SIAM J. NUMER. ANAL.
Vol. 61, No. 2, pp. 973–994

VALUE-GRADIENT BASED FORMULATION OF OPTIMAL CONTROL PROBLEM AND MACHINE LEARNING ALGORITHM*

ALAIN BENSOUSSAN†‡, JIAYUE HAN‡,
SHEUNG CHI PHILLIP YAM§, AND XIANG ZHOU‡¶

Abstract. Optimal control problem is typically solved by first finding the value function through the Hamilton–Jacobi equation (HJE) and then taking the minimizer of the Hamiltonian to obtain the control. In this work, instead of focusing on the value function, we propose a new formulation for the gradient of the value function (value-gradient) as a decoupled system of partial differential equations in the context of a continuous-time deterministic discounted optimal control problem. We develop an efficient iterative scheme for this system of equations in parallel by utilizing the fact that they share the same characteristic curves as the HJE for the value function. For the theoretical part, we prove that this iterative scheme converges linearly in L^2_α sense for some suitable exponent α in a weight function. For the numerical method, we combine a characteristic line method with machine learning techniques. Specifically, we generate multiple characteristic curves at each policy iteration from an ensemble of initial states and compute both the value function and its gradient simultaneously on each curve as the labeled data. Then supervised machine learning is applied to minimize the weighted squared loss for both the value function and its gradients. Experimental results demonstrate that this new method not only significantly increases the accuracy but also improves the efficiency and robustness of the numerical estimates, particularly with less characteristics data or fewer training steps.

Key words. optimal control, value function, Hamilton–Jacobi equation, machine learning, characteristic curve

MSC codes. 65K05, 93-08

DOI. 10.1137/21M1442838

1. Introduction.

It is well known that the study of the Hamilton–Jacobi equation (HJE) is one of the core topics in optimal control theory for controlling continuous-time differential dynamical systems via the principle of dynamical programming [26, 12, 25, 10]. This equation is a first-order nonlinear partial differential equation (PDE) for the value function which maps an arbitrary given initial state to

* Received by the editors August 30, 2021; accepted for publication (in revised form) October 13, 2022; published electronically April 26, 2023.
https://doi.org/10.1137/21M1442838

Funding: The work of the first author was supported by National Science Foundation grant DMS-1905449 ,grant HKSAR-GRF grant 14301321 and grant NSF-DMS 2204795. The work of the second author was supported by the HKUGC for Ph.D. candidates; part of the current work contributes to the partial fulfillment of her Ph.D. dissertation. The work of the third author was partially supported by HKGRF grant 14300319 with the project title "Shape-constrained Inference: Testing for Monotonicity" and HKGRF grant 14301321 with the project title "General Theory for Infinite Dimensional Stochastic Control: Mean Field and Some Classical Problems" awarded by Hong Kong RGC. The work of the fourth author was supported by Hong Kong RGC GRF grants 11307319, 11308121, and 11318522.

† Naveen Jindal School of Management, University of Texas at Dallas, Richardson, TX 75080-3021 USA (axb046100@utdallas.edu).

‡ School of Data Science, City University of Hong Kong, Kowloon, Hong Kong SAR (jyhan5-c@my.cityu.edu.hk, xizhou@cityu.edu.hk, axb046100@utdallas.edu).

§ Department of Statistics, Chinese University of Hong Kong, Shatin, NT, Hong Kong SAR (scpyam@sta.cuhk.edu.hk).

¶ Department of Mathematics, City University of Hong Kong, Kowloon, Hong Kong SAR (xizhou@cityu.edu.hk).

the optimal value of the cost function. Once this HJE solution is known, it can be used to construct the optimal control by taking the minimizer of the Hamiltonian. Such an optimal control is the feedback control, and it does not depend on knowledge of initial conditions.

Although theoretically welldeveloped, numerical methods for the problem are yet to be studied because only a few optimal control problems, such as the linear quadratic problem (LQ) [10], have analytical solutions. Solving the PDE given by the HJE is not easy, even for the LQ case, in which the HJE is converted into a Riccati equation. Moreover, since the dimension of the HJE is the dimension d of state variable x in the dynamical system, the size of the state-discretized problems in solving HJE increases exponentially with d. This "curse of dimensionality" has been the long-standing challenge in solving high-dimensional HJE, but recently there have appeared rapid and abundant developments for mitigating this challenge by combining optimal control algorithms with machine learning algorithms, particularly reinforcement learning and deep neural networks [50, 13, 49, 11].

In the literature, there exists extensive research on various numerical methods for finding the approximate solution to the HJEs. One important idea attracting a considerable amount of attention is termed the successive approximation method [4, 5, 6], which aims to handle the nonlinearity in the HJE. The successive approximation method reduces the nonlinear HJE to an iterative sequence of linear PDEs called the generalized Hamilton–Jacobi equation (GHJE) and the pointwise optimization of taking the minimizer of the Hamiltonian. The GHJE is linear since the feedback control is given from the previous iteration. Therefore traditional numerical PDE methods, such as the Galerkin spectral method (successive Galerkin approximation [4]) for small d can be applied to solve these GHJEs. If the dimension is moderately large, various methods based on low-dimensional *ansatz,* such as polynomial or low-rank tensor product [28, 33, 45], usually work in many applications. For very high dimensional settings, the use of deep neural networks is prevalent. This two-step procedure in the successive approximation follows exactly the same idea as *policy iteration* in reinforcement learning [50, 13].

When the Hamiltonian minimization has a closed form, the HJE can be solved directly by using grids and finite difference discretization, e.g., Dijkstra-type methods such as level set [44], fast marching [53], fast sweeping [52], and semi-Lagrangian approximation [24]. But these grid-based methods suffer from the curse of dimensionality, i.e., they generally scale up exponentially with increases in dimension in the space. There have been tremendous advances in numerical methods and empirical tests for high- dimensional PDEs, taking advantage of neural networks to represent high-dimensional functions. For the HJE in deterministic optimal control problems, various approaches have been proposed, and most are based on certain forms of Lagrangian formulation equivalent to the HJE. For example, under certain conditions (such as convexity) on the Hamiltonian or the terminal cost, the inspiring work of [16, 17, 18, 22, 41] relies on the generalized Lax and Hopf formulas to transform the computation of the value function at an arbitrarily given space-time point into an optimization problem for the terminal value of the Lagrangian multiplier p[1], subject to the characteristics equation of a Hamiltonian ordinary differential equation (ODE) for (x, p). In a similar style, [34] worked with the Pontryagin maximum principle (PMP) by considering the characteristic equations of the state $x(t)$ and the costate $\lambda(t)$ as a two-point boundary value problem (BVP). The optimal feedback control, the value

[1] also called costate or adjoint variable.

function, and the gradient of the value function on the optimal trajectories are computed first by solving the BVP numerically. With the data generated from the BVP on characteristic trajectories, the HJE solution is then interpolated at any point by either using sparse grid interpolants [35] or minimizing the mean square errors [32, 42, 43]. This step is the standard form of supervised learning, and the numerical accuracy is determined by the quality of the interpolant and the amount of training data. For a very large d, the curse of dimensionality is mitigated by the supreme power of deep neural networks in deep learning. For a review on solving high-dimensional PDEs, including the HJE, we refer the reader to the recent review paper [23].

We also state both the connection and the difference between our method and other deep learning based methods for optimal control problems. The authors of [20] provide a network architecture for time-related HJE; it digs deep into network design and numerical experiments. References [2] and [31] solve stochastic optimal control problems through various reinforcement learning related methods, for example, NNContPI, ClassifPI, and Hybrid-LaterQ. Studies [21, 15, 14, 19] can be considered as a series of works that apply the Lax–Oleinik formula for optimal control problems. Compared with them, our work is designed more from a control perspective and follows a totally different path: we directly obtain PDEs for the value function and value-gradient from the HJE and then solve them through an iterative method.

Meanwhile, since the introduction of deep Q-learning [30] in 2016, Q-iteration has been used to solve optimal control problems; see [37, 39, 36, 46]. These works approximate Q-function $Q(x, a^*(x))$ instead of the value function $\Phi(x)$. The value function can be considered as the Q-function under the optimal action, i.e., $\Phi(x) = Q(x, a^*(x))$, where $a^*(x)$ is the optimal control. Since the optimal control should be obtained together with the value function, our method could be more direct than using the Q-function. Among the works mentioned above, [37] also applies the gradient of the Q-function; however, the "gradient" in that work is not the same as ours; that gradient refers to the gradient of Q with respect to control a, which is mainly used for updating the control a. There is no PDE for ∇Q_a or ∇Q_x in [37] which can help calculate the Q-function more efficiently; the presence of a supremum taking in their formulation makes the derivation of PDEs from this equation difficult. However, in our paper, we have a decoupled PDE system (see (3.8) in our paper) for the gradient of the value function. This PDE system enables us to obtain the value function and the value-gradient simultaneously, which can ensure the robustness of the numerical method, as we have already demonstrated in the experiments.

In the present paper, we shall develop a new formulation as an alternative to the HJE for the optimal control theory. This formulation focuses on the gradient of the value function instead of the value function itself. For brevity, we call this vector-valued gradient function a *value-gradient* function. One of our motivations is the fact that in practical applications, the optimal feedback control, or the optimal policy, is the ultimate goal of the decision maker, and this optimal policy is completely determined by the value-gradient in minimizing the Hamiltonian. Another motivation for investigating this value-gradient function comes from the training step, where we want to provide the data not only for the value function but also for its gradient to enhance the accuracy of the interpolation. Our new formulation has the following nice properties: (1) The proposed method has a linear convergence rate. (2) It is a closed system of PDEs for components of vector-valued value-gradient functions. (3) The system is essentially decoupled in each component and is perfectly suitable for parallel computing in policy iteration. (4) Each PDE in the system has exactly the same characteristics equation as the original HJE for the value function. (5) After

simulating characteristics curves, we obtain the results of the value function and the value-gradient function simultaneously on the characteristics curves to train the value function in the whole space.

