a^k}_n, a^k_{t_n})^\top Y^{a^k}_n + f_a(t_n, X^{a^k}_n, a^k_{t_n})^\top\right], \quad n = 0, 1, \ldots, N_T - 1. \tag{16}$$

Then, the iterative scheme for finding the optimal control, i.e. the optimal policy in the RL problem, becomes

$$a^{k+1}_{t_n} = a^k_{t_n} - \alpha_k \nabla J_a(\hat{a}^k_{t_n}), \qquad n = 0, 1, 2, \ldots, N_T - 1. \tag{17}$$

To implement the gradient descent iteration (17) with the gradient fully calculated by the approximation scheme (16), one needs to evaluate the expectation $E[\cdot]$ in Eq. (16) as well as the conditional expectation $E^{X^k_n}[\cdot]$ in scheme (15), which is needed to approximate the numerical solutions $X^{a^k}_{n+1}$ and $Y^{a^k}_n$. The standard approach to evaluate an expectation (especially in high-dimensional spaces) is the Monte Carlo method, in which we use simulated Monte Carlo samples to represent the random variables and use the average of Monte Carlo samples as an approximation to the desired expected value. However, when utilizing the Monte Carlo method in gradient descent optimization with numerical solution $Y^{a^k}_n$ of the adjoint BSDE, in addition to simulating the expectation for the expected gradient ∇J_a with Monte Carlo samples, one also needs to generate a large number of state samples for X at each time step t_n in order to evaluate the conditional expectation $E^{X^k_n}$ in the numerical scheme (15) for the calculation of $Y^{a^k}_n$. Since $Y^{a^k}_t$ is a random variable whose value is corresponding to the state $X^{a^k}_t$, which is also a random variable that continuously takes values in the state space $\mathbb{R}^d$, a Monte Carlo type representation of Y with a set of random samples requires numerical approximation for Y in the entire state space. This would cause a very challenging computational task of high-dimensional approximation when the state dimension d is large, which is often computationally prohibitive due to the so-called "curse of dimensionality". Moreover, the numerical approximation for expectation E and conditional expectation $E^{X^k_n}$ needs to be calculated repeatedly over the gradient descent iteration procedure, which will make the full calculation of the gradient descent optimization procedure infeasible in practice.

In our backward action learning approach for the stochastic optimal control problem, we use a single realization of Monte Carlo sample (or a mini-batch of samples) in the Monte Carlo approximation to represent the entire state of the random variable in an expectation. Specically, at each iteration stage k we use the Euler-Maruyama scheme to generate one sample-path $\{\tilde{X}^{a^k}_{n+1}\}_{n=1}^{N_T}$ for the state process as follows

$$\tilde{X}^{a^k}_{n+1} = \tilde{X}^{a^k}_n + b(t_n, \tilde{X}^{a^k}_n, a_{t_n}, {}^k)\Delta t_n + \sigma_{t_n}\sqrt{\Delta t_n}\,\xi_n, \qquad n = 0, 1, 2, \ldots, N_T - 1, \tag{18}$$

where ξ_n is a random sample drawn from the standard Gaussian distribution, and $\tilde{X}^{a^k}_0 = X_0$ is the initial state of the agent at time $t = 0$. Note that the single-realization representation of the state process

coincides with an agent trial in the training procedure under the RL framework.

When solving the BSDE, we use the single-realization of the state sample $\{\tilde{X}^{a^k}_{n+1}, g\}_{n=1}^{N_T}$ generated by (18) to represent the state process in the FBSDE system. In this way, we rewrite the numerical scheme for the BSDE and obtain the following sample-wise approximation for the adjoint process Y

$$\tilde{Y}^{a^k}_n = \tilde{Y}^{a^k}_{n+1} + b_x(t_{n+1}, X^{a^k}_{n+1}, a^k_{t_{n+1}})^\top \tilde{Y}^{a^k}_{n+1} \Delta t_n + f_x(t_{n+1}, X^{a^k}_{n+1}, a^k_{t_{n+1}})^\top \Delta t_n, \qquad n = N_T - 1, \ldots, 0, \tag{19}$$

where $\tilde{Y}^{a^k}_n$ and $\tilde{Y}^{a^k}_{n+1}$ are approximations for $Y^{a^k}_{t_n}$ and $Y^{a^k}_{t_{n+1}}$ corresponding to the state samples $X^{a^k}_n$ and $X^{a^k}_{n+1}$, respectively, i.e. $Y^{a^k}_{t_n} = \tilde{Y}^{a^k}_n(X^{a^k}_n)$ and $\tilde{Y}^{a^k}_{n+1} = \tilde{Y}^{a^k}_{n+1}(X^{a^k}_{n+1})$, and we have used stochastic approximation to approximate conditional expectations in Eq. (15) as follows:

$$\tilde{Y}^{a^k}_{n+1}(X^{a^k}_{n+1}) \approx E_n[\tilde{Y}^{a^k}_{n+1}],$$

and

$$E_n\left[b_x(t_{n+1}, X^{a^k}_{n+1}, a^k_{t_{n+1}})^\top \tilde{Y}^{a^k}_{n+1} \Delta t_n + f_x(t_{n+1}, X^{a^k}_{n+1}, a^k_{t_{n+1}})^\top \Delta t_n\right].$$

Then, we use sample-wise approximations introduced in (18) and (19) to represent the stochastic processes X and Y in the gradient ∇J_a and get the following sample-wise approximation for the gradient

$$\nabla \tilde{J}_a(a^k_{t_n}) = b_a(t_n, X^{a^k}_n, a^k_{t_n})^\top \tilde{Y}^{a^k}_n + f_a(t_n, X^{a^k}_n, a^k_{t_n})^\top \Delta t_n. \tag{20}$$

As a result, the fully calculated gradient descent scheme (17) becomes the following stochastic gradient descent scheme

$$a^{k+1}_{t_n} = a^k_{t_n} - \alpha_k \nabla \tilde{J}_a(a^k_{t_n}), \qquad n = 0, 1, 2, \ldots, N_T - 1, \qquad k = 0, 1, 2, \ldots, K - 1. \tag{21}$$

Although the sample-wise approximator $\{\tilde{Y}^{a^k}_n, g\}_{n=1}^{N_T}$ cannot provide a comprehensive representation for the adjoint process $\{Y^{a^k}_{t_n}, g\}_{n=1}^{N_T}$ in the entire state space since the conditional expectations are only approximated by single-realization of state samples, it's important to point out that Y only appears in the gradient process under expectation, and the primary contribution of Y is to incorporate the differential dynamics of the adjoint process into the gradient and guide the search of the optimal control (i.e. policy). Therefore, our sample-wise approximation for Y can already embed information of the differential dynamics into the gradient process, and the rationale of applying stochastic approximation in stochastic gradient descent can be used to justify the sample-wise solver for the FBSDE system.

In this work, we name the sample-wise numerical solver introduced by schemes (18) - (21) the "backward action learning (BAL)" method for solving the stochastic optimal control problem, and such

a BAL framework constitutes the key mechanism of our SMP type approach for policy improvement in reinforcement learning.

3.3. Combine direct lter with backward action learning for solving the reinforcement learning problem

Now, we combine the direct lter based exploration method with the BAL based exploitation method and construct a numerical algorithm to solve the RL problem. Since the key of the numerical recipe of our SMP approach for solving the RL problem is the BAL method for the stochastic optimal control problem, in the rest of this paper we will also call our SMP based RL solver \the BAL method" for convenience of presentation.

To proceed, we assume that we can instantly receive the state of the agent during the training procedure. With online reception of the agent state, instead of applying the direct lter to estimate the environment parameter after each training episode to update our understanding of the environment with the information of the entire agent trial trajectory (as we introduced in Section 3.1), in the BAL algorithm for solving the RL problem we implement the direct lter dynamically at each time step in each training episode. This time-step parameter estimation implementation will allow us to better utilize the state information of the agent. As a result, we can more frequently update the environment information and therefore more suciently explore the environment.

Specically, we let $\{\theta^{(q)}_{k+1;t_0}\}_{q=1}^{Q} := \{\theta^{(q)}_{k}\}_{q=1}^{Q}$ be the initial parameter particles at the beginning of the $k+1$-th training episode, where $\{\theta^{(q)}_{k}\}_{q=1}^{Q}$ formulates an empirical distribution for the conditional PDF $p(\theta|X_k)$ of the environment parameter after the k-th training episode. Assuming that we have a set of (equally-weighted) parameter particles $\{\theta^{(q)}_{k+1;t_n}\}_{q=1}^{Q}$ at time step t_n under the temporal partition $\mathcal{N}_T$, we use the following zero-dynamics scheme, which is similar (9), to generate a set of predicted parameter particles $\{\tilde{\theta}^{(q)}_{k+1;t_{n+1}}\}_{q=1}^{Q}$ for time step t_{n+1} in the $k+1$-th training episode

$$\tilde{\theta}^{(q)}_{k+1;t_{n+1}} = \theta^{(q)}_{k+1;t_n} + \theta^{(q)}_{k+1;t_n}; \qquad q = 1; 2; \ldots; Q: \tag{22}$$

Then, we discretize the state equation at time step t_n corresponding to the parameter particles as follows

$$\tilde{X}^{a^k;\tilde{\theta}^{(q)}_{k+1;t_{n+1}}}_{n+1} = \tilde{X}^{a^k;\tilde{\theta}^{(q)}_{k+1;t_n}}_{n} + b(t_n; \tilde{X}^{a^k;\tilde{\theta}^{(q)}_{k+1;t_n}}_{n}; a_{t_n}; \theta^{(q)}_{k+1;t_{n+1}})t_n + \sqrt{t_n}\,\bar{\sigma}^{(q)}; \quad \bar{\sigma}_n \qquad q = 1; 2; \ldots; Q: \tag{23}$$

