Revisiting Continuous-Time Reinforcement Learning. A Study of HJB Solvers Based on PINNs and FEMs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Despite recent advances in Reinforcement Learning (RL), the Markov Decision Processes (MDPs) are not always the best choice to model complex dynamical systems requiring interactions at high frequency. Being able to work with arbitrary time intervals, Continuous Time Reinforcement Learning (CTRL) is more suitable for those problems. Instead of the Bellman equation operating in discrete time, it is the Hamiltonian Jacobi Bellman (HJB) equation that describes value function evolution in CTRL. Even though the value function is a solution of the HJB equation, it may not be its unique solution. To distinguish the value function from other solutions, it is important to look for the viscosity solutions of the HJB equation. The viscosity solutions constitute a special class of solutions that possess uniqueness and stability properties. In this paper, we bring together the formalism of viscosity solutions and practical methods for finding them. We also propose a novel way of training neural networks to obtain viscosity solutions. Finally, we do a comparison of those methods with discrete time RL (DTRL) algorithms to emphasize the benefits of considering the continuous time setting. This paper aims at providing the necessary theoretical basis for working with CTRL and setting a few possible directions for future research.

1 Introduction

Reinforcement learning (RL) is getting more and more attention. Most of state-of-the-art methods are designed to work with Markov Decision Processes and, in particular, rely on a discrete time assumption. Even though this assumption is not that restrictive in tasks such as games, it is no longer valid in complex problems such as driving cars or finance trading. Such problems are usualy described by a dynamical system where DTRL often struggles to provide accurate control within short time intervals due to several reasons [8, 31, 23]. As DTRL algorithms, such as Q-learning or policy gradients, typically operate in discrete time steps, it becomes challenging to capture the nuances of continuous control within small intervals. Even when the discretization step in time is set to be small, DTRL fails to learn the optimal policy as exploration cannot be done efficiently.

Continuous Time Reinforcement Learning (CTRL) derives from the optimal control theory and thus provides a promising theoretical framework to tackle the shortcomings of DTRL. It allows us to have control over the discretization of time and it provides smoother control. The key ingredient in CTRL is the Hamilton-Jacobi-Bellman (HJB) equation [8, 25], which is a Partial Differential Equation (PDE). This PDE describes the evolution of the value function with respect to a state of the system, which in turn evolves continuously with time. Solving this equation would lead to an optimal control policy and its associated value function. However, this remains a challenging task. In [25], it was shown that the HJB equation may have multiple solutions in a class of continuous functions.

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

Distinguishing the value function from the others solution lies in the search of viscosity solutions that correspond to good solutions of PDEs that can satisfy uniqueness and stability properties.

However, in general, there is no guarantee that the solution found by existing methods is the good solution, *i.e.* the viscosity solution. Moreover, the existing methods for solving the HJB equation are limited and mainly applicable to specific cases such as Linear Quadratic Regulator (LQR) problems or variational problems [9]. There are some approaches that can be applied in general case, such as Finite Difference (FD), Finite Element Methods (FEM) [10], and dynamic programming methods [25] that transform the problem back to DTRL due to discretization. However, their effectiveness is hindered by the exponentially growing complexity with respect to the state space dimensionality, and moreover it is also affected by discretization error.

Conversely, neural networks are emerging as PDE solvers that can cope with the curse of dimensionality [27]. Although Neural Networks (NNs) have shown a great potential in various domains [19, 15], applying them to solve the HJB equation requires caution because of non-uniqueness problem. To address these challenges, a promising direction is to leverage the formalism of viscosity solutions in conjunction with neural networks.