We demonstrate our novel method by focusing on the infinite-horizon discounted deterministic optimal control problem. This setup will simplify our presentation since the HJE is stationary in time. In addition, we assume the value functions of concern are sufficiently smooth, at least C^2, which can be guaranteed by imposing appropriate conditions on the state dynamics and the running cost functions. So, we can interpret the system of PDEs for value-gradient functions in the classical sense.

We develop the numerical algorithm based on the *policy iteration* [50] and the method of characteristics [40]. Under an assumption on the dynamics and the payoff function, we show by mathematical induction that the value-gradient function at each iteration and its corresponding control are uniformly bounded by linear growth functions, while the gradients of these two functions are uniformly bounded by constants. With Lemmas 3.1 and 3.2, this algorithm is proved to converge linearly in L^2_α sense (see Theorem 3.5) for some suitable exponent α in a weight function. As for the algorithm, in each policy iteration, only linear equations are solved on the characteristics curves starting from a collection of initial states. The interpolation, or the training step, is to minimize the convex combination of the mean squared errors of both the value function and the value-gradient function. One prominent benefit of our algorithm is that we can combine the data from both the value and value-gradient since they share the same characteristics. So, the output of our algorithm is still the value function, which is approximated by any type of nonparametric functions, such as radial basis functions or neural networks. The value-gradient function is obtained by automatic differentiation. Our extensive numerical examples confirm that the accuracy and the robustness are both significantly improved in comparison to only solving the HJE in the same policy iteration method. Finally, we remark that a preliminary idea in this paper has appeared in the authors' recent handbook [11] on a review of machine learning and control theory. Here we present the full development and propose detailed numerical methods based on machine learning, with emphasis on theoretical proof of L^2_α convergence.

The paper is organized as follows. Section 2 presents the problem setup for the optimal control problem and a review of HJE and the Pontryagin maximum principle (PMP), with their connections to the theory of optimal control. Section 3 gives our new formulation in terms of the value-gradient function, with the convergence analysis of the iterative scheme. Section 4 presents our main algorithms, and section 5 introduces our numerical examples. Section 6 includes some discussions on generalization and ends with a brief conclusion.

2. Problem formulation and review of HJE.

2.1. Discounted deterministic control problem in infinite horizon. The optimal control problem in our study aims at minimizing the cost function with a discount factor $\rho \geq 0$: as the following:

$$J_x(u(\cdot)) := \int_0^{+\infty} e^{-\rho t}\, l(x(t), u(t))\mathrm{d}t \tag{2.1}$$

subject to the state equation

$$\begin{cases} \mathrm{d}x(t) = g(x(t), u(t))\mathrm{d}t, \\ x(0) = x, \end{cases} \tag{2.2}$$

where $x(\cdot):\mathbb{R}\to\mathbb{R}^d$ is the state variable, $u(\cdot):\mathbb{R}\to\mathbb{R}^p$ is the control function such that $\int_0^{+\infty} e^{-\rho t}|l(x(t),u(t))|\mathrm{d}t<\infty$, and $u(t)\in\mathcal{U}_{ad}$ a.e. t, in which $\mathcal{U}_{ad}$ is a nonempty closed convex subset of $\mathbb{R}^p$.

A feedback control u means there is a function $a(\cdot)$ in the state variable x: $\mathbb{R}^d\to\mathbb{R}^p$, such that the control $u(t)=a(x(t))$ with $x(t)$ satisfying the ODE (2.2) in the autonomous form: $\mathrm{d}x(t)=g(x(t),a(x(t)))\mathrm{d}t$. Throughout the paper, we shall use $g(x,a)$ and $g(x,u)$ interchangeably for the function g. Also, $g(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^p\to\mathbb{R}^d$ and $l(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^p\to\mathbb{R}$ carry the following assumptions [10].

Assumption 2.1. There exist some positive constants $\bar{g}$, $\bar{g}_2$, $\bar{l}$, $\bar{l}_1$, $\bar{l}_2$, c_0, c_s and a matrix c in $\mathbb{R}^{p\times p}$ with its norm $\|c\|=\bar{c}$, such that

A1. $g(x,a)=g_1(x)+c^\top a:\mathbb{R}^d\times\mathbb{R}^p\to\mathbb{R}^d$ and

$$\begin{aligned}&\|g(x,a)\|\le\bar{g}\,(1+\|x\|+\|a\|)\,;\|D_x g(x,a)\|\le\bar{g};\\&\sum_{i=1}^{d}\|D_x(\partial_{x_i}g(x,a))\|\le\frac{\bar{g}_2}{1+\|x\|},\end{aligned}\tag{2.3}$$

where x_i is the ith component of x.

A2. $l(x,a):\mathbb{R}^d\times\mathbb{R}^p\to\mathbb{R}$ is strictly convex and satisfies

$$\begin{aligned}&|l(x,a)|\le\bar{l}\left(1+\|x\|^2+\|a\|^2\right);\\&|l(x,a)-l(x',a')|\le\bar{l}\big((1+\max(\|x\|,\|x'\|)+\max(\|a\|,\|a'\|))\\&\qquad\qquad(\|x-x'\|+\|a-a'\|)\big);\\&\|\nabla_a l(x,a)\|\ge\bar{l}_1\|a\|-c_0;\ \left\|(\nabla_a\nabla_a^\top)l(x,a)\right\|\ge c_s,\end{aligned}\tag{2.4}$$

and the norm of all the second-order derivatives,
i.e., $\|(\nabla_x\nabla_a^\top)l(x,a)\|$ $\|(\nabla_a\nabla_x^\top)l(x,a)\|$, and $\|(\nabla_x\nabla_x^\top)l(x,a)\|$ are bounded by $\bar{l}_2$ from above.

The value function $\Phi(x)$ is defined by

$$\Phi(x)=\inf_{a(\cdot)\in\mathcal{U}_{ad}}J_x(a(\cdot)).\tag{2.5}$$

Note that $\Phi(x)$ also refers to the solution of the HJE in our paper; however, it can be shown by the verification theorem that the solution of the HJE is the value function (see Theorem 3.4). So, for the sake of simplicity we do not distinguish between these two with different notation .

Notation. ∇ and $(\nabla\nabla^\top)$ refer to the gradient and Hessian matrix, respectively, of a scalar function. In general, D is used for the derivatives of a vector-valued function, i.e., the Jacobi matrix. For example, $D_x g(x,a)$ refers to the Jacobi matrix in the x variable with (i,j) entry $\frac{\partial g_i}{\partial x_j}(x,a)$. $D_x^{\mathsf{T}}g$ means the transpose of the Jacobi matrix $D_x g$.

2.2. Hamilton–Jacobi equation. By the theory of dynamic programming, the value function $\Phi(\cdot)$ of (2.5) satisfies the (stationary) HJE

$$\rho\Phi(x)=g(x,\hat{a}(x))\cdot\nabla\Phi(x)+l(x,\hat{a}(x)),\tag{2.6}$$

where the optimal policy is

$$\hat{a}(x)\in\operatorname{argmin}_a\,[g(x,a)\cdot\nabla\Phi(x)+l(x,a)].\tag{2.7}$$

We drop the possible constraint $a \in \mathcal{U}_{ad}$ under argmin or min for convenience. The first equation, (2.6), is a *linear* stationary hyperbolic PDE with advection velocity field $g(x, \hat{a}(x))$. It is the convention to introduce the Hamiltonian

$$H(x, \lambda, a) := g(x, a) \cdot \lambda + l(x, a),$$

and the HJE can be written as

$$\rho \Phi(x) = \min_a H(x, \nabla \Phi, a). \tag{2.8}$$

2.3. The Pontryagin maximum principle. The PMP generally refers to the first-order necessary optimality conditions for problems of optimal control [47]. For the optimal control problem specified in section 2.1, the PMP takes the form

$$\frac{\mathrm{d}}{\mathrm{d}t} x^*(t) = H_\lambda(x^*, \lambda^*, u^*) = g(x^*, u^*); \tag{2.9a}$$

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}(e^{-\rho t}\lambda^*(t)) &= -e^{-\rho t} H_x(x^*, \lambda^*, u^*) \\ &= -e^{-\rho t}[\nabla_x l(x^*, u^*) + D_x^{\mathsf{T}} g(x^*, u^*)\lambda^*]; \end{aligned} \tag{2.9b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(e^{-\rho t} v^*(t)) = -e^{-\rho t} l(x^*, u^*), \tag{2.9c}$$

where $u^*(t) \in \mathcal{U}_{ad}$ is as defined by $\hat{a}(x^*(t))$ in (2.7), i.e.,

$$u^*(t) = \operatorname*{argmin}_u H(x^*(t), u, \lambda^*(t)).$$

$\lambda^*(t)$ is the costate or adjoint variable, and $v^*(t)$ is the cost. Note that (2.9a) has the initial condition $x^*(0) = x$, while (2.9b) and (2.9c) have the terminal conditions vanishing at infinity: $e^{-\rho t}\lambda^*(t) \to 0$ and $e^{-\rho t} v^*(t) \to 0$.

2.4. Value iteration and policy iteration for the HJE. Based on (2.6) and (2.7) as a fixed-point problem for the pair Φ and $\hat{a}$, many iterative computational methods have been developed [7, 8, 29]. They can roughly be divided into two categories: value iteration and policy iteration, which are central concepts in reinforcement learning [50].

In our model of (2.8), the value iteration, roughly speaking, refers to the sequence of functions recursively defined by

$$\Phi^{(k+1)}(x) := \rho^{-1} \min_a \left[g(x, a) \cdot \nabla \Phi^{(k)}(x) + l(x, a) \right] \ \forall x. \tag{2.10}$$

By contrast, the policy iteration requires one to solve the so-called generalized HJE. It starts with an initial policy function $a^{(0)}$ and runs the iteration from $a^{(k)}$ to $a^{(k+1)}$ as in Algorithm 2.1.