Note that the Gaussian random samples $\{\theta^{(q)}_{k+1;t_n}\}_{q=1}^{Q}$ add articial noise to the parameter particles $\{\tilde{\theta}^{(q)}_{k+1;t_{n+1}}\}_{q=1}^{Q}$, and it can provide a natural mechanism that encourages the agent to explore the environment. Then, we use $X^{a^k;}$ to denote the trajectory of the $k+1$-th agent trial, which interacts with the real environment, and we compare the simulated state-parameter samples $\tilde{X}^{a^{(q)};\tilde{\theta}^{(q)}_{k+1;t_{n+1}}}_{n+1}$ (generated

by Eq. (23)) with the real agent state $X_{t_{n+1}}^{a^k;}$ at time t_{n+1} to derive the following likelihood value for each predicted parameter particle $\tilde{}_{k+1;t_{n+1}}^{(q)}$

$$p(X_{t_{n+1}}^{a^k;}; f(t_{n+1}; X_{t_{n+1}}^{a^k;}; a^k)_{k+1;t_{n+1}}^{\tilde{}(q)})$$

$$\exp\left(\frac{X_{t_{n+1}}^{a^k;} \quad X_{n+1}^{a^k;\tilde{}(q)}{}_{k+1;t_{n+1}}{}^2 + f(t_{n+1}; X_{t_{n+1}}^{a^k;}; a^k) \quad f^{\tilde{}(q)}_{k+1;t_{n+1}}(t_{n+1}; X_{n+1}^{a^k;\tilde{}(q)}{}_{k+1;t_{n+1}}; a^k)^2}{2\ ^2{}_k + (t\ t)^2_n}\right); \tag{24}$$

where the likelihood $p(X_{t_{n+1}}^{a^k;}; f(t_{n+1}; X_{t_{n+1}}^{a^k;}; a^k)_{k+1;t_{n+1}}^{\tilde{}(q)})$ focuses on the comparison of the agent state at time t_{n+1}, which plays the same role as the likelihood for the entire agent trajectory introduced in Eq. (10). As a result, the particle-weight pairs $f_{k+1;t_{n+1}}^{\tilde{}(q)}; !_{k+1;t_{n+1}}^{(q)} g_{q=1}^{Q}$, where the weight value is assigned as

$$!_{k+1;t_{n+1}}^{(q)} = p(X_{t_{n+1}}^{a^k;}; f(t_{n+1}; X_{t_{n+1}}^{a^k;}; a^k)_{k+1;t_{n+1}}^{\tilde{}(q)})=C$$

with an appropriate normalization factor C, form a weighted empirical distribution for the poste-rior distribution of the parameter at time instant t_{n+1} in the $k+1$-th training episode. To avoid the degeneracy issue of the particle lter, we resample all the particles based on the weighted pairs $f_{k+1;t_{n+1}}^{\tilde{}(q)}; !_{k+1;t_{n+1}}^{(q)} g_{q=1}^{Q}$ and generate a set of equally-weighted particles $f_{k+1;t_{n+1}}^{(q)} g_{q=1}^{Q}$ for the next time step.

In this way, at the terminal time T we have carried out $N_T \quad 1$ times parameter estimation procedure, which can eectively incorporate the information of the agent state at each time instant in the $k+1$-th training episode, and the particle set $f_{k+1}^{(q)} g_{q=1}^{Q} := f_{k+1;t_{N_T}}^{(q)} g_{q=1}^{Q}$ provides our \best" understanding of the environment after considering the state of the $k+1$-th agent trial. Then, we use the mean of the parameter particles $f_{k+1}^{(q)} g_{q=1}^{Q}$, denoted by $_{k+1}$, as our estimate for the environment parameter in the $k+1$-th training episode.

With the updated estimate $_{k+1}$ for the environment parameter, we follow the BAL algorithm discussed in Section 3.2 to improve the policy in the exploitation procedure. Recall that the main theme of the BAL method is to use a single-realization of the state trajectory as a stochastic approximation to the state process when approximating expectations. Since the real agent trajectory $X^{a^k;}$ that follows policy a^k already provides a path of the agent state, we can use the real agent trial state at the temporal partition points, i.e. $fX_{t_n}^{a^k;} g_{n=1}^{N_T}$, to replace the single-realization simulated state trajectory $f\tilde{X}_n^{a^k;} g_{n=1}^{N_T}$ (introduced in Eq. (18)) for the BAL optimal control solver. On the other hand, the adjoint equation, i.e., the BSDE, is still driven by the current estimated parameter $_{k+1}$ except that the forward process is replaced by the real agent state. In this way, we introduce the following sample-wise solution for the

adjoint BSDE in the $k+1$-th training episode:

$$\begin{aligned} Y_n^{a^k;\,k+1} &= Y_{n+1}^{a^k;\,k+1} + b_x(t_{n+1}; X_{t_{n+1}}^{a^k;\,k}; a_{t_{n+1}}^{k}; \theta^{k+1})^\top Y_{n+1}^{a^k;\,k+1} \\ &\quad + f_x^{k+1}(t_{n+1}; X_{t_{n+1}}^{a^k;\,k}; a_{t_{n+1}}^{k})^\top t_n; \qquad n = N_T - 1; \ldots; 0. \end{aligned} \tag{25}$$

Note that we have used θ^{k+1} as our estimated environment parameter in b and f when θ is explicitly needed in the numerical scheme for the BSDE, and the single-realization representation of the forward SDE is given by the real agent trial trajectory $\{X_{t_n}^{a^k;}\}_{n=1}^{N_T}$.

Then, we derive the sample-wise approximation for the gradient as follows

$$\nabla J_a(a_{t_n}^k) = b_a(t_n; X_{t_n}^{a^k;}; a_{t_n}^k; \theta^{k+1})^\top Y_n^{a^k;\,k+1} + f_a^{k+1}(t_n; X_{t_n}^{a^k;}; a_{t_n}^k)^\top t_n, \tag{26}$$

and we carry out the following stochastic gradient descent iteration to improve the policy

$$a_{t_n}^{k+1} = a_{t_n}^k - \alpha_k \nabla J_a(a_{t_n}^k); \qquad n = 0; 1; 2; \ldots; N_T - 1. \tag{27}$$

Different from the classic stochastic optimal control problem, which aims to find the optimal control with an explicitly given environment, the agent in the RL problem needs to explore the environment corresponding to the state space. In the above BAL approach for solving the RL problem, the exploration is implemented through the parameter estimation procedure, and the artificial noise added to the pseudo parameter dynamics (as described in Eq. (22)) allows the agent to explore. In order to explore more actively and detect other possibilities, we adopt the ε-greedy type exploration algorithm and provide a mechanism for the agent to randomly explore the environment.

Specifically, for the suboptimal policy a^k that we obtained in the k-th training episode, we let the updated policy for the $k+1$-th training episode as follows

$$a_{t_n}^{k+1} = \begin{cases} a_{t_n}^{k+1}; & \text{with probability } 1-\varepsilon \\ a_{t_n}^{k+1} + \xi_n & \text{with probability } \varepsilon \end{cases}; \qquad n = 0; 1; 2; \ldots; N_T - 1; \tag{28}$$

where $\xi_n \sim N(0; \sigma)$ is a user defined noise level that brings random perturbations to the policy with the size of a pre-determined covariance σ, and $0 < \varepsilon < 1$ is the probability of implementing the enhanced policy exploration.

It's important to point out that the estimated optimal policy a^K obtained through the above exploratory stochastic gradient descent optimization procedure only gives a deterministic policy process, which is based on the initial state $X_{t_0} = X_0$. In this way, only the policy at time $t_0 = 0$, i.e. $a_{t_0}^K = a_{t_0}^K(X_{t_0})$, is a time/state-dependent action that reflects the actual state-to-action map, and the policy process

beyond time t_0 only gives an estimate based on the expected future behavior of the agent. In order to let the agent take optimal actions based on its current state beyond the initial time t_0 as desired in the RL problem, we need to screen the state space and calculate the optimal policy corresponding to the state at any time instant t_n. In what follows, we shall modify the BAL method to generate a time/state-dependent optimal policy process.