In this paper, we revisit CTRL approaches and we analyse how the latest advances in deep learning can be applied to solve the HJB equation. We focus on the case of deterministic environments with known dynamics. The extension to stochastic environments and unknown dynamics is possible (see [31, 34]), but left for the future work. In particular, we consider how NN-based methods can be combined to find viscosity solutions, offering a novel training scheme. We start with the analysis of the existing literature in Section 2. Then, we carefully introduce the definitions and notations related to CTRL in Section 3.1, describe the HJB equation in Section 3.2, define viscosity solutions in Section 3.3 and discuss boundary conditions in Section 3.4. Then, different ways of integrating neural networks into the process of solving the HJB equation are discussed in Section 4. Finally, we demonstrate performances of some approaches from Section 4 on the inverted pendulum, a classical use case from RL, in Section 5.

2 Related Works

One of the first papers to study CTRL were [8, 25, 5]. Those works introduced the HJB equation as a key equation for finding the optimal policy. In [8], different discretization schemes and algorithms are analysed for computing the value function, including continuous TD(λ) and continuous Actor-Critic. The work of [5] did the study of applicability of methods from [8] to a large class of control problems. The work of [25] tackled CTRL by the study of viscosity solutions and their properties. There, it was demonstrated the challenges of solving the HJB equation such as the non-uniqueness of solutions and inequality boundary conditions. In addition to that, convergent numerical schemes based on dynamic programming were derived to approximate the value function. One of the problems of numerical schemes is the curse of dimensionality. In order to mitigate this problem, sparse grid [14] can be used instead of naive uniform grid. In this article, we take the formalism proposed in [25], but we consider neural network based approaches to find viscosity solutions (while [25] consider tabular algorithms).

In contrast, neural networks can also be applied for solving the HJB equation [24, 18, 4, 29, 20, 12, 1]. It was first demonstrated in [24]. In [29], the training is regularized to avoid finding "bad" solutions. Moreover, the same work raised the problem of falling in a local minimum of the squared HJB residual, and some solutions to avoid it were proposed. Compared to previous papers, [18, 1] have emphasized the robustness of the resulting controller and the stability of the proposed algorithm [18]. Adapting the former methods for more practical cases such as the inverted pendulum and the cartpole was done in [20]. We consider similar approaches as [29, 20], but unlike them we use the formalism proposed in [25] to develop an approach that can converge to viscosity solutions.

Several extensions of the classical HJB equation were also introduced such as HJB for an explorative reward function [31], the soft HJB equation with maximum entropy regularization [17, 11] and the distributional HJB equation [32]. Those works extend the existing theory to other definitions of the value function, however experiments are conducted on a limited set of simple problems.Futhermore, it is also possible to extend the HJB equation to continuous-time partially observable markov decision processes(CTPOMDPs). In [2], they proposed a formalism to describ CTPOMDPs, including the CTPOMDP HJB equation. In addtion, some experiments are shown. The work of [16] tackles the problem of CTRL by adapting the well-known DQN algorithm to this framework. A definition for

the Q-function in the continuous case is given and the "HJB equation for the Q-function" is derived, which results in a DQN-like algorithm for the semi-discrete time setting. However, this approach is limited to lipschitz continuous control. The HJB equation was used to improve DTRL algorithms like PPO in [23] and it resulted in a significant improvement on MuJoCo, proving that HJB loss is better adapted for learning value functions of dynamical systems.

There were some attempts to propose alternative ways for solving the HJB equation, among which [26, 7]. In [26], there is another NN approach to approximate the value function that in contrast to previous NN approach is using Pontryagin's maximum principle. Conversely, [7] considers some special neural network architectures that can work with min-plus algebra, though this approach is suitable only for some optimal control problems.

Another line of research exploits the continuity to do more accurate Monte Carlo estimations of value function [22, 21, 34] rather than using the HJB equation. The authors of [22] adapted the fitted value iteration algorithm to the continuous-time reinforcement learning framework. In [34], a model-based algorithm was introduced, which aims at solving the problem in the actor-critic manner. Bayesian Neural ODEs are used in order to learn the dynamic of the system.