Step 1 is usually referred to as *policy evaluation*. Step 2 is usually referred to as *policy improvement,* and $a^{(k+1)}$ is the *greedy policy*.

The policy iteration is known to have superlinear convergence in many cases, provided the initial guess is sufficiently close to the solution and generally behaves better than the value iteration [1]. The convergence of policy iteration can be found in [48].

3. Formulation for value-gradient functions. We start by presenting our main theoretic results and deriving the new system of PDEs for the gradient of the value function.

Algorithm 2.1 Policy iteration (successive approximation) for the HJE.

1. Solve the linear PDE (2.6) for the value function $\Phi^{(k+1)}$ with the given policy $\hat{a} = a^{(k)}$:

$$\rho\Phi^{(k+1)}(x) = g(x, a^{(k)}(x)) \cdot \nabla\Phi^{(k+1)}(x) + l(x, a^{(k)}(x)). \tag{2.11}$$

   This linear equation is referred to as generalized HJE.
2. Then, $a^{(k+1)}$ is obtained from the optimization subproblem (2.7) pointwisely for each x:

$$a^{(k+1)}(x) := \operatorname*{argmin}_a [g(x,a) \cdot \nabla\Phi^{(k+1)}(x) + l(x,a)].$$

3.1. Equation for the value-gradient functions. Define the *value-gradient* function

$$\lambda(x) = \nabla\Phi(x);$$

then the HJE (2.6) reads

$$\rho\Phi(x) = g(x, \hat{a}(x)) \cdot \lambda(x) + l(x, \hat{a}(x)), \tag{3.1}$$

where $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. Now differentiating both sides w.r.t. x_i, we have

$$\begin{aligned}\rho\lambda_i(x) = \sum_n \lambda_n(x) \left\{ \left(\frac{\partial}{\partial x_i} + \sum_j \frac{\partial \hat{a}_j}{\partial x_i} \frac{\partial}{\partial a_j} \right) g_n(x, \hat{a}(x)) \right\} \\ + \sum_n g_n(x, \hat{a}(x)) \frac{\partial \lambda_n}{\partial x_i}(x) + \left(\frac{\partial}{\partial x_i} + \sum_j \frac{\partial \hat{a}_j}{\partial x_i} \frac{\partial}{\partial a_j} \right) l(x, \hat{a}(x)),\end{aligned}$$

where λ_i and $\hat{a}_i$ are the ith components of λ and $\hat{a}$, respectively. We assume that the Hamiltonian minimization (2.7) has the unique minimizer $\hat{a}(x)$ which is continuously differential. Then the minimizer $\hat{a}(x)$ satisfies the following first-order necessary condition:

$$\sum_n \frac{\partial g_n}{\partial a_j}(x, \hat{a}) \lambda_n(x) + \frac{\partial l}{\partial a_j}(x, \hat{a}) = 0 \qquad \forall j. \tag{3.2}$$

With both of the above equalities, we have that $\lambda(x) = (\lambda_1, \ldots, \lambda_d)$ satisfies the system of linear hyperbolic PDEs

$$\rho\lambda_i = \sum_n g_n \frac{\partial \lambda_n}{\partial x_i} + \sum_n \lambda_n \frac{\partial g_n}{\partial x_i} + \frac{\partial l}{\partial x_i}, \tag{3.3}$$

or in the compact form,

$$\rho\lambda(x) = D^{\mathsf{T}}\lambda(x) g(x, \hat{a}(x)) + D_x^{\mathsf{T}} g(x, \hat{a}(x)) \lambda(x) + \nabla_x l(x, \hat{a}(x)), \tag{3.4}$$

and $\hat{a}(x)$ defined by (2.7) can now be written as

$$\hat{a}(x) = \operatorname{argmin}_a [g(x,a) \cdot \lambda(x) + l(x,a)]. \tag{3.5}$$

Equations (3.4) and (3.5) are coupled as (2.6) and (2.7) in the HJE and serve as the foundation for the new development of the algorithms, based on the policy iteration method.

Given a policy $\hat{a}$, the system of coupled PDEs (3.4) is a closed form involving only the dynamic function g and the running cost function l; it does not need other information, such as the value function. It plays a role similar to the generalized HJE (2.6) for the value function Φ. Equations (3.4) and (3.5) together can replace the traditional dynamic programming in the form of HJE if Φ is sufficiently smooth. The main focus of our work is how to develop efficient numerical methods from this formulation of the gradient of the value function.

Since $\lambda(x)$ is the gradient of the value function Φ, $D\lambda(x) = \nabla^2\Phi(x)$ should be symmetric, i.e., $D\lambda = D^{\mathsf{T}}\lambda$. Then the value-gradient satisfies

$$\rho\lambda_i(x) = \nabla\lambda_i(x)\cdot g(x,\hat{a}(x)) + \sum_n \frac{\partial g_n}{\partial x_i}\lambda_n(x) + \frac{\partial l}{\partial x_i}(x,\hat{a}(x)) \tag{3.6}$$

or

$$\rho\lambda(x) = (D\lambda)g + (D_x^{\mathsf{T}}g)\lambda(x) + \nabla_x l, \tag{3.7}$$

where $\hat{a}(x)$ is defined, as in (3.5), as the unique minimizer of the Hamiltonian $H(x,\lambda(x))$. In addition, if $\lambda(x)$ satisfies the systems of PDEs (3.6), then for x^* as the optimal trajectory satisfying the characteristics equation (2.9a), $\lambda^*(t) := \lambda(x^*(t))$ satisfies (2.9b). The conclusion that $\lambda^*(t) := \lambda(x^*(t))$ satisfies (2.9b) follows from the fact that

$$\frac{\mathrm{d}}{\mathrm{d}t}\lambda(t) = (D\lambda)g = \rho\lambda^*(t) - \left[(D_x^{\mathsf{T}}g)\lambda(x^*) + \nabla_x l\right].$$

The advantage of (3.6) over (3.3) is that the advection terms $D\lambda_i\cdot g$ are now decoupled for each component i and are the same as in the GHJE (2.6). This property will allow us to develop a fully parallel iterative method.

3.2. Policy iteration for the value-gradient. The natural idea for solving the PDEs for (3.6) and the minimization for $\hat{a}$ in (3.5) is to use the policy iteration by recursively solving (3.6) and (3.5) as the policy iteration for the value function dictated in section 2.4, that is, start with an initial policy function $a^{(0)}$ with $k=0$:

1. Solve the system (3.6) with the given policy $\hat{a} = a^{(k)}$ to have $\lambda^{(k+1)}$;
2. $a^{(k+1)}$ is obtained from the optimization subproblem (2.7).

This iteration will produce a sequence of pairs $(a^{(k)},\lambda^{(k)})$, $k\geq 1$. The main task is then to solve (3.6) (or (3.7)), the system of linear PDEs for $\lambda(x)$, with a given policy a. We will first propose the method for this system of linear PDEs; more details are given in section 4. We summarize our main algorithm, **policy iteration based on λ (PI-lambda),** below.

The merit of (3.8) is that the components of $\lambda^{(k+1)}(x)$ are completely decoupled and can be solved in parallel. Each equation of these d components is in exactly the same form as the GHJE (2.11) for the value function. So the method of characteristics, which will be detailed in the next section, can be applied to both the GHJE (2.11) and the system (3.8).

3.3. Convergence analysis for PI-lambda. In this subsection, we will state and prove our main theorem, Theorem 3.5, which states that the PI-lambda algorithm converges linearly in the L^2_α sense (see (3.13)) for a suitable choice of exponent α

Algorithm 3.1 PI-lambda: policy iteration based on λ.

1. For $i=1,\dots,d$, solve the PDE for each $\lambda_i^{(k+1)}$ in parallel,

$$\begin{aligned}\rho\lambda_i^{(k+1)}(x)-D\lambda_i^{(k+1)}(x)\cdot g(x,a^{(k)}(x))\\ =\sum_n \frac{\partial g_n}{\partial x_i}\lambda_n^{(k)}(x)+\frac{\partial l}{\partial x_i}(x,a^{(k)}(x)),\end{aligned} \tag{3.8}$$

   with the given policy $\hat{a}=a^{(k)}$ having $\lambda^{(k+1)}=(\lambda_1^{(k+1)},\dots,\lambda_d^{(k+1)})$;
2. $a^{(k+1)}$ is obtained from the optimization subproblem (2.7):

$$a^{(k+1)}(x)=\operatorname*{argmin}_a [g(x,a)\cdot\lambda^{(k+1)}(x)+l(x,a)].$$

in a weight factor. The proof will need two important lemmas, Lemmas 3.1 and 3.2, which are proved in the supplementary material (ex_supplement.pdf [local/web 1.84MB]). $\lambda^{(k)}(x)$ and $a^{(k)}(x)$ stand for the value-gradient and control function of the kth iteration in PI-lambda, respectively.