Note that the state of the agent at time t_{n+1} only depends on the previous states and actions. Therefore, we don't need to search the optimal policy in the entire state space. At the initial time t_0, assume that we have calculated the policy process $\{a^K_{t_n}\}_{n=0}^{N_T-1}$ through schemes (25) - (28) based on the estimated environment parameter obtained by the direct filter method. With the initial state X_0 and the optimal action at time t_0, i.e. $a^K_{t_0} = a^K_{t_0}(X_{t_0})$, we carry out the following Euler-Maruyama scheme to generate M samples for $X^{a^K}_{t_1}$:

$$\tilde{X}^{a^K;(m)}_{t_1} = X_{t_0} + b(t_n, X_{t_0}, a^K_{t_0}(X_{t_0}))t_0 + \sqrt{t_0}\,\xi_0^{(m)}, \quad \xi_0 \qquad m = 1, 2, \ldots, M, \tag{29}$$

where $\{\xi_0^{(m)}\}_{m=1}^M$ are M samples drawn from the standard Gaussian distribution, and the state samples $\{\tilde{X}^{a^K;(m)}_{t_1}\}_{m=1}^M$ characterize the state variable $X^{a^K}_{t_1}$ in the state space. Then, we let D_{t_1} be the region in the state space that covers all the state samples $\{\tilde{X}^{a^K;(m)}_{t_1}\}_{m=1}^M$. Apparently, if we run Eq.(29) with a large enough number M, the simulated samples $\{\tilde{X}^{a^K;(m)}_{t_1}\}_{m=1}^M$ would provide very reliable predictions for the future state $X^{a^K}_{t_1}$, and the corresponding region D_{t_1} would be large enough to cover the agent state at time t_1 if the agent takes the optimal action at time t_0. In this way, actions corresponding to state points in the region D_{t_1} at time t_1 are needed. To proceed, we introduce a set of state points, denoted by X_{t_1}, as a spatial discretization for the state region D_{t_1}. Then, we carry out the same BAL algorithm (25) - (28) from initial time t_1 to terminal time t_{N_T}, and the initial state is chosen among the state points in X_{t_1}, i.e. $X^{a^K}_{t_1} = x \in X_{t_1}$. As a result, we obtain a set of state-dependent optimal actions $\{a^K_{t_1}(x)\}_{x\in X_{t_1}}$ at time instant t_1, and we use $\{a^K_{t_1}(x)\}_{x\in X_{t_1}}$ as our policy variable (or the policy table) to guide the agent at time t_1. Similarly, following the above procedure, if we start from state points in X_{t_n} with their optimal actions $\{a^K_{t_n}(x)\}_{x\in X_{t_n}}$, we can determine the state region $D_{t_{n+1}}$ and the state points $X_{t_{n+1}}$ at time t_{n+1}. Then, we can compute the optimal policy $\{a^K_{t_{n+1}}(x)\}_{x\in X_{t_{n+1}}}$ corresponding to the state points in $X_{t_{n+1}}$ by using the BAL method. As a result, we can adaptively calculate the optimal policy over the state space along the temporal partition $_{N_T}$.

We also want to point out that indeed the training procedure for the optimal policy needs to be repeated carried out step-by-step in time, the direct filter based exploration can provide good understanding of the environment through the exploration procedure even at the first time instant t_0 since the agent trial trajectories already explored the environment with a well-designed deterministic policy. Although the uncertainty in the state model may lead the agent to different possible paths in the future,

the deterministic optimal policy can be close to the stochastic optimal policy. The training procedure for the optimal policy after time instant t_0 mainly incorporates stochasticity to the policy process so that the agent can take appropriate actions corresponding to the random state due to the uncertainty generated by the Brownian motion W in the state dynamics. In other words, we can get better and more complete estimate for the environment parameter as the time step increases. This makes the RL problem that we consider in this work become more and more like a classic stochastic optimal control problem except that we allow the agent to test the BAL designed policy as trials.

3.4. Summary of the algorithm

To summarize our BAL algorithm for solving the RL problem, we provide a pseudo algorithm in Table 1.

4. Numerical experiments

In this section, we use three numerical examples to demonstrate the performance of our BAL method for solving the RL problem.

4.1. Example1: Classic linear-quadratic control with a hidden environment parameter.

In the rst example, we solve a classic linear-quadratic stochastic optimal control problem, in which the state model contains a hidden parameter that represents the unknown in the environment. The main purpose of demonstrating a linear-quadratic control example is that the optimal control can be explicitly derived, hence we can use this example to present the accuracy of our BAL method by comparing with the analytically derived solution. On the other hand, the unknown parameter in the state model requires an exploration procedure to determine the environment, which also makes the control problem in this example an RL problem.

To proceed, we consider an agent, whose state is formulated by the following 2-dimensional linear stochastic dynamical system:

$$dX_t^{a;} = (A(t)X_t^{a;} + Ba_t)dt + dW_t; \tag{30}$$

where $X_t^{a;} \ 2 \ R^2$ is the state of the agent; $A(t) = [\sin t; 0; 0; \cos t]$ is the drift coecient for state X, which contains an unknown parameter , and we choose $= 2$ in our numerical experiments in this example; $B = (0:5; 0:5)^>$ is a 2-dimensional constant vector as the coecient of the scalar policy term a_t; $= [0:1; 0; 0; 0:1]$ is the diusion coecient for the stochastic integral driven by the 2-dimensional Brownian motion W. The cost functional, which is equivalent to the penalty in the RL problem, is in

Table 1: Summary of the algorithm

Algorithm: Backward action learning method for reinforcement learning

Set up the initial state X_0, environment guess particles $\{\theta_0^{(q)}\}_{q=1}^{Q}$ for θ_0, initial guess a^0 for the policy, and choose the user dened constants: K, Q, M, $\{\kappa_k\}_{k=1}^{K}$, and ε.

for $n = 0, 1, 2, \ldots, N_T - 1$

Let $X_{t_n}^{a^0} = x \in X_{t_n}$ be the initial state at time t_n, and obtain the optimal policy $a_{t_n}^{K}(x)$ for each $x \in X_{t_n}$ as follows:

while $k = 0, 1, 2, \ldots, K$, do

Implement policy $a_{t_n:N_T-1}^{k}$ and obtain the agent trial state trajectory $X_{t_n:N_T}^{a^k;}$.

Implement the direct lter method to explore:

Let $\{\theta_{k+1;t_n}^{(q)}\}_{q=1}^{Q} = \{\theta_k^{(q)}\}_{q=1}^{Q}$ be the particles for the estimated environment parameter for $l = n, n+1, \ldots, N_T - 1$

- Generate a set of Q predicted parameter particles $\{\tilde{\theta}_{k+1;t_{l+1}}^{(q)}\}_{q=1}^{Q}$ through the pseudo parameter dynamics Eq. (22);
- Generate Q versions of the state samples $\{\tilde{X}_{l+1}^{a^k;\tilde{\theta}_{k+1;t_{l+1}}^{(q)}}\}_{q=1}^{Q}$ from the approximation scheme Eq. (23) for the next state stage;
- Calculate likelihood values for $\{\tilde{\theta}_{k+1;t_{l+1}}^{(q)}\}_{q=1}^{Q}$ as particle weights $\{\omega_{k+1;t_{l+1}}^{(q)}\}_{q=1}^{Q}$ by comparing $\{\tilde{X}_{l+1}^{a^k;\tilde{\theta}_{k+1;t_{l+1}}^{(q)}}\}_{q=1}^{Q}$ with the real agent state $X_{t_{l+1}}^{a^k;}$ through Eq. (24);
- Resample the particle-weight pairs $\{\tilde{\theta}_{k+1;t_{l+1}}^{(q)}, \omega_{k+1;t_{l+1}}^{(q)}\}_{q=1}^{Q}$ to generate a set of equally weighted particles $\{\theta_{k+1;t_{l+1}}^{(q)}\}_{q=1}^{Q}$.

end for

Set $\{\theta_{k+1}^{(q)}\}_{q=1}^{Q} = \{\theta_{k+1;t_{N_T}}^{(q)}\}_{q=1}^{Q}$ to initialize the next training episode, and let $\kappa_{k+1} = \frac{1}{Q}\sum_{q=1}^{Q}\theta_{k+1}^{(q)}$ be the estimated environment parameter in the training episode $k+1$.

Implement the BAL method to exploit:

- Solve the adjoint BSDE through the numerical BSDE scheme Eq. (25) by using the agent state trajectory $X_{t_n:N_T}^{a^k;}$ and the estimated environment parameter κ_{k+1};
- Calculate the sample-wise approximator ∇J_a for the gradient introduced in Eq. (26);
- Carry out stochastic gradient descent scheme (27), to get the improved policy a^{k+1}.

Use ε-greedy method (28) to enhance exploration and obtain a^{k+1}.

end while

Generate the next state region $D_{t_{n+1}}$ from the state points X_{t_n} and the policy table $\{a^K(x)\}_{x \in X_{t_n}}$ at time t_n through scheme (29). Discretize $D_{t_{n+1}}$ and create state points $X_{t_{n+1}}$.

end for

the following quadratic form

$$J(a) = E\left[\frac{1}{2}\int_0^T \langle QX_t^{a,\alpha}, X_t^{a,\alpha}\rangle + \langle Ra_t, a_t\rangle dt + \frac{1}{2}\langle FX_T^{a,\alpha}, X_T^{a,\alpha}\rangle\right],$$

where $Q = I_2$ and $F = I_2$ are given constant matrices, and we let $R = 1$. In the numerical experiments, we introduce a temporal partition over $[0; 1]$ with the uniform step-size $\Delta t = 0.01$, i.e. $N_T = 100$.

We first show the performance of our direct filter parameter estimation method for the purpose of exploration in the RL task, and we use $Q = 100$ particles to describe the empirical distribution for the unknown parameter α. The initial state of the agent is chosen as $X_0^{a,\alpha} = (6, 2)^\top$, and we assume that the initial guess for the environment parameter is -2. In Figure 1, we present the estimated parameter

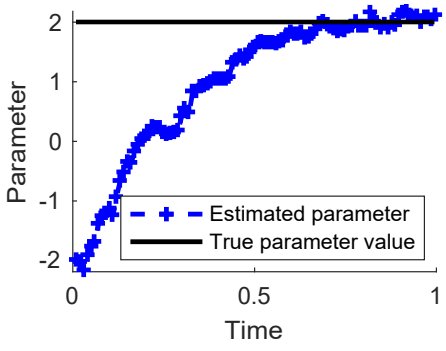


Figure 1: Example 1. Parameter estimation with respect to time in the first episode.

with respect to time in the first training episode, where the solid black line shows the true environment parameter value $\alpha = 2$ while the blue dashed curve marked by plus signs gives our estimated parameter values corresponding to time in the training episode $k = 1$. We can see from this figure that the direct filter method can quickly capture the true parameter even in the first training episode in this 1-dimensional state-independent parameter estimation case. In Example 2 and Example 3, we will consider more complicated situations with state-dependent environment parameters, which can be challenging for standard reinforcement learning techniques.