3 Hamiltonian-Jacobi-Bellman Equation

3.1 Formalism for Reinforcement Learning in the Continuous Case

The main difference between continuous time case and discrete time case is that now transitions and rewards depend on time t, which is a continuous variable. Furthermore, we consider the problem of finding optimal trajectories in the continuous state space. We denote with $O \subseteq \mathbb{R}^d$, the open set of admissible states, and thus the admissible control $u \in U$ is the one that keeps the state trajectory inside the domain O. In this case, $U \subset \mathbb{R}^m$ corresponds to the action/control space. We assume that U is bounded. If there is a time when the state trajectory reaches the boundary ∂O and then goes beyond it, then it means that the system has exited the control domain, which corresponds to reaching the terminal state in the discrete time case. The exit time is denoted τ, with $0 < \tau \leq \infty$ (the case $\tau = \infty$ is also possible in the situation when exiting $\bar{O}$ is not optimal, see Cases 2 and 3 in Section 3.4).

More formally, in the continuous case the states do not form a sequence $(X_t)_t$ but a trajectory $x : \mathbb{R}_+ \to \bar{O} \subset \mathbb{R}^d$. Similarly, the actions are defined for any $t < \tau$: $u : \mathbb{R}_+ \to U$. In particular, we want to find an optimal policy function $\pi : \bar{O} \to U$, such that $u^*(t) = \pi(x^*(t))$ for any t, where $u^*(t)$ and $x^*(t)$ are control and state of the optimal trajectory respectively. The dynamics of the environment is defined through the following ordinary differential equation (ODE):

$$\frac{dx(t)}{dt} = f(x(t), u(t)) \tag{1}$$

The function $f : \bar{O} \times U \to \mathbb{R}^d$ is called the state dynamics. While the state x is inside the control domain O (*i.e.* $t < \tau$), the reward $r : \bar{O} \times U \to \mathbb{R}$ is received, conversely once the system exits the control domain, then the exit reward $R : \partial O \to \mathbb{R}$ is obtained. Let us note that $R(x)$ is well defined only when $O \subset \mathbb{R}^d$, *i.e.* $O \neq \mathbb{R}^d$. Thus, our goal is to find an optimal control $u^*(t)$ for $0 \leq t \leq \tau$ that maximizes the following quantity:

$$J(x_0; u(t)) = \int_0^\tau \gamma^t r(x(t), u(t))dt + \gamma^\tau R(x(\tau)) \tag{2}$$

for any initial state x_0. J is called the reinforcement functional and $\gamma \in [0, 1)$ is the discount factor. In the continuous case, the value function is defined as:

$$V(x) = \sup_{u(t)} J(x; u(t)) \tag{3}$$

Here, the value function is defined as the maximum value that J can take when we start at x at time $t = 0$. Furthermore, we introduce additional assumptions on functions $f(x, u)$ and $r(x, u)$.

Assumption 3.1 *(i) U is bounded,*

(ii) f, r are bounded, f is continuous on $\mathbb{R}^d \times U$ and r is uniform continuous on $\mathbb{R}^d \times U$,

(iii) there exists L_f, such that $\|f(x,u) - f(y,u)\| \leq L_f \|x - y\|$ for any $x, y \in \mathbb{R}^d$.

The next assumption is necessary for the "reachability" of the boundary, *i.e.* the state of the system can reach any boundary point from its vicinity and vice versa. This assumption is necessary for value function to be continuous.

Assumption 3.2 *(i) For any $x \in \partial O$, $\exists v(x) \in U$ with $f(x, v(x))^T \eta(x) < 0$;*

(ii) For any $x \in \partial O$, $\exists v(x) \in U$ with $f(x, v(x))^T \eta(x) > 0$;

Assumption 3.3 (Regularity condition) *There exist $\epsilon_0, r > 0$ and $\hat{\eta}(x)$, a bounded and uniform continuous map of $\bar{O}$, satisfying*

$$B(x + \epsilon\hat{\eta}(x), r\epsilon) \subset O, \quad \forall x \in O, \epsilon \in (0, \epsilon_0] \tag{4}$$

with $B(x, r) = \{y \in \mathbb{R}^d : \|x - y\| < r\}$.