LEMMA 3.1. *Under Assumptions* 2.1*, at the kth iteration of the value-gradient, if there exist constants* $\bar{\lambda}^{(k)}$, $\bar{\lambda}'^{(k)}$, $\bar{a}^{(k)}$, $\bar{a}'^{(k)}$ *such that*

$$\begin{aligned}&\|\lambda^{(k)}(x)\|\le\bar{\lambda}^{(k)}(1+\|x\|),\left\|D\lambda^{(k)}(x)\right\|\le\bar{\lambda}'^{(k)},\\ &\|a^{(k)}(x)\|\le\bar{a}^{(k)}(1+\|x\|),\ \left\|Da^{(k)}(x)\right\|\le\bar{a}'^{(k)},\end{aligned}$$

and if

$$\rho>\bar{g}(1+\bar{a}^{(k)})+\bar{c}\bar{a}'^{(k)},$$

then

$$\begin{aligned}&\|\lambda^{(k+1)}(x)\|\le\bar{\lambda}^{(k+1)}(1+\|x\|),\left\|D\lambda^{(k+1)}(x)\right\|\le\bar{\lambda}'^{(k+1)},\\ &\|a^{(k+1)}(x)\|\le\bar{a}^{(k+1)}(1+\|x\|),\ \left\|Da^{(k+1)}(x)\right\|\le\bar{a}'^{(k+1)},\end{aligned}$$

where the constants

$$\bar{\lambda}^{(k+1)}=\frac{\bar{l}+\bar{l}\bar{a}^{(k)}+\bar{g}\bar{\lambda}^{(k)}}{\rho-\bar{g}(1+\bar{a}^{(k)})}>0, \tag{3.9}$$

$$\bar{\lambda}'^{(k+1)}=\frac{\bar{l}_2+\bar{l}_2\bar{a}'^{(k)}+\bar{g}_2\bar{\lambda}^{(k)}+\bar{g}\bar{\lambda}'^{(k)}}{\rho-(\bar{g}+\bar{c}\bar{a}'^{(k)})}>0, \tag{3.10}$$

$$\bar{a}^{(k+1)}=\frac{\bar{c}\bar{\lambda}^{(k+1)}+c_0}{\bar{l}_1}, \tag{3.11}$$

$$\bar{a}'^{(k+1)}=\frac{\bar{l}_2+\bar{c}\bar{\lambda}'^{(k+1)}}{c_s}. \tag{3.12}$$

LEMMA 3.2. *There exists a constant* ρ_1 *such that the sequence* $\{\bar{a}^{(k)}\}$, $\{\bar{a}'^{(k)}\}$, $\{\bar{\lambda}^{(k)}\}$, $\{\bar{\lambda}'^{(k)}\}$ *in Lemma* 3.1 *are uniformly bounded by constants* C_1, C_2, C_3, C_4, *respectively, if* $\rho>\rho_1$ *and the initial seed satisfies*

$$\|\lambda^{(0)}(x)\| \le C_3(1+\|x\|),\ \left\|D\lambda^{(0)}(x)\right\| \le C_4,$$

where the constants are

$$C_1 = \sqrt{\frac{\bar{c}\bar{l}(1+\frac{c_0}{\bar{l}_1})}{\bar{g}\bar{l}_1}} + \frac{c_0}{\bar{l}_1};\quad C_2 = \frac{1}{c_s}\left(\bar{l}_2 + \sqrt{c_s\bar{l}_2 + \bar{l}_2^2 + \bar{g}_2\sqrt{\frac{c_s(\bar{l}^2+c_0\bar{l})}{\bar{g}}}}\right);$$

$$C_3 = \sqrt{\frac{\bar{l}_1\bar{l}(1+\frac{c_0}{\bar{l}_1})}{\bar{g}\bar{c}}};\qquad C_4 = \frac{1}{\bar{c}}\sqrt{c_s\bar{l}_2 + \bar{l}_2^2 + \bar{g}_2\sqrt{\frac{c_s(\bar{l}^2+c_0\bar{l})}{\bar{g}}}}.$$

We then state the regularity of $\Phi(x)$ in Lemma 3.3.

LEMMA 3.3. *For the solution of HJE* (2.6)*, there exist positive constants* C_3 *and* C *such that* for all $x, y \in \mathbb{R}^d$,

$$|\Phi(x)-\Phi(y)| \le C_3(1+\|x\|+\|y\|)\|x-y\|,$$
$$|\Phi(x)| \le C(1+\|x\|^2),$$

where $C_3 = \sqrt{\frac{\bar{l}_1\bar{l}\left(1+\frac{c_0}{\bar{l}_1}\right)}{\bar{g}\bar{c}}}$ *as defined in Lemma* 3.2*, and* $C := \max\{\hat{C} + C_3\bar{C} + C_3\bar{C}^2 + \frac{1+C_3\bar{C}+\bar{C}}{2}, \frac{1+C_3\bar{C}+\bar{C}}{2} + C_3^2\}$ *for two positive numbers* $\bar{C}, \hat{C}$ *such that there exists* $\hat{x} \in \mathbb{R}^d$ *satisfying* $\|\hat{x}\| \le \bar{C}$ *so that* $|\Phi(\hat{x})| \le \hat{C}$.

In Theorem 3.4, we show that the value function and the solution for HJE are the same by the verification theorem.

THEOREM 3.4. *Let* $\Phi(\cdot) \in C^2(\mathbb{R}^d;\mathbb{R})$ *be the solution of HJE* (2.6)*. Then for all* $x \in \mathbb{R}^d$ *we have the following:*

1. $\Phi(x) \le J(x, a(x))$ *for every* $a(\cdot) \in \mathcal{U}_{ad}$.
2. *If there exists an admissible control* a^* *such that*

$$a^*(x) = \operatorname*{argmin}_a \left[g(x,a)\cdot\nabla\Phi(x) + l(x,a)\right] \text{a.e. in } x \in \mathbb{R}^d,$$

then $\Phi(x) = J(x, a^*(\cdot))$.

Next, we state our main theorem that shows the convergence of the **PI-lambda** algorithm.

THEOREM 3.5. *Under Assumption* 2.1*, for any* $\alpha > 1$*, there exists a large enough* ρ_2 *such that if* $\rho > \rho_2$*, then we define*

$$e^{(k)} := \int_{\mathbb{R}^d} \frac{\|\lambda^{(k)}(x) - \lambda^{(k-1)}(x)\|^2}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x. \tag{3.13}$$

We have $e^{(k+1)} \le \eta e^{(k)}$ *with* $\eta \in (0,1)$*. Therefore* $\{\lambda^{(k)}\}$ *forms a Cauchy sequence in the* L^2_α *sense.*

Remark. Note that $\rho_2 \ge \rho_1$, which suggests that if ρ satisfies the inequality condition in Theorem 3.5, then it satisfies the inequality condition in Lemma 3.2.

Proof. Recall that in (3.8), $\lambda^{(k)}(x)$ and $\lambda^{(k+1)}(x)$ are defined by

$$\begin{aligned}\rho\lambda^{(k)}(x) = {} & D\lambda^{(k)}(x)g\left(x, a^{(k-1)}(x)\right) \\ & + D_x g\left(x, a^{(k-1)}(x)\right)\lambda^{(k-1)}(x) + \nabla_x l\left(x, a^{(k-1)}(x)\right)\end{aligned} \tag{3.14}$$

and

$$\begin{aligned}\rho\lambda^{(k+1)}(x) = {} & D\lambda^{(k+1)}(x)g(x, a^{(k)}(x)) \\ & + D_x g\left(x, a^{(k)}(x)\right)\lambda^{(k)}(x) + \nabla_x l\left(x, a^{(k)}(x)\right).\end{aligned}$$

Then the difference in $\lambda^{(k+1)} - \lambda^{(k)}(x)$ is

$$\begin{aligned}&\rho(\lambda^{(k+1)} - \lambda^{(k)}(x)) \\ &= D\lambda^{(k+1)}(x)g(x, a^{(k)}(x)) - D\lambda^{(k)}(x)g(x, a^{(k-1)}(x)) + D_x g(x, a^{(k)}(x))\lambda^{(k)}(x) \\ &\quad - D_x g(x, a^{(k-1)}(x))\lambda^{(k-1)}(x) + \nabla_x l(x, a^{(k)}(x)) - \nabla_x l(x, a^{(k-1)}(x)) \\ &= D\left(\lambda^{(k+1)}(x) - \lambda^{(k)}(x)\right) g(x, a^{(k)}(x)) \\ &\quad + D\lambda^{(k+1)}(x)\left(g(x, a^{(k)}(x)) - g(x, a^{(k-1)}(x))\right) \\ &\quad + \left(D_x g(x, a^{(k)}(x)) - D_x g(x, a^{(k-1)}(x))\right)\lambda^{(k)}(x) + \nabla_x l(x, a^{(k)}(x)) \\ &\quad - \nabla_x l(x, a^{(k-1)}(x)) + D_x g(x, a^{(k-1)}(x))\left(\lambda^{(k)}(x) - \lambda^{(k-1)}(x)\right).\end{aligned}$$

We consider the error in the following L^2_α sense with $\alpha > 1$. Taking the inner product of $\lambda^{(k+1)} - \lambda^{(k)}(x)$ with the previous expression, we have

$$\begin{aligned}\rho e^{(k+1)} &:= \rho \int_{\mathbb{R}^d} \frac{\left\|\lambda^{(k+1)}(x) - \lambda^{(k)}(x)\right\|^2}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \\ &\le \frac{1}{2}\left\| \int_{\mathbb{R}^d} D(\|\lambda^{(k+1)}(x) - \lambda^{(k)}(x)\|^2) \frac{g(x, a^{(k)}(x))}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \right\| \\ &\quad + \left\| \int_{\mathbb{R}^d} D\lambda^{(k+1)}(x)\Big(g(x, a^{(k)})(x) - g(x, a^{(k-1)}(x))\Big) \frac{\lambda^{(k+1)}(x) - \lambda^{(k)}(x)}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \right\| \\ &\quad + \left\| \int_{\mathbb{R}^d} \Big(D_x g(x, a^{(k)}(x)) - D_x g(x, a^{(k-1)}(x))\Big)\lambda^{(k)}(x) \frac{\lambda^{(k+1)}(x) - \lambda^{(k)}(x)}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \right\| \\ &\quad + \left\| \int_{\mathbb{R}^d} \Big(\nabla_x l(x, a^{(k)}(x)) - \nabla_x l(x, a^{(k-1)}(x))\Big) \frac{\lambda^{(k+1)}(x) - \lambda^{(k)}(x)}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \right\| \\ &\quad + \left\| \int_{\mathbb{R}^d} D_x g(x, a^{(k-1)}(x)) \frac{(\lambda^{(k)}(x) - \lambda^{(k-1)}(x))\left(\lambda^{(k+1)}(x) - \lambda^{(k)}(x)\right)}{(1+\|x\|^2)^{2\alpha}} \mathrm{d}x \right\| \\ &:= I_1 + I_2 + I_3 + I_4 + I_5.\end{aligned}$$