With the accurately estimated environment, we present the performance of policy estimation with respect to the number of training episodes in Figure 2. In subplots (a), (b), (c), and (d) we compare the BAL method estimated optimal policy (the black dashed curves marked by circles) with the analytical optimal policy (the black dashed curves), which is given by

$$a_t = -R^{-1}B^\top P(t)X_t^{a,\alpha}; \tag{31}$$

where X_t is the agent state, and $P(t)$ is the unique solution of the following so-called Riccati equation

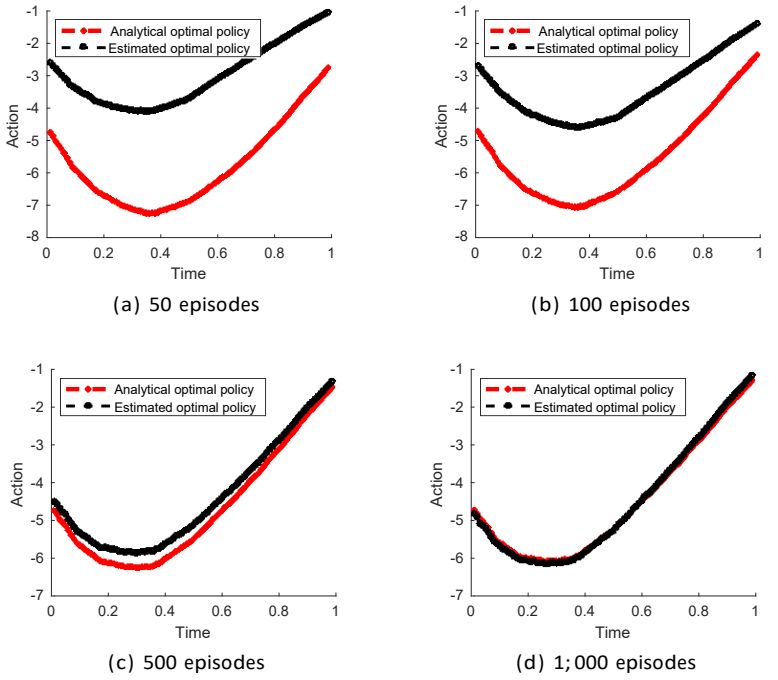


Figure 2: Example 1. Comparison between the estimated optimal policy (actions) and the analytical optimal control.

corresponding to the state equation (30)

$$\frac{dP(t)}{dt} = \quad P(t)A(t) \quad A^{T}(t)P(t) + P(t)BR^{-1}B^{T}P(t) \quad Q; \quad P(T) = F:$$

From this gure, we can see that as we increase the number of training episodes, i.e. $K = 50$, $K = 100$, $K = 500$, and $K = 1;000$ in (a), (b), (c), and (d), respectively, we obtain better and better policy estimation results. When carrying out $K = 1;000$ training episodes, the estimated optimal policy is almost perfectly aligned with the analytical optimal policy. The convergence analysis for the BAL method in the linear case has been discussed in [3]. However, for nonlinear problems, it's hard to guarrantee the global convergence to the optimal control (policy). We also refer to [3] for the general convergence study of the BAL method.

Finally, in Figure 3 we show the accuracy of our policy estimation with dierent initial states $X_0^{a;} = (4;5)^{>}$, $X_0^{a;} = (4;1)^{>}$, and $X_0^{a;} = (6; \ 2)^{>}$. We can see that the BAL method constantly provides accurate policy estimation results.

4.2. Example 2: Reinforcement learning for atomic level manufacture.

In the second example, we solve a mathematically modied material science problem that motivated us to develop such an SMP approach to solve the RL problem with parameterized environment. We want to use this problem as an example to show that there are application problems which require physics

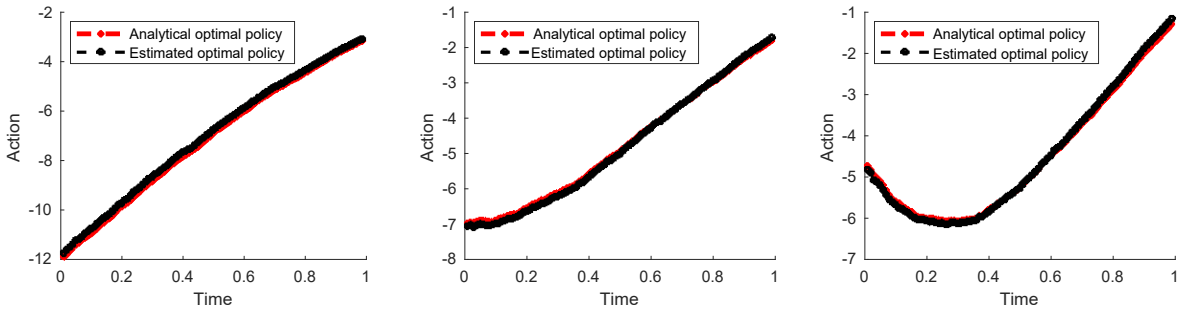


Figure 3: Example 1. Policy accuracy with dierent initial state.

knowledge to be incorporated into the RL model, and there are needs to parameterize the environment in scientic machine learning practices.

The scientic background of the RL problem that we consider in this example is known as the \atomic forge", which is a technique to control and design materials at the nano scale [14]. A new nano-phase fabrication approach has been developed to utilize a scanning transmission electron microscope (STEM) to assemble and manipulate matter atom-by-atom [9, 18]. Although many practical challenges still need to be addressed to achieve the atomic forge technique from the physics aspect, in this work we focus on the mathematical problem about how to automatically control atoms and formulate an RL method to design an eective policy to move a target atom to a pre-designated location on a 2-dimensional material surface, where two-atom potential models can be applied [3].

The motion of a target atom, which is the agent in the RL problem, is mainly driven by atomic forces derived from intermolecular potentials. One of the most important intermolecular potentials is the Lennard-Jones (LJ) potential, which models soft repulsive and attractive interactions between two atoms. In this work, we consider the following AB form of the LJ potential

$$V_{LJ}(r) := \frac{A}{r^{12}} - \frac{B}{r^{6}};$$

where r is the distance between two interacting atoms, ϵ is the depth of the potential-well (usually referred to as \dispersion energy"), and A, B are constant values referred to as \size of the atom". In this work, we let $A = B = 0.5$ be pre-chosen constants, and the depth of the potential-well ϵ is an unknown value to be determined. Since we try to move a target atom on a material surface, the value of ϵ may vary depending on the type of the xed background atom, which is interacting with the moving

[3] Moving atoms in the 3-dimensional space would be similar from the mathematical aspect. However, it could be more challenging in physics.

target atom in the environment.

The atomic force between atoms is in the form of the gradient of the potential $-\nabla V_{LJ}$. In Figure 4, we present a demonstration for the field of the atomic force generated by the LJ potential V_{LJ} corresponding to two types of altogether 9 background atoms with depth parameters $_{deep} = 30$ and $_{shallow} = 1$. We

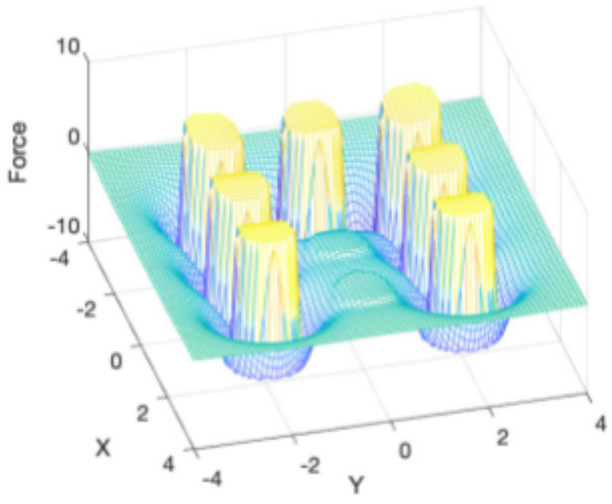


Figure 4: Example 2. The atomic force on the material surface.

can see from the figure that corresponding to the deep potential parameter, i.e. $_{deep} = 30$, both the repulsive and attractive forces are large, while on the other hand the intermolecular force is generally much small near the atoms with the shallow potential parameter, i.e. $_{shallow} = 1$. Also, the moving target atom cannot get too close to the fixed background atom due to the exponentially increased repulsive force, and the target atom can be trapped by one of the background atoms with deep potential well due to the large attractive force applied to the target atom.