3.2 Hamilton-Jacobi-Bellman Equation

We can prove that the value function defined in Eq. (3) satisfies the following result:

Theorem 3.1 *(Hamilton-Jacobi-Bellman). If the value function V is differentiable at x, then the Hamilton-Jacobi-Bellman (HJB) equation holds at any $x \in O$:*

$$V(x)\ln(\gamma) + \sup_{u \in U} \left\{ \nabla_x V(x)^T f(x,u) + r(x,u) \right\} = 0. \tag{5}$$

We want to compute the value function to find the optimal control. Indeed, from the value function we can define a function called the feed-back control policy $\pi : \bar{O} \to U$ such that $\pi(x(t)) = u^*(t)$ by setting:

$$\pi(x) \in \operatorname*{argsup}_{u \in U} \left\{ \nabla_x V(x)^T f(x,u) + r(x,u) \right\} \tag{6}$$

Therefore, it is crucial to find an efficient way to compute V to get the optimal control u^*. However, as it was mentioned in [25], solving the HJB equation has several challenges. First, the value function V is often non-smooth, and only continuous with respect to states. Second, the HJB equation may have multiple generalized continuous solutions. Third, it is common that the HJB equation (5) should be solved under inequality boundary conditions. We address those points in the following sections.

3.3 Viscosity

To address the problem of non-uniqueness of HJB solution, we need to introduce the viscosity property. Let $H(x, W, \nabla_x W) = -W(x)\ln\gamma - \sup_{u \in \mathcal{U}}[\nabla_x W(x)^T f(x,u) + r(x,u)]$, then the HJB equation can be expressed as

$$H(x, W, \nabla_x W) = 0. \tag{7}$$

Using these notations, let us define what a viscosity solution is.

Definition 3.1 (Viscosity solution)

- *$W \in C(O)$ is a viscosity subsolution of the HJB equation in O if $\forall \phi \in C^1(O)$ and $\forall x \in O$ local maximum of $W - \phi$ such that $W(x) = \phi(x)$, we have:*

$$H(x, \phi(x), D\phi(x)) \leq 0$$

- *$W \in C(O)$ is a viscosity supersolution of the HJB equation in O if $\forall \phi \in C^1(O)$ and $\forall x \in O$ local minimum of $W - \phi$ such that $W(x) = \phi(x)$, we have:*

$$H(x, \phi(x), D\phi(x)) \geq 0$$

- *If $W \in C(O)$ is a viscosity subsolution and a supersolution then it is a viscosity solution.*

Viscosity is a very important property that allows us to separate "good" solutions from "bad" ones. Viscosity solutions were first introduced in [6] and they are proven be unique for multiple types of PDEs. The HJB equation is not an exception. Next, we state a few useful results that were proven in [9].

Theorem 3.2 *Given Assumption 3.1, the value function V is uniformly continuous and bounded in $\mathbb{R}$ and thus it is a unique viscosity solution of the HJB equation* (7).

This theorem is given for the case when $O = \mathbb{R}^d$ and therefore no boundary conditions take place. The other cases are discussed in the next sections.

The proof of Theorem 3.2 consists of several parts. First, one can prove that under Assumption 3.1, the value function is indeed uniform continuous and bounded. Then, one can show that it is a viscosity solution due to the dynamic programming principle that holds in continuous time case as well. The uniqueness comes from the comparison principle that states that under Assumptions 3.1 and provided that W and V are bounded and uniformly continuous functions, if W and V are viscosity subsolution and supersolution respectively, then $W \leq V$. Thus, if there exists at least one viscosity solution that is bounded and uniform continuous then it is unique, therefore V is unique viscosity solution of the HJB equation in $\mathbb{R}^d$.

3.4 Discussion of Boundary Conditions

Now let us consider the situation of $O \subset \mathbb{R}^d$, then it becomes necessary to