By Lemma 3.1 and Assumption 2.1, we have

$$\|g(x, a^{(k)})\| \le \bar{g}(1 + C_1)(1 + \|x\|).$$

Integration by part for the first term I_1 gives

$$
\begin{aligned}
I_1 \leq & \frac{1}{2}\int_{\mathbb{R}^d} \frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|^2}{(1+\|x\|^2)^{2\alpha}}\Big[\Big\|D_x g(x,a^{(k)}(x))\Big\| \\
& + \Big\|D_a g(x,a^{(k)}(x)\Big\|\Big\|Da^{(k)}(x)\Big\| + \frac{4\alpha\|x\|}{1+\|x\|^2}\bar{g}(1+C_1)(1+\|x\|)\Big]\mathrm{d}x \\
\leq & \frac{1}{2}\int_{\mathbb{R}^d} \frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|^2}{(1+\|x\|^2)^{2\alpha}}\Big[\Big\|D_x g(x,a^{(k)}(x))\Big\| \\
& + \bar{c}\Big\|Da^{(k)}(x)\Big\| + 5\alpha\bar{g}(1+C_1)\Big]\mathrm{d}x \\
\leq & e^{(k+1)}\left(\bar{g}+\bar{c}C_2+5\alpha\bar{g}(1+C_1)\right),
\end{aligned}
\tag{3.15}
$$

using $\frac{\|x\|(1+\|x\|)}{1+\|x\|^2} < \frac{5}{4}$ for all $x \in \mathbb{R}$ in the last second equation.
By the mean value theorem for $g(x,\cdot)$ and Lemma 3.1, the second term I_2 is

$$
\begin{aligned}
I_2 = & \int_{\mathbb{R}^d}\Big\|D\lambda^{(k+1)}(x)D_a g\Big(x,a^{(k-1)}(x)+\delta_1(x)(a^{(k)}(x)-a^{(k-1)}(x))\Big) \\
& \cdot (a^{(k)}(x)-a^{(k-1)}(x))\frac{\lambda^{(k+1)}(x)-\lambda^{(k)}(x)}{(1+\|x\|^2)^{2\alpha}}\Big\|\mathrm{d}x \\
\leq & \bar{c}C_4\int_{\mathbb{R}^d}\frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|\|a^{(k)}(x)-a^{(k-1)}(x)\|}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x,
\end{aligned}
$$

where a function $\delta_1(x)$ is $\mathbb{R}^d \to \mathbb{R}$.
The third term $I_3 = 0$ because $D_x g(x,a)$ is independent of a. The fourth term I_4 is

$$
I_4 \leq \bar{l}_2\int_{\mathbb{R}^d}\frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|\|a^{(k)}(x)-a^{(k-1)}(x)\|}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x.
$$

The last term I_5 is

$$
\begin{aligned}
I_5 := & \frac{1}{2}\int_{\mathbb{R}^d}\frac{\left\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\right\|^2}{(1+\|x\|^2)^{2\alpha}}\left\|D_x g(x,a^{(k-1)}(x))\right\|\mathrm{d}x \\
& + \frac{1}{2}\int_{\mathbb{R}^d}\frac{\left\|\lambda^{(k)}(x)-\lambda^{(k-1)}(x)\right\|^2}{(1+\|x\|^2)^{2\alpha}}\left\|D_x g(x,a^{(k-1)}(x))\right\|\mathrm{d}x \\
\leq & \frac{\bar{g}}{2}\left(e^{(k+1)}+e^{(k)}\right).
\end{aligned}
\tag{3.16}
$$

Next, we estimate the bound of $\|a^{(k)}(x)-a^{(k-1)}(x)\|$ by $\|\lambda^{(k)}(x)-\lambda^{(k-1)}(x)\|$. By the first-order necessary condition, we have

$$
\begin{aligned}
0 &= \nabla_a l(x,a^{(k)}) + c^\top\lambda^{(k)}(x); \\
0 &= \nabla_a l(x,a^{(k-1)}) + c^\top\lambda^{(k-1)}(x).
\end{aligned}
$$

Then, by the mean value theorem, there exists $\gamma_1^{(k+1)}(x):\mathbb{R}^d\to\mathbb{R}$ such that

$$
\begin{aligned}
(\nabla_a\nabla_a^\top)l\left(x,a^{(k-1)}(x)+\gamma_1^{(k)}(x)(a^{(k)}-a^{(k-1)}(x))\right)(a^{(k)}-a^{(k-1)}(x)) \\
+ c^\top(\lambda^{(k)}(x)-\lambda^{(k-1)}(x)) = 0.
\end{aligned}
$$

Thus

$$a^{(k)}(x)-a^{(k-1)}(x)=-\left((\nabla_a\nabla_a^\top)l\left(x,a^{(k-1)}(x)+\gamma_1^{(k)}(x)(a^{(k)}-a^{(k-1)}(x))\right)\right)^{-1}\cdot\left(c^\top(\lambda^{(k)}(x)-\lambda^{(k-1)}(x))\right).$$

Since $\|(\nabla_a\nabla_a^\top)l(\cdot,\cdot)\|>c_s$, we have

$$\left\|a^{(k)}(x)-a^{(k-1)}(x)\right\|\le\frac{\bar{c}}{c_s}\left\|\lambda^{(k)}(x)-\lambda^{(k-1)}(x)\right\|,$$

and then

(3.17)
$$I_2\le\frac{\bar{c}^2C_4}{c_s}\int_{\mathbb{R}^d}\frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|\|\lambda^{(k)}(x)-\lambda^{(k-1)}(x)\|}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x\le\frac{\bar{c}^2C_4}{2c_s}(e^{(k+1)}+e^{(k)})$$

and

$$I_4\le\bar{l}_2\int_{\mathbb{R}^d}\frac{\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\|\|\lambda^{(k)}(x)-\lambda^{(k-1)}(x)\|}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x\le\frac{\bar{l}_2}{2}(e^{(k+1)}+e^{(k)}).\tag{3.18}$$

Combining (3.15), (3.16), (3.17), and (3.18), we have

$$\begin{aligned}\rho e^{(k+1)}\le{}& e^{(k+1)}\left(\bar{g}+\bar{c}C_2+5\alpha\bar{g}(1+C_1)\right)\\&+\frac{\bar{c}^2C_4}{2c_s}(e^{(k+1)}+e^{(k)})+\frac{\bar{l}_2}{2}(e^{(k+1)}+e^{(k)})+\frac{\bar{g}}{2}(e^{(k+1)}+e^{(k)}).\end{aligned}$$

Consequently,

$$e^{(k+1)}\le\frac{\frac{\bar{c}^2C_4}{2c_s}+\frac{\bar{l}_2}{2}+\frac{\bar{g}}{2}}{\rho-(\bar{g}+\bar{c}C_2+5\alpha\bar{g}(1+C_1))-\frac{\bar{c}^2C_4}{2c_s}-\frac{\bar{l}_2}{2}-\frac{\bar{g}}{2}}e^{(k)}:=\eta e^{(k)}.\tag{3.19}$$

Select ρ_2 to be

$$\rho_2=\max\left\{\rho_1,2\bar{g}+\bar{c}C_2+5\alpha\bar{g}(1+C_1)+\frac{\bar{c}^2C_4}{c_s}+\bar{l}_2\right\}.\tag{3.20}$$

Then for $\rho>\rho_2$, we have $e^{(k+1)}\le\eta e^{(k)}$ where $\eta\in(0,1)$. $e^{(k+1)}$ will converge to 0 as $k\to\infty$. That is,

$$\lim_{k\to\infty}\int_{\mathbb{R}^d}\frac{\left\|\lambda^{(k+1)}(x)-\lambda^{(k)}(x)\right\|^2}{(1+\|x\|^2)^{2\alpha}}\mathrm{d}x=0.\tag{3.21}$$

∎

Finally, we show that the sequence $\{\lambda^{(k)}\}$ does converge to the classical solution by the corollary below; the proof is shown in the supplementary material ex_supplement.pdf [local/web 1.84MB].

Corollary 3.6. *If there exists a classical solution of PDE* (3.4)*, then* $\lambda^{(k)}$ *converges to the solution in the* L^2_α *sense.*

4. Numerical methods. Our algorithm is the policy iteration based on λ, and it is clear that the main challenge is to solve the system of linear PDEs (3.8) in any dimension. It is worthwhile to point out that each PDE in (3.8) is the same type of PDE as the GHJE (2.11). So, the Galerkin approximate approach also can be applied for these equations in (3.8), but to directly aim for the high-dimensional problems, we use the method of characteristics and supervised learning.

Specifically, we first consider a family of functions, such as neural networks, with $\widehat{\Phi}(x;\theta)$ to numerically represent the value function, where $\theta \in \Theta$ is the set of parameters. The value-gradient function $\widehat{\lambda}(x;\theta) = \nabla_x \widehat{\Phi}(x;\theta)$ is then computed by automatic differentiation instead of finite difference. Second, in each policy iteration k, we compute the characteristics by numerical integrating the state dynamics, and we calculate the true value $\Phi^{(k+1)}$ and value-gradient functions $\lambda^{(k+1)}$ on the characteristics curves based on the PDEs (2.11) and (3.8). Then these labeled data $(X(t), \Phi(X(t)), \lambda(X(t)))$ are fed into the supervised learning protocol by minimizing the mean squared error to find the optimal $\theta^{(k+1)}$.