With the assumed intermolecular potential V_{LJ} and a background atomic structure as the environment, the trajectory of the target atom can be formulated as

$$dX_t^{a;} = (\ \nabla V_{LJ}(r) + a_t)dt + {}_t dW_t; \tag{32}$$

where ∇V_{LJ} is the gradient of the LJ potential V_{LJ}, which is determined by the depth parameter of the potential-well, and $r = kX_t^{a;}\ Atom_{background}k_2$ is the distance between the target atom and it's closest background atom, i.e. $Atom_{background}$. The policy a_t is the control actions that we apply to drive the target atom and guide it to the pre-designated location. More specically, we let $a_t := (f_t \cos {}_t; f_t \sin {}_t)^{>}$, where f_t is the amount of external force that we apply to counter-eect the background atomic force caused by the LJ potentials, and $_t$ is the steering angle that determines the

direction of the external force.

To demonstrate how an RL algorithm can be applied to the atomic forge technique, we design a background atomic structure in Figure 5, where the blue dots show the locations of the background atoms that can generate shallow potential wells, the yellow dots show the locations of the background atoms that can generate deep potential wells, and the color bar on the right hand side shows the mapping from color to LJ potential values. In this example, we consider the moving target atom as the agent in

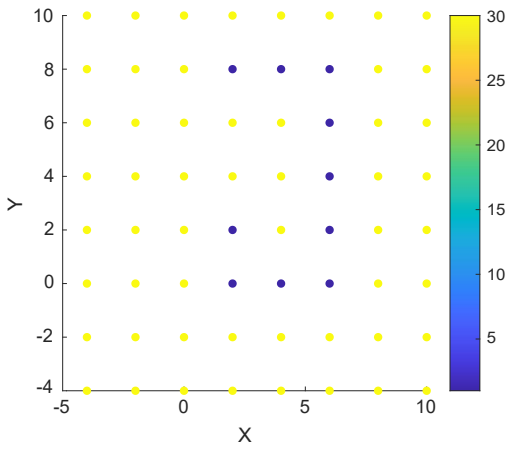


Figure 5: Example 2. The true potential parameters on the material surface.

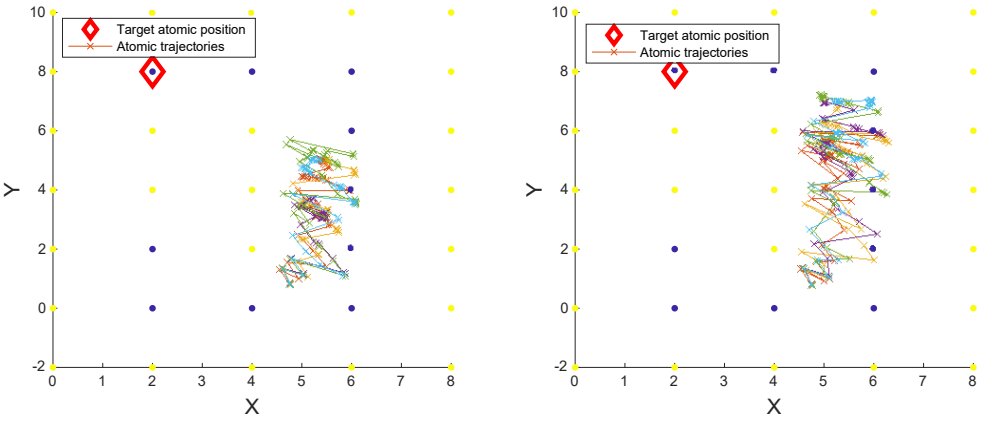
the RL problem, and we choose a pre-designated destination $X_{destination} = (2; 8)^{>}$ to be arrived at the terminal time $T = 10$. The initial state of the agent is chosen as $X_0 = (4:75; 0:75)^{>}$, which is near a background atom that generates shallow potential well. The cost for the RL problem is dened by

$$J(a) = E\left[\int_0^T |f_t|dt + F\,\|X_T^a - X_{destination}\|_2\right]; \tag{33}$$

where $F = 50$ is the amount of penalty for not being able to arrive at the destination at the terminal time T, and $\int_0^T |f_t|dt$ is the running cost that measures how much eort (or energy) that we make to move the agent. The RL problem is to nd an optimal policy a that minimizes the cost $J(a)$. It's worthy to point out that such an RL problem is quite challenging since once the target atom is \captured" by one of the deep potential-well atoms, it needs very large force to drive it out of the potential-well.

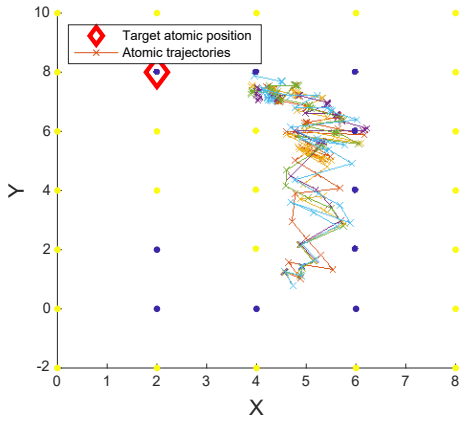
We rst use the standard Q-Learning method with -greed exploration (see [21, 27]) to solve the RL problem described by Eq. (32)-(33). To discretize the original continuous problem, we introduce a temporal partition with step-size $t = 0:2$, i.e. $N_T = 50$, and we introduce a uniform spatial partition with step-size $x = 0:1$ in the state space to generate the q-table. The policy approximation for the q-table is chose as $f = 0:2$ for discretizing the external force f and $= {}_8$ for discretizing the steering direction .

In Figure 6, we show 5 agent performance trajectories following the policy determined by the trained q-table with 10^3, 10^4, 10^5, and 10^6 training episodes in subplots (a), (b), (c), and (d), respectively. We

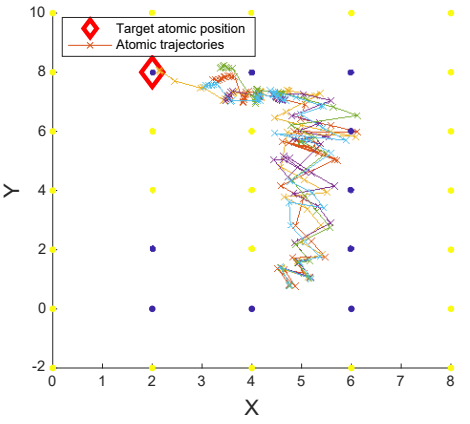


(a) 10^3 training episodes in Q-Learning

(b) 10^4 training episodes in Q-Learning



(c) 10^5 training episodes in Q-Learning



(d) 10^6 training episodes in Q-Learning

Figure 6: Example 2. Performance of Q-Learning.

can see from this Q-Learning performance experiment that with 10^3 training episodes in the Q-Learning method, the agent could avoid the shallow potential \trap atoms" on the left side, which looks more promising at beginning but has some deep potential-well atoms as the \barrier" towards the destination atom (marked by the red diamond). However, with only 10^3 episodes, the Q-Learning method cannot generate a suciently trained q-table that guides the agent towards the nal destination. We can also observe from Figure 6 that with longer and longer training procedures, the agent performance becomes better and better. With 10^6 training episodes, one of those 5 agents can nally arrive at $X_{destination}$, and the other 4 agents also get close to the destination.

To show the advantageous performance of our algorithm, we also solve the RL problem for the

atomic-forge technique by using our BAL method. To compare with the Q-Learning method, we use the same temporal partition, and we introduce the same uniform spatial partition with step-size x = 0:1 to approximate the state region D_{t_n} at each time t_n. Therefore, we approximate the continuous RL problem Eq. (32)-(33) with the same level of discretization accuracy as the Q-Learning method when using the BAL method. To explore the environment, we use Q = 100 particles in the direct lter to estimate the environment parameter, and we carry out K = 10^3 episodes in the training procedure. The initial guess for the potential depth parameter of each atom is chosen as $_{guess}$ = 1. We rst present

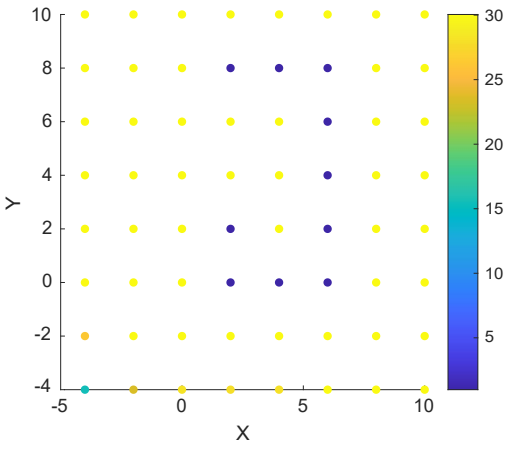


Figure 7: Example 2. The estimated potential parameters on the material surface.

the performance of direct lter based exploration in Figure 7, where we also use the same color bar as we used in Figure 5 to demonstrate the estimated potential depth parameter values corresponding to dierent background atoms. By comparing the estimated environment in Figure 7 with the true atomic environment in Figure 5, we can see that the environment learned in the BAL method is very similar to the true environment except for the left-bottom corner. Note that the agent (i.e. the target atom) tries to move around the atoms with shallow potential-well (blue atoms) to save energy, and the agent does not have much experience near the left-bottom corner, which makes the left-bottom corner insuciently explored.

In Figure 8, we present 5 performance trajectories of the agent guided by the trained policy, which is obtained by the BAL method with 10^3 training episodes. From this gure, we can see that all 5 agents arrived at the designated target location at the terminal time. We also want to mention that due to the exploration mechanism in the BAL approach for RL, the agent has good understanding of the potential-well depths for the atoms near its route towards the destination. On the other hand, since the agent does not need to move around near the atoms at the left-bottom in the graph, it cannot learn the potential-well depth parameters very well for those atoms.