In what follows, we discuss the details of the method of characteristics for solving the PDEs (2.11) and (3.8) on characteristics curves. We drop the **PI-lambda** iteration index k in this section for notational ease.

4.1. Method of characteristics. Bearing in mind the similar form of (2.11) and (3.8), which are both hyperbolic linear PDEs with the same advection, we consider a general discussion. Given a control function $a(\cdot)$, we denote $G(x) = g(x, a(x))$ and define $X(t)$ as the characteristic curve satisfying the following ODE with an arbitrary initial state $X_0 \in \mathbb{R}^d$:

$$\begin{cases} \mathrm{d}X(t) = G(X)dt, \\ X(0) = X_0. \end{cases} \tag{4.1}$$

We consider the PDE of the function v,

$$\rho v(x) - Dv(x) \cdot G(x) = R(x), \tag{4.2}$$

where the source term R is given. Note that (2.11) and (3.8) are special cases of (4.2) with different R terms. Along the characteristic curve $X(t)$, by (4.1) and (4.2) we derive that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{-\rho t} v(X(t))\right] = -\rho e^{-\rho t} v(X(t)) + e^{-\rho t} Dv(X(t)) \cdot \frac{\mathrm{d}X}{\mathrm{d}t} = -e^{-\rho t} R(X(t)).$$

After taking integration in time,

$$\lim_{s \to +\infty} e^{-\rho s} v(X(s)) - e^{-\rho t} v(X(t)) = \int_t^{+\infty} -e^{-\rho \tau} R(X(\tau)) \mathrm{d}\tau. \tag{4.3}$$

As time s tends to infinity, suppose ρ is large enough; then we have

$$v(X(t)) = \int_t^{+\infty} e^{-\rho(\tau - t)} R(X(\tau)) \mathrm{d}\tau.$$

4.2. Compute the value function and the gradient on the characteristics. We apply the above method of characteristics to compute the value function Φ and the gradient $\lambda = \nabla \Phi$. For the value function in (2.6), the R function in (4.2) is $l(x, a(x))$. Then Φ in (2.6) has the values on $X(t)$:

$$\Phi(X(t)) = \int_t^{+\infty} e^{-\rho(\tau - t)} l(X(\tau), a(X(\tau)))\, \mathrm{d}\tau. \tag{4.4}$$

For $\lambda^{(k+1)}$ in (3.8), for each component i, $R(x)$ in (4.2) now refers to the right-hand side in function (3.8); then

$$\lambda_i^{(k+1)}(x(t)) = \int_t^{+\infty} e^{-\rho(\tau-t)} r_i^{(k)}(\tau)\,\mathrm{d}\tau, \tag{4.5}$$

where

$$r_i^{(k)}(\tau) = \sum_n \frac{\partial g_n}{\partial x_i}\lambda_n^{(k)}(X(\tau)) + \frac{\partial l}{\partial x_i}(X(\tau), a^{(k)}(X(\tau))).$$

4.3. Supervised learning: Interpolate the characteristic curve to the whole space. With a characteristic curve $X(\cdot)$ computed from (4.1), we can obtain the value of the value function Φ and the gradient $\lambda_i = \frac{\partial \Phi}{\partial x_i}$, $i = 1, \ldots, d$, along $X(t)$ *simultaneously*. By running multiple characteristic curves starting from a set of the initial points $\{X_0^{(n)}, 1 \le n \le N\}$, which are generally sampled uniformly, we obtain a collection of observations of $\Phi(X^{(n)}(t))$ and $\lambda(X^{(n)}(t))$ on these characteristics trajectories $\{X^{(n)}(t) : t \ge 0, 1 \le n \le N\}$. In practice, the continuous path $X^{(n)}(t)$ is represented by a finite number of "images" on the curve, and these images on each curve are chosen to be of roughly equal distance from each neighboring image.

To interpolate the labeled data from the computed curves to the whole space, a family of approximate functions $\widehat{\Phi}(x;\theta)$ should be proposed first by the users, which could be a Galerkin form of basis functions, radial basis functions, or neural networks, etc. Then the parameters θ is found by minimizing the following loss function $L(\theta)$ combining two mean square errors:

$$\begin{aligned} L(\theta) = \mu \sum_{n=1}^{N} \int \left| \Phi(X^{(n)}(t)) - \widehat{\Phi}_\theta(X^{(n)}(t)) \right|^2 \mathrm{d}t \\ + (1-\mu) \sum_{n=1}^{N} \int \left\| \lambda(X^{(n)}(t)) - \nabla \widehat{\Phi}_\theta(X^{(n)}(t)) \right\|^2 \mathrm{d}t, \end{aligned} \tag{4.6}$$

where $0 \le \mu \le 1$ is a factor for balancing the loss from the value function, and the gradient. $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^d$. The gradient $\nabla\widehat{\Phi}_\theta$ is the gradient w.r.t. the state variable x and computed by automatic differentiation. The training process of the models is to minimize the loss function (4.6) w.r.t. θ by some standard gradient-descent optimization method, such as ADAM [38].

The following remarks explain our practical algorithm more clearly

- Our algorithmic framework is the policy iteration based on λ. So, the computation of the data points on the characteristics curves and the training of the loss (4.6) are performed at each policy iteration k. One can adjust the number of characteristic trajectories N and the number of training steps (the steps within the minimization procedure for the loss function). The trajectory number N determines the amount of data, and the training step determines the accuracy of supervised learning.
- The loss (4.6) simply writes the contribution from each trajectory in the continuous L_2 integration in time. Practically, this integration is represented by the sum from each discrete point on the curves. For better fitting of the function $\widehat{\Phi}_\theta$, these points are not assumed to correspond to equal step sizes in the time variable but should be arranged to spread out evenly in space. There are many practical ways to achieve this target, such as using the arc-length parametrization or setting a small ball as the forbidden region for each

prior point. Our numerical tests use the arc-length parametrization for each trajectory.

- The choice of the initial states $\{X_0^{(n)} : 1 \le n \le N\}$ can affect how the corresponding characteristics curves behave in the space, and we hope these finite numbers of curves can explore the space efficiently. Some adaptive ideas are worth trying in practice. For example, more points may be sampled where the residual of the HJE is larger. However, since all of the characteristics curves nonlinearly depend on the initial states, we use the uniform distribution in our numerical tests for simplicity.

5. Numerical examples. This section presents numerical experiments showing the advantage of our new method of policy iteration using λ and Φ over the method only using Φ. We test three problems in all: Linear-quadratic regulator, advertising process, and cart-pole balancing task. Due to space limitations, we only show the last example in this section. The remaining two can be found in the supplementary material ex_supplement.pdf [local/web 1.84MB]. Additional experiments on high-dimensional cases, choices of loss function, choices of network structure, and approximation of control $a(x)$ are also shown in the supplementary material ex_supplement.pdf [local/web 1.84MB].

The cart-pole balancing task is a 4-dim nonlinear case [3]. The physical model of this task includes a car, a pole, and a ball. The ball is connected to one end of the pole, and the other end of the pole is fixed to the car. The pole can rotate around the end fixed to the car, while the car is put on a flat surface, being able to move left or right. The aim of this task is to balance the pole in the upright vertical position.

The state variable has four dimensions: the angular velocity of the ball, denoted by ω; the included angle of the pole and the vertical direction, denoted by $\psi \in [-\pi, \pi]$; the velocity of the car, denoted by v; and the position of the car, denoted by z. The control of this problem is the force applied to the car, denoted by F.

The control problem is to let ψ be as small as possible. To eliminate the translation invariant in the horizontal position, we also want z to be small. So we aim to minimize $-\cos(\psi)$ and $|z|^2$ with the cost function

$$J(u) = \int_0^\infty e^{-\rho t} \left(-\cos(\psi(t)) + \eta |z(t)|^2 \right) dt,$$

with $\rho = 5$ and $\eta = 0.2$ and subject to the dynamical system

$$\begin{cases} \dot{\omega} = \dfrac{g \sin\psi + \frac{\left(\mu_c \operatorname{sgn}(v) - F - ml\omega^2 \sin\psi\right)\cos\psi}{m+m_c} - \frac{\mu_p \omega}{ml}}{l\left(\frac{4}{3} - \frac{m}{m+m_c}\cos^2\psi\right)}, \\ \dot{\psi} = \omega, \\ \dot{v} = \dfrac{F + ml\left(\omega^2 \sin\psi - \dot{\omega}\cos\psi\right) - \mu_c \operatorname{sgn}(v)}{m+m_c}, \\ \dot{z} = v, \\ \omega(0) = \omega_0, \psi(0) = \psi_0, v(0) = v_0, z(0) = z_0, \end{cases}$$

where m is the mass of the ball, m_c is the mass of the car, l is the length of the pole, and g is the gravitational constant. A constraint is imposed on the control: $|F| \le F_{max}$, $F_{max} \ge 0$ is the largest control we can have. These hyperparameters are set to be

$$m = 0.1,\ l = 0.5,\ m_c = 1,\ \mu_c = 5 \times 10^{-4},\ \mu_p = 2\ \times 10^{-6},\ F_{max} = 10.$$

The state variable is $x = (\omega, \psi, v, z) \in \mathbb{R}^4$ with $\psi \in [-\pi, \pi)$. The value function is approximated by a neural network with radial basis function ($n = 50$ modes). In total, there are $(2d+1)n = 450$ parameters to learn. We compute the value function on the domain $\Omega = [-2\pi, 2\pi] \times [-\pi, \pi] \times [-0.5, 0.5] \times [-2.4, 2.4]$. So, the initial values of the characteristics $X_0^{(n)}$ are uniformly sampled from Ω. But the characteristics are computed in the whole space with sufficiently long time until $e^{-\rho t}\Phi(X(t))$ and $e^{-\rho t}\lambda(X(t))$ are both sufficiently small. The error of the numerical solution $\widehat{\Phi}_\theta$ is measured by the HJE residual calculated on $N_p = 10{,}000$ points uniformly sampled from Ω:

$$res = \frac{1}{N_p}\sum_{j=1}^{N_p}\left|\rho\hat{\Phi}_\theta\left(x^{(j)}\right) - g\left(x^{(j)}, a^*\left(x^{(j)}\right)\right)\cdot\nabla\hat{\Phi}_\theta\left(x^{(j)}\right) - l\left(x^{(j)}, a^*\left(x^{(j)}\right)\right)\right|,$$

where $x^{(j)}$ is the jth data point.