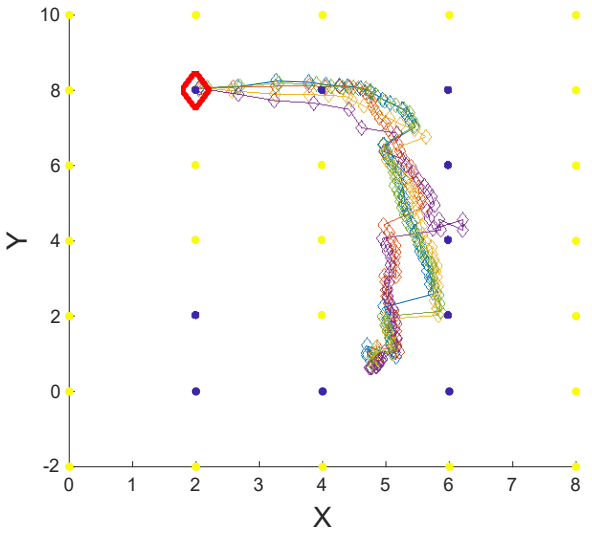


Figure 8: Example 2. Performance of BAL.

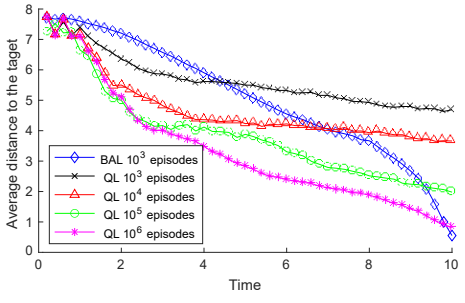


Figure 9: Example 2. Comparison of average distance to the destination.

To further demonstrate the performance comparison between the Q-Learning method and the BAL method, we calculate the average distance to the destination by letting the agent repeatedly run 100 trajectories, and we present the average distance in Figure 9. We can see from this gure that the BAL method with 10^3 training episodes outperforms the Q-Learning method with 10^3, 10^4, and 10^5 training episodes in terms of the average distance to the destination, and it slightly outperforms the Q-Learning method with 10^6 training episodes with a very small margin. The main reason that the BAL would outperform the temporal-dierence type RL with fewer training episodes is that the gradient process (with respect to policy) can give us a direction to improve the policy. Therefore, the policy improvement in the BAL method is more eective.

On the other hand, the criteria for the general performance of the agent is not only the distance to the designated destination. The consumption of energy for moving the agent, i.e. the running cost $\int_0^T |f_t| dt$, should also count for the performance. In Figure 10, we present the average running cost of

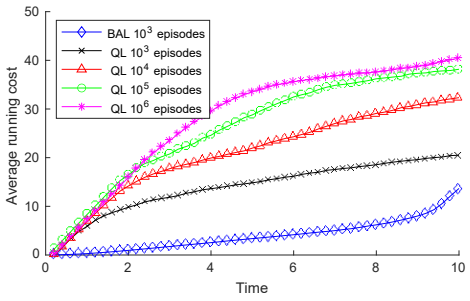


Figure 10: Example 2. Comparison of average running costs.

each implementation with respect to time. We can see from this gure that the BAL requires much lower energy consumption compared with all the Q-Learning implementations { regardless of the distance to the destination.

4.3. Example 3: Reinforcement learning for continuous maneuvering of a robot in a maze.

In this example, we solve an RL problem for continuous maneuvering of a robot in a 2-dimensional maze, which is a continuous version of the maze solver problem [5, 30]. We want to use this example to show the major advantage of the SMP approach over the DPP approach. As an SMP type approach, the BAL method treats the entire control procedure as a whole task, which allows the RL algorithm to design the control policy based on the predicted future trajectories of the agent. On the other hand, the DPP based methods, such like Q-learning, typically balance between the short-term optimal performance and the possible ultimate goal of the control task, and the design of the policy in the Q-Learning method does not rely on the comprehensive understanding of the entire environment. As a result, the Q-Learning designed policy could make the agent stuck in a local dilemma.

To proceed, we consider the following stochastic dynamics that describe the agent state

$$\begin{aligned} dX_t^{(1)} &= v\cos(\theta)dt + dW_t^{(1)}; \\ dX_t^{(2)} &= v\sin(\theta)dt + dW_t^{(2)}; \end{aligned} \tag{34}$$

where $X_t = (X_t^{(1)}; X_t^{(2)})^\top$ is the 2-dimensional location of a robot, the policy term $a = (v; \theta)$ controls the velocity v and the steering action θ. The goal of the RL task in this example is to let the robot, i.e. the agent, learn how to arrive at a pre-designated destination X_T at the given terminal time T with the lowest cost, and the cost function that we aim to minimize during the learning procedure is

$$J(a) = E\Big[\int_0^T {}_x\|X_t - X_0\|_2^2 dt + F\|X_T - X_T\|_2^2\Big]; \tag{35}$$

where $F = 20$ is a terminal cost constant that denes the amount of penalty for not being able to arrive

at the destination X_T at the terminal time T, and $\int_0^T {}_x \| X_t - X_0 \|^2 dt$ is the running cost term with an unknown parameter ${}_x$. Dierent from the RL problem for the atomic forge technique, in which we try to learn environment parameters that determine the state dynamics. In this work, as a RL problem for maze solver, the parameters that represent the unknowns in the environment are in the cost function $J(a)$ dened in Eq. (35). Moreover, the RL problem that we try to solve in this example is dierent from a standard maze solver RL problem, in which the environment contains discrete obstacles (that add constant penalties to the cost if the agent touches them) and barriers (that stop the robot from getting through). In the RL problem Eq. (34) - (35), we introduce a space-dependent cost parameter ${}_x$, which does not stop the agent from getting through, and the cost parameter ${}_x$ will continuously add dierent levels of cost as the agent moving in the environment. Therefore, stepping into a small region with high cost parameter may not bring very high cost to the overall cost function $J(a)$. On the other hand, if the agent moves in a region with a xed cost parameter, the faster the agent moves (or the farther the agent travels in a unit time) the more cost will be generated due to the accumulated running cost with respect to the distance that the agent traveled, i.e. $\| X_t - X_0 \|_2$ in Eq. (35).

To carry out numerical experiments, we introduce a temporal partition over time interval $[0; T]$ with $T = 20$, and we choose the time step-size $t = 0{:}2$, i.e. $N_T = 100$. The diusion coecient in Eq. (34) is chosen as $= 0{:}05$, which can bring relatively large amount of uncertainty to the state process given the length of the time interval and the time step-size. Also, we introduce a spatial partition to the environment by letting the x-dimension step-size $x = 0{:}2$ and y-dimension step-size $y = 0{:}25$, and we introduce a small base running cost ${}^{base} = 0{:}02$ everywhere in the state-space. The initial state of the agent is chosen as $X_0 = (5; 4)^>$, and the destination location is $X_T = (5; 25)^>$.

We rst solve the above maze problem by using the Q-Learning method. Apparently, if the parameter ${}_x$ remains as the small invariant base cost ^{base} over the entire state space, the Q-Learning method would quickly converge and provide a policy that guides the agent to arrive at the destination directly. However, when an unknown high cost obstacle region appears, the maze problem could be more challenging. In the rst Q-Learning experiment, we put a rectangle region $[4{:}8; 5{:}4] \times [7; 11]$ in the environment and let the running cost coecient in the region be ${}_x = 20$. Then, we train the q-table for 5×10^5 episodes and present 30 testing agent trajectories using the trained q-table in Figure 11, where the red diamond shows the location of the destination X_T, and the background mesh reects the spatial partition for the state space. We can see from this gure that all the agents know that they should move up towards the destination. Once they step into the obstacle region, since they are still getting closer to the destination, the cost (or penalty) in the near future may not be large enough to stop them from moving forward. In other words, the high penalty of not being able to reach the destination may persuade the agent to overcome the short-term diculties. However, when the agent accumulates large enough cost as it gets deeper into the obstacle region, the cost would only increase no matter where the agent goes. Therefore,

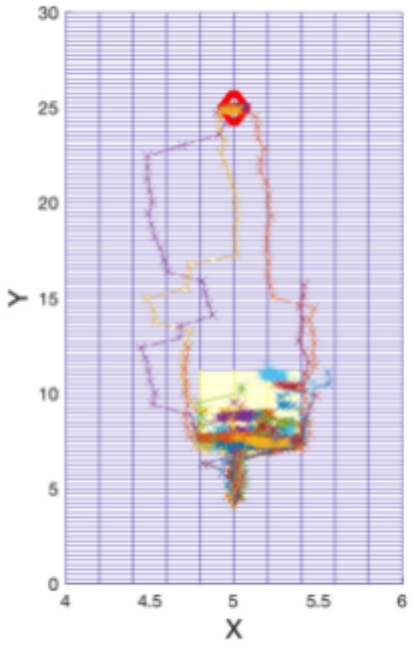
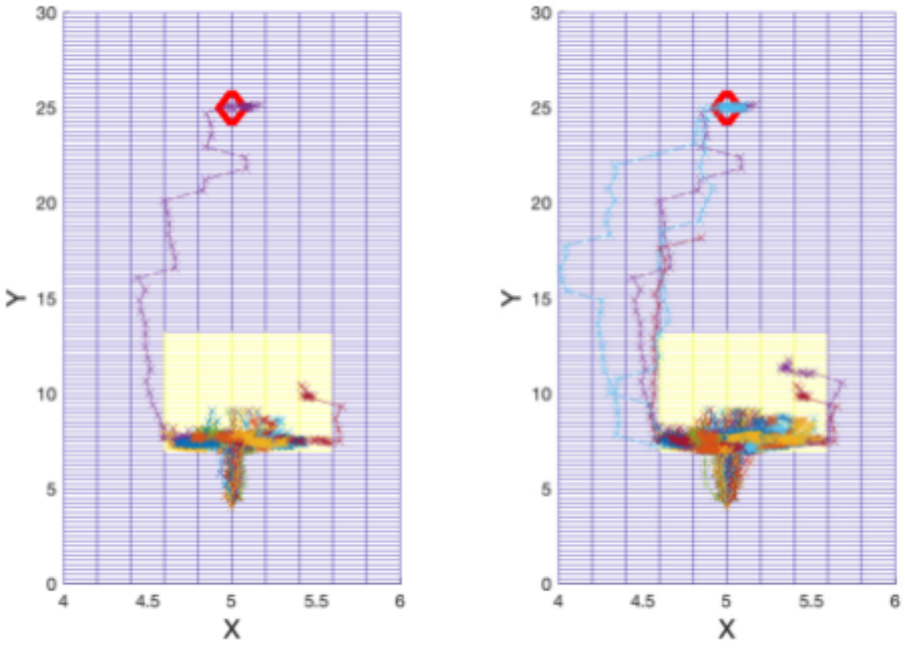