To better evaluate the performance, we introduce the "successful roll-up": In a 20-second simulation ($T = 20$), if

- $|\psi(t)| < \pi/4$ lasts for at least 10 seconds, and
- $|z(t)| < 10$ for all $t \in [0, T]$,

then we call this run a "successful roll-up."

The initial conditions for measuring the successful roll-up numbers are $(\omega(0), \psi(0), v(0) = 0, z(0) = 0)$ with 100 pairs of $(\omega(0), \psi(0))$ from the 10×10 mesh grid of $[-2\pi, 2\pi) \times [-\pi, \pi)$.

We conduct two experiments for this 4-dim nonlinear case which test the performance under insufficient data or incomplete training. Here the control is solved analytically at each iteration for both experiments to better show the robustness of PI-lambda. We refer the reader to the supplementary material ex_supplement.pdf [local/web 1.84MB] for the experiments where a^* is solved numerically.

Experiment 1. In this experiment, we study how an insufficient amount of characteristics data will affect the performance. Specifically, we test the performance of trajectory numbers 2, 5, and 10, while the training for the supervised learning to minimize the loss $L(\theta)$ takes a fixed number of 100 ADAM steps [38]. Fewer trajectories mean a smaller amount of labeled data for the method of characteristics.

Table 1 shows the results when μ varies for each test. For each given N, the collection of N initial states is the same at different μ for consistent comparison. The average residual error of the last 20 iterations is reported in the table. For each setting, the best residual is highlighted in bold, s and the worst residual is emphasized in italic. From the table, we can see that $\mu = 1$ performs the worst in all cases. In fact, a huge improvement can be observed in the residual and successful roll-up number when the gradient information is used. Also, this table confirms that with the number of characteristics increasing, the final accuracy of the numerical value functions always gets better since more labeled data are provided.

To investigate the effect of μ on the decay of the error, we plot the residual error during the policy iteration in Figure 1. This figure clearly demonstrates that $\mu = 1$ has the slowest convergence among all μ we tested, and we can find that adding even a small portion of the loss for the value-gradient, i.e., $\mu < 1$, can improve the convergence. Also, we plot the successful roll-up number of $\mu = 1$ compared with $\mu = 0.8$ at each iteration. It can be seen that the value-gradient significantly improves the performance.

TABLE 1

The error (HJE residual) and the number of successful roll-ups for different μ when the trajectory number N changes in the cart-pole balancing task. The training step is 100.

		Number of trajectories		
	μ	2	5	10
Residual	1.0	*2.088*	*0.844*	*0.934*
	0.8	0.696	0.281	0.100
	0.6	0.450	0.169	0.117
	0.4	0.441	0.147	0.091
	0.2	0.181	**0.113**	**0.082**
	0.0	**0.166**	0.124	0.094
Successful roll-up	1.0	14.85	*19.25*	*12.30*
	0.8	10.65	25.10	**60.8**
	0.6	**27.10**	25.15	38.65
	0.4	11.05	39.25	44.20
	0.2	*9.30*	40.95	50.45
	0.0	20.95	**44.95**	55.00

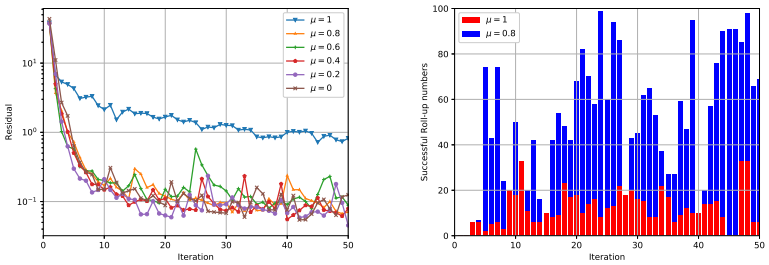

FIG. 1. *Residual and the successful roll-ups with trajectory number of* 10 *and training step of* 100 *in the cart-pole task.*

Experiment 2. The purpose of this experiment is to test the performance of the methods when the training process is not sufficiently long. In this experiment, the training steps 50, 100, 150, and 200 are tested. A small training step means less accuracy in fitting the value function. The trajectory number is now fixed as 10.

As shown in Table 2, the accuracy improves quite remarkably as long as the value-gradient is included in the formulation. The successful roll-ups also show a better performance for $\mu < 1$, particularly when the number of trajectories increases.

Figure 2 shows the residual and successful roll-ups with respect to the policy iteration for different μ values. As expected, choosing $\mu < 1$ considerably improves these results.

To conclude the above experiment, we have performed the numerical tests by changing the amount of characteristics data and the training steps, which are two important factors in practical computation. By comparing the performance measured by the HJE residual as the error and the successful roll-ups as the robustness, we find that these numerical results consistently show high performance when using the characteristics data from both the value and value-gradient functions. Although the four tested values of $\mu = 0.2, 0.4, 0.6, 0.8$ between 0 and 1 always beat the traditional method at $\mu = 1$, the optimal value μ actually varies on the specific settings, and the difference among these four values for the performance is marginal.

TABLE 2

The error (HJE residual) and the number of successful roll-ups for different μ when training steps change in the cart-pole balancing task. The trajectory number is 10.

		Training step			
	μ	50	100	150	200
Residual	1.0	*1.355*	*0.934*	*0.500*	*0.471*
	0.8	0.260	0.100	0.151	0.155
	0.6	0.175	0.117	**0.092**	0.097
	0.4	**0.096**	0.091	0.105	0.103
	0.2	0.106	**0.082**	0.094	**0.080**
	0.0	0.130	0.094	0.070	0.083
Successful roll-up	1.0	*13.75*	*12.30*	28.55	*30.00*
	0.8	58.25	**60.80**	41.45	35.80
	0.6	28.05	38.65	*16.50*	35.55
	0.4	56.40	44.20	36.50	35.75
	0.2	**61.65**	50.45	35.30	47.00
	0.0	21.85	55.00	**49.30**	**54.10**

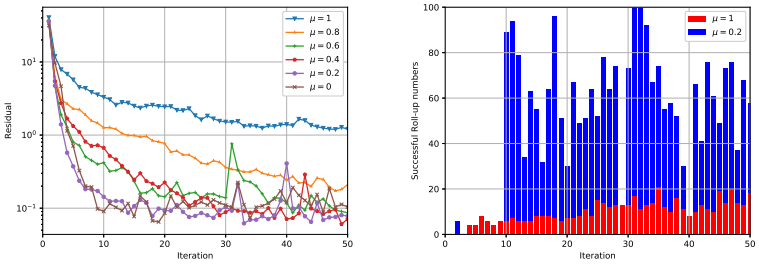

FIG. 2. *Residual and the successful roll-ups with trajectory number of* 10 *and training step of* 50 *in the cart-pole task.*

6. Conclusion. Based on the system of PDEs for the value-gradient functions we derived in this paper, we develop a new policy iteration framework, called **PI-lambda**, for the numerical solution of the value function for optimal control problems. We show the convergence property of this iterative scheme under Assumption 2.1. The system of PDEs for the value-gradient functions $\lambda(x)$ is closed since it does not involve the value function $\Phi(x)$ at all, so one could, in principle, use neural networks only for λ. This is distinct from many existing methods based on the value function (e.g., [27]). The system for λ is also essentially decoupled and shares the same characteristics ODE with the generalized HJE. By simulating characteristics curves in parallel for the state variable by any classic ODE solver (such as the Runge–Kutta method), both the value Φ and the value-gradient functions λ on each characteristics curve can be computed. Equipped with any state-of-the-art function representation technique and large-scale minimization techniques from supervised learning, these labeled data can be generalized to the whole space to deal with high-dimensional problems. Policy iteration has the computational convenience of simulating characteristics equations only forward in time instead of solving any boundary-value problem for *optimal* trajectories directly as in [32, 35, 43]. Policy iteration is also convenient when Hamiltonian minimization has no analytical expression. The learning procedure of supervised learning in our method is not new, and it has been applied, for example, in [51, 32, 42], to combine the losses from the policy data, the value function data, and the value-gradient data. Our distinction from these works is to formulate the costate variable

as the gradient function $\lambda(x)$ of the state and not as a function of the time $\lambda(t)$ in the PMP.

The generalization to the finite horizon control problem on $[0,T]$ is straightforward: to replace $\rho\lambda(x)$ by $-\partial_t\lambda(t,x)$ in (3.7) and add the transversality condition $\lambda(T,x)=\nabla_x h(T,x)$ when there is a terminal cost $h(T,x(T))$. The main algorithm in this paper based on the policy iteration, **PI-lambda** (Algorithm 3.1), is still applicable in this case, and our main theorem (Theorem 3.5) can be easily generalized.

Some practical computational issues which are not fully discussed here include the choice of initial policy $a^{(0)}$, and the number of trajectories N and their initial locations $\{X_0\}$. For the initial policy, it should be chosen conservatively to stabilize the dynamics. For the characteristics curves, N may be changed from iteration to iteration, and adaptive sampling for the initial states is a good issue for further exploration [42].