Figure 11: Example 3. Performance of Q-Learning with a smaller obstacle region in the environment.

we can see from Figure 11 that most agents stop somewhere in the obstacle region. At the same time, we can also see from the gure that there are still several policy paths that would guide the agent to avoid the obstacle region, and it's more likely that those agents can arrive at the designated destination.

To explore more along this direction and study a more challenging scenario for the Q-Learning method, we increase the size of the obstacle region to $[4:6; 5:6]$ $[7; 13]$ and still train the q-table for 5 10^5 episodes. The performance of 30 testing agent trajectories is plotted in Figure 12 (a). From this gure, we can see more clearly how the agents moved horizontally and tried to move out of the obstacle region. One agent trajectory on the right hand side actually moved out of the obstacle region. Unfortunately, as this agent turned back towards the destination, it's trapped by the obstacle region again. The only agent that successfully arrived at the destination is plotted by the trajectory on the left. To further demonstrate the performance of the agent in this experiment, we present 100 agent testing trajectories in Figure 12 (b). From this gure, we can see that the \successful" policy followed by the trained q-table is on the left, and only the agents that follow the left-side-policy could arrive at the destination.

In Figure 13, we design a much more complicated environment with 6 obstacle regions, and each region has a dierent cost parameter value. Specically, from the right to the left we let $_x = 5, _x^1 = 20,$ $_x{}^2 = 15, _x \overset{3}{=} 25, _x = {}^4 10, _x = 30.$ The dierent running cost parameter values are presented by the heights of the obstacle regions in Figure 13. In Figure 14, we show the performance



(a) Performance of Q-Learning: 30 agent trajectories

(b) Performance of Q-Learning: 100 agent trajectories

Figure 12: Example 2. Performance of Q-Learning with a larger obstacle region in the environment.

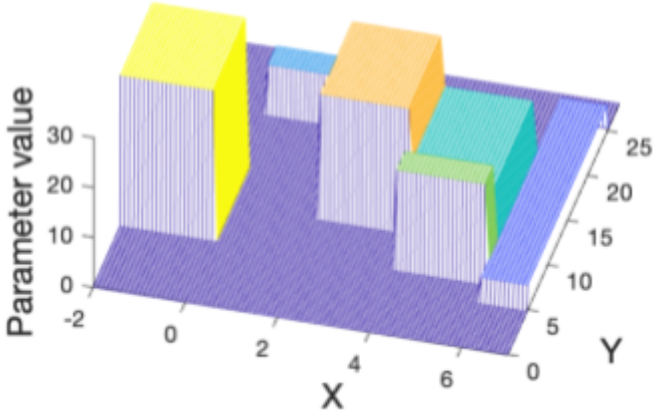


Figure 13: Example 3. The 3D view of the true environment of the maze. The heights of the obstacle regions show dierent cost parameter values.

of the agent with 30 testing trajectories that follow the policy obtained by the Q-Learning method with $5\,10^5$ training episodes. There's no surprise that the agent cannot nd a path to avoid all the obstacles

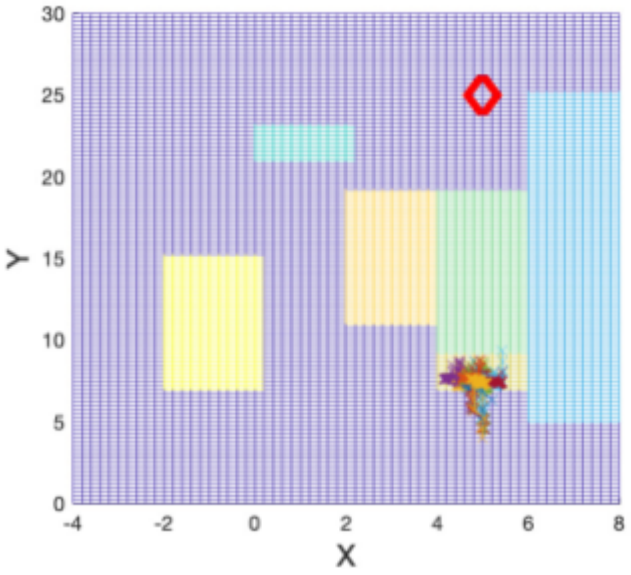


Figure 14: Example 3. Performance of Q-Learning in the maze presented in Figure 13.

and arrive at the destination. Although we can conjecture that by training the agent with more and more episodes, the q-table may eventually be well-trained enough to create some \successful" policies. However, for such a complicated environment with so many trapping obstacle regions, it would be very computationally expensive for Q-Learning to nd a path to arrive at the destination.

In the following experiments, we show the success of the BAL method in solving the RL problem Eq. (34) - (35). To implement the BAL method, we use Q = 100 particles to carry out the direct lter based parameter estimation for exploration, and we carry out K = 1000 training episodes in the BAL algorithm. The temporal partition and the spatial partition that we use for the BAL method are the same as the Q-Learning method. In this experiment, we don't assume that the agent knows there are altogether 6 obstacle regions. Instead, we use the direct lter based exploration technique to estimate the running cost parameter in every articially partitioned spatial block with partition size x y.

In Figure 15, we rst present the estimated environment learned by the BAL method, where the background mesh shows all the articially partitioned spatial blocks that determine the size of environment parameter. We can see by comparing Figure 15 with Figure 13 that the BAL method successfully recovered the environment, and the estimate for every obstacle region is very accurate { in both the size of each obstacle region and the value of each cost parameter. Although we only implemented 1000 training episodes, since the direct lter updates the estimate for the environment at every time instant in each training episode, the parameter estimation algorithm has suciently incorporated the agent trial states into the exploration procedure, and this helps the agent have very good understanding of the

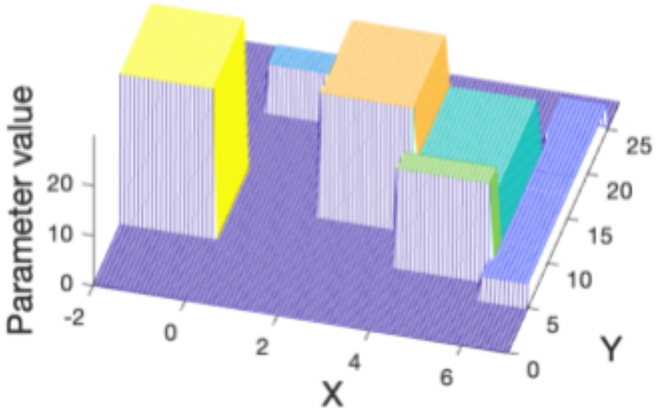


Figure 15: Example 3. The estimated estimated environment of the maze obtained by the BAL method.

environment.

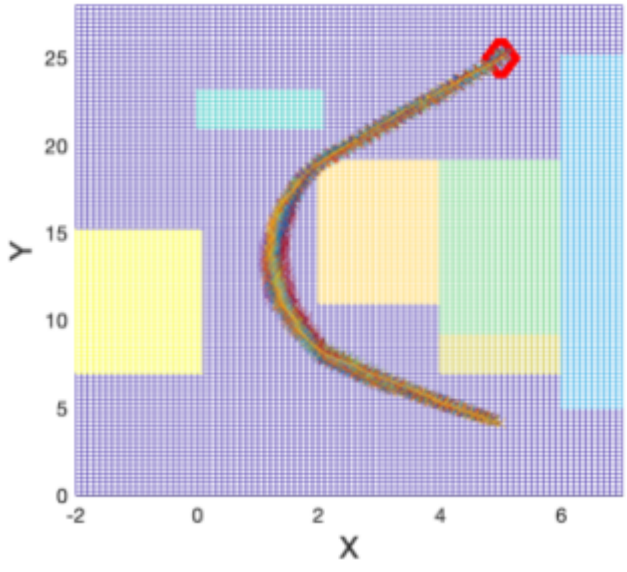


Figure 16: Example 3. Performance of the BAL method in the maze.

In Figure 16, we present 30 agent trajectories guided by the BAL trained policy with initial state $X_0 = (5; 4)^>$, and terminal destination is still chosen as $X_T = (5; 25)^>$. We can see from this gure that all the agent trajectories follow very smooth paths towards the destination, and they all arrived at the destination at the terminal time.

To further demonstrate the performance of the BAL method in this example and show the robustness of our method, we let the agent start from randomly picked initial states in the spatial area $[4;6]\,[3;5]$, and we present 30 agent trajectories following the BAL trained policy in Figure 17. We can see from this

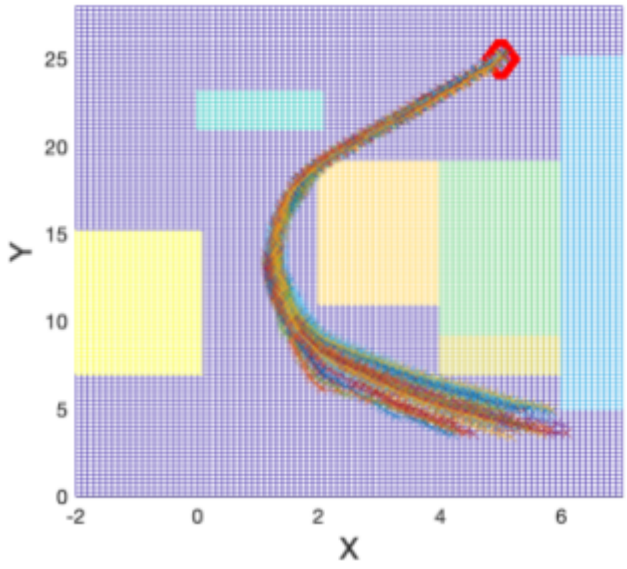


Figure 17: Example 3. Performance of the BAL method in the maze with random initial states.

gure that no matter where the agent started, it can always nd the right path towards the destination, and it can perfectly avoid the obstacle regions.

By comparing the performance of the Q-Learning trained policy with the BAL trained policy from the above experiments, we can see that the BAL method clearly outperforms the Q-Learning method in this continuous maze solver RL problem.

5. Conclusions

In this work, we developed a stochastic maximum principle (SMP) approach for solving the reinforcement learning (RL) problem in the case that the environment can be parameterized. To explore the environment, we introduced a direct lter method as an online parameter estimation method to learn the environment parameters during the training procedure, and the exploitation task is carried out through an ecient backward action learning (BAL) algorithm for nding the optimal policy under the SMP framework. The main advantage of such an SMP approach, compared with dynamic programming principle (DPP) based methods, is that the gradient of the cost with respect to the control aims to nd the improvement direction for the control over the entire performance period, which could potentially provide better overall performance for the long-run. In contrast, the standard temporal dierence (TD)

learning methods under the DPP framework, such like Q-Learning, only consider short-term rewards or penalties, which could ignore possible future opportunities. Especially, like the numerical experiments presented in Example 4.3, if the environment is carefully designed so that the agent is deeply trapped in a situation where no short-term strategy can lead it out, any TD learning method will struggle unless it can consider long-term predictions. However, TD learning with long-term predictions can be very dicult when the environment is unknown. Although our BAL method also needs to consider long-term predictions, the direct lter method can be eectively combined with the BAL algorithm to establish a comprehensive understanding of the environment, and this allows us to dynamically learn the en-tire environment while searching for the long-term optimal policy. The major drawback of our BAL method for solving the RL problem is that we require a parameterization for the environment. But for applications of RL in science, such like the atomic forge technique, physics knowledge and pre-dened models are necessary to make the RL problem \physics-informed". In this case, having a parameterized environment is a reasonable assumption.

Although the BAL outperforms the DPP based RL in reliability, we also acknowledge that with help of deep learning, the so-called deep RL is potentially capable to exploit the expressive power of neural networks to handle very large-scale problems. In the future, we plan to adopt deep neural network techniques in our BAL framework and develop deep learning based BAL to solve high-dimensional problems.

Acknowledgement

This work is partially supported by U.S. Department of Energy through FASTMath Institute and Oce of Science, Advanced Scientic Computing Research program under the grant DE-SC0022297. The second author (FB) would also like to acknowledge the support from U.S. National Science Foundation through project DMS-2142672. The third author (JY) would like to acknowledge the support from U.S. National Science Foundation through project DMS-2305475.

References

[1] C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. J. R. Statist. Soc. B, 72(3):269{342, 2010.

[2] R. Archibald, F. Bao, and X. Tu. A direct lter method for parameter estimation. J. Comput. Phys., 398:108871, 17, 2019.

[3] R. Archibald, F. Bao, Y. Cao, and H. Sun. Convergence Analysis for Training Stochastic Neural Networks via Stochastic Gradient Descent. arXiv preprint arXiv:2212.08924, 2022.

[4] R. Archibald, F. Bao, J. Yong, and T. Zhou. An ecient numerical algorithm for solving data driven feedback control problems. Journal of Scientic Computing, 85(51), 2020.

[5] Bram Bakker. Reinforcement learning with long short-term memory. Advances in neural information processing systems, 14, 2001.

[6] F. Bao, N. Cogan, A. Dobreva, and R. Paus. Data assimilation of synthetic data as a novel strategy for predicting disease progression in alopecia areata. Mathematical Medicine and Biology: A Journal of the IMA, 2021.

[7] Feng Bao, Yanzhao Cao, Amnon Meir, and Weidong Zhao. A rst order scheme for backward doubly stochastic dierential equations. SIAM/ASA J. Uncertain. Quantif., 4(1):413{445, 2016.

[8] D. Crisan and A. Doucet. A survey of convergence results on particle ltering methods for practitioners. IEEE Trans. Sig. Proc., 50(3):736{746, 2002.

[9] O. Dyck, M. Ziatdinov, S. Jesse, F. Bao, A. Yousefzadi Nobakht, A. Maksov, B.G. Sumpter, R. Archibald, K.J.H. Law, and S.V. Kalinin. Probing potential energy landscapes via electron-beam-induced single atom dynamics. Acta Materialia, 203:116508, 2021.

[10] Pierre Yves Glorennec and Lionel Joue. Fuzzy q-learning. In Proceedings of 6th international fuzzy systems conference, volume 2, pages 659{662. IEEE, 1997.

[11] Bo Gong, Wenbin Liu, Tao Tang, Weidong Zhao, and Tao Zhou. An ecient gradient projection method for stochastic optimal control problems. SIAM J. Numer. Anal., 55(6):2982{3005, 2017.

[12] N.J Gordon, D.J Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. IEE PROCEEDING-F, 140(2):107{113, 1993.

[13] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In International conference on machine learning, pages 2829{2838. PMLR, 2016.

[14] S. Kalinin, A. Borisevich, and S. Jesse. Fire up the atom forge. Nature, 22 November 2016.

[15] P. E. Kloeden and E. Platen. Numerical solution of stochastic dierential equations, volume 23 of Applications of Mathematics (New York). Springer-Verlag, Berlin, 1992.

[16] Viraj Mehta, Biswajit Paria, Je Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning, 2021.

[17] M. Morzfeld, X. Tu, E. Atkins, and A. J. Chorin. A random map implementation of implicit lters. J. Comput. Phys., 231(4):2049{2066, 2012.

[18] Ali Yousefzadi Nobakht, Ondrej Dyck, David B Lingerfelt, Feng Bao, Maxim Ziatdinov, Artem Maksov, Bobby G Sumpter, Richard Archibald, and Sergei V Kalinin Stephen Jesse, and Kody JH Law. Reconstruction of eective potential from statistical analysis of dynamic trajectories. AIP Advances, 10:065034, 2020.

[19] Jing Peng and Ronald J Williams. Incremental multi-step q-learning. In Machine Learning Proceedings 1994, pages 226{232. Elsevier, 1994.

[20] Shi Ge Peng. A general stochastic maximum principle for optimal control problems. SIAM J. Control Optim., 28(4):966{979, 1990.

[21] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction second edition: 2014, 2015. 2014.

[22] Gerald Tesauro et al. Temporal dierence learning and td-gammon. Communications of the ACM, 38(3):58{68, 1995.

[23] Michel Tokic and Gunther Palm. Value-dierence based exploration: adaptive control between epsilon-greedy and softmax. In Annual conference on articial intelligence, pages 335{346. Springer, 2011.

[24] Neythen J. Treloar, Nathan Brani, Brian Ingalls, and Chris P. Barnes. Deep reinforcement learning for optimal experimental design in biology. bioRxiv, 2022.

[25] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on articial intelligence, volume 30, 2016.

[26] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. Journal of Machine Learning Research, 21(198):1{34, 2020.

[27] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3):279{292, 1992.

[28] Jiongmin Yong and Xun Yu Zhou. Stochastic controls, volume 43 of Applications of Mathematics (New York). Springer-Verlag, New York, 1999. Hamiltonian systems and HJB equations.

[29] Jianfeng Zhang. A numerical scheme for BSDEs. Ann. Appl. Probab., 14(1):459{488, 2004.

[30] Xiaoping Zhang, Yihao Liu, Dunli Hu, and Lei Liu. A maze robot autonomous navigation method based on curiosity and reinforcement learning. In The 7th Int. Workshop on Advanced Computational Intelligence and Intelligent Informatics (IWACIII 2021), Article, number M1-6, page 1, 2021.

[31] Weidong Zhao, Yu Fu, and Tao Zhou. New kinds of high-order multistep schemes for coupled forward backward stochastic dierential equations. SIAM J. Sci. Comput., 36(4):A1731{A1751, 2014.