An obvious question to address in the future is how to formulate the equations of λ for stochastic optimal control so as to leverage benefits similar to our method. One may consider the splitting method in [9].

Acknowledgments. We thank Dr. Bohan Li and Dr. Yiqun Li for offering academic suggestions on some results and their proofs.

REFERENCES

[1] A. Alla, M. Falcone, and D. Kalise, *An efficient policy iteration algorithm for dynamic programming equations*, SIAM J. Sci. Comput., 37 (2015), pp. A181–A200, https://doi.org/10.1137/130932284.

[2] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep Neural Networks Algorithms for Stochastic Control Problems on Finite Horizon: Numerical Applications*, Methodol. Comput. Appl. Probab., 24 (2022), pp. 143–178.

[3] S. Barto, *Neuronlike adaptive elements that can solve difficult learning control problems*, IEEE Trans. Syst. Man Cybern., 13 (1983), pp. 834–846.

[4] R. W. Bea, *Successive Galerkin approximation algorithms for nonlinear optimal and robust control*, Int. J. Control, 71 (1998), pp. 717–743.

[5] R. W. Beard, G. N. Saridis, and J. T. Wen, *Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation*, Automatica, 33 (1997), pp. 2159–2177.

[6] R. W. Beard, G. N. Saridis, and J. T. Wen, *Approximate solutions to the time-invariant Hamilton-Jacobi–Bellman equation*, J. Optim. Theory Appl., 96 (1998), pp. 589–626.

[7] R. Bellman, *A Markovian decision process*, Indiana Univ. Math. J., 6 (1957), pp. 679–684, https://doi.org/10.1512/iumj.1957.6.56038.

[8] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[9] A. Bensoussan, *Splitting up method in the context of stochastic PDE*, in Stochastic Partial Differential Equations and Their Applications, B. L. Rozovskii and R. B. Sowers, eds., Springer, Berlin, Heidelberg, 1992, pp. 22–31.

[10] A. Bensoussan, *Estimation and Control of Dynamical Systems*, Interdiscip. Appl. Math., Springer, Cham, 2018.

[11] A. Bensoussan, Y. Li, D. P. C. Nguyen, M.-B. Tran, S. C. P. Yam, and X. Zhou, *Machine learning and control theory*, in Handbook of Numerical Analysis, E. Trélat and E. Zuazua, eds., Elsevier B.V., 2022, pp. 531–558.

[12] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. I, 2nd ed., Athena Scientific, Belmont, MA, 2001.

[13] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*, Athena Scientific, Belmont, MA, 2019.

[14] P. Chen, J. Darbon, and T. Meng, *Hopf-Type Representation Formulas and Efficient Algorithms for Certain High-Dimensional Optimal Control Problems*, preprint, https://arxiv.org/abs/2110.02541, 2021.

[15] P. Chen, J. Darbon, and T. Meng, *Lax-Oleinik-Type Formulas and Efficient Algorithms for Certain High-Dimensional Optimal Control Problems*, http://arxiv.org/abs/2109.14849, 2021.

[16] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, *Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton-Jacobi equations arising from optimal control and differential games problems*, J. Sci. Comput., 73 (2017), pp. 617–643, https://doi.org/10.1007/s10915-017-0436-5.

[17] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, *Algorithm for overcoming the curse of dimensionality for certain non-convex Hamilton-Jacobi equations, projections and differential games*, Ann. Math. Sci. Appl., 3 (2018), pp. 369–403.

[18] Y. T. Chow, W. Li, S. Osher, and W. Yin, *Algorithm for Hamilton-Jacobi equations in density space via a generalized Hopf formula*, J. Sci. Comput., 80 (2019), pp. 1195–1239.

[19] J. Darbon, P. M. Dower, and T. Meng, *Neural Network Architectures Using Min Plus Algebra for Solving Certain High Dimensional Optimal Control Problems and Hamilton-Jacobi PDEs*, preprint, http://arxiv.org/abs/2105.03336, 2021.

[20] J. Darbon, G. P. Langlois, and T. Meng, *Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures*, Res. Math. Sci., 7 (2020), 20, https://doi.org/10.1007/s40687-020-00215-6.

[21] J. Darbon and T. Meng, *On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton–Jacobi partial differential equations*, J. Comput. Phys., 425 (2021), 109907, https://doi.org/10.1016/j.jcp.2020.109907.

[22] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere*, Res. Math. Sci., 3 (2016), pp. 1–26.

[23] W. E., J. Han, and A. Jentzen, *Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning*, Nonlinearity, 35 (2022), pp. 278–310.

[24] M. Falcone and R. Ferretti, *Semi-Lagrangian Approximation Schemes for Linear and Hamilton–Jacobi Equations*, SIAM, 2013, https://doi.org/10.1137/1.9781611973051.

[25] W. Fleming and H. Soner, *Controlled Markov Processes and Viscosity Solutions*, 2nd ed., Stoch. Model. Appl. Probab. 25, Springer, New York, 2006.

[26] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*, Appl. Math. 1, Springer-Verlag, New York, 1975.

[27] J. Han, A. Jentzen, and E. W., *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci. USA, 115 (2018), pp. 8505–8510.

[28] M. B. Horowitz, A. Damle, and J. W. Burdick, *Linear Hamilton-Jacobi-Bellman equations in high dimensions*, in Proceedings of the 53rd IEEE Conference on Decision and Control, 2014, pp. 5880–5887.

[29] R. A. Howard, *Dynamic Programming and Markov Processes*, Technology Press of M.I.T., Cambridge, Mass.; John Wiley & Sons, New York-London, 1960.

[30] T. P. Lillicrap. J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous Control with Deep Reinforcement Learning*, preprint, https://arxiv.org/abs/1509.02971, 2016.

[31] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon: Convergence analysis*, SIAM J. Numer. Anal., 59 (2021), pp. 525–557, https://doi.org/10.1137/20M1316640.

[32] D. Izzo, E. Öztürk, and M. Märtens, *Interplanetary transfers via deep representations of the optimal policy and/or of the value function*, in Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19), ACM, ACM, New York, 2019, pp. 1971–1979, https://doi.org/10.1145/3319619.3326834.

[33] D. Kalise and K. Kunisch, *Polynomial approximation of high-dimensional Hamilton–Jacobi–Bellman equations and applications to feedback control of semilinear parabolic PDES*, SIAM J. Sci. Comput., 40 (2018), pp. A629–A652, https://doi.org/10.1137/17M1116635.

[34] W. Kang and L. Wilcox, *A causality free computational method for HJB equations with application to rigid body satellites*, in Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2015, p. 2009.

[35] W. Kang and L. C. Wilcox, *Mitigating the curse of dimensionality: Sparse grid characteristics method for optimal feedback control and HJB equations*, Comput. Optim. Appl., 68 (2017), pp. 289–315, https://doi.org/10.1007/s10589-017-9910-0.

[36] J. Kim, *Hamilton-Jacobi-Bellman equations for Q-learning in continuous time*, 120 (2020), pp. 1–10, https://proceedings.mlr.press/v120/kim20b.html.

[37] J. Kim, J. Shin, and I. Yang, *Hamilton-Jacobi deep Q-learning for deterministic continuous-time systems with Lipschitz continuous controls*, J. Mach. Learn. Res., 22 (2021), 206, https://dl.acm.org/doi/abs/10.5555/3546258.3546464.

[38] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, preprint, https://arxiv.org/abs/1412.6980, 2014.

[39] G. P. Kontoudis and K. G. Vamvoudakis, *Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework,* Neural Netw. Learn. Syst., 30 (2019), pp. 3808–3817.

[40] E. C. Lawrence, *Partial Differential Equations*, 2nd ed., American Mathematical Society, 2010.

[41] A. T. Lin, Y. T. Chow, and S. J. Osher, *A splitting method for overcoming the curse of dimensionality in Hamilton-Jacobi equations arising from nonlinear optimal control and differential games with applications to trajectory generation*, Commun. Math. Sci., 16 (2018), pp. 1933–1973, https://doi.org/10.4310/cms.2018.v16.n7.a9.

[42] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations*, SIAM J. Sci. Comput., 43 (2021), pp. A1221–A1247, https://doi.org/10.1137/19M1288802.

[43] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *Qrnet: Optimal regulator design with LQR-augmented neural networks*, IEEE Control Syst. Lett., 5 (2021), pp. 1303–1308.

[44] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[45] M. Oster, L. Sallandt, and R. Schneider, *Approximating the Stationary Hamilton-Jacobi-Bellman Equation by Hierarchical Tensor Products*, preprint, https://arxiv.org/abs/1911.00279, 2019.

[46] M. Palanisamy, H. Modares, F. L. Lewis, and M. Aurangzeb, *Continuous-time Q-learning for infinite-horizon discounted cost linear quadratic regulator problems*, IEEE Trans. Cybernet. 45 (2015), pp. 165–176.

[47] L. S. Pontryagin, *Mathematical Theory of Optimal Processes*, CRC Press, Boca Raton, FL, 1987.

[48] M. L. Puterman and S. L. Brumelle, *On the convergence of policy iteration in stationary dynamic programming*, Math. Oper. Res., 4 (1979), pp. 60–69.

[49] B. Recht, *A tour of reinforcement learning: The view from continuous control*, Ann. Rev. Control Robot. Auton. Syst., 2 (2019), pp. 253–279.

[50] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Adaptive Comput. Mach. Learn., MIT Press, Cambridge, MA, 2018.

[51] D. Tailor and D. Izzo, *Learning the optimal state-feedback via supervised imitation learning*, Astrodynamics, 3 (2019), pp. 361–374, https://doi.org/10.1007/s42064-019-0054-0.

[52] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694, https://doi.org/10.1137/S0036142901396533.

[53] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, in Proceedings of the 1994 33rd IEEE Conference on Decision and Control, vol. 2, 1994, pp. 1368–1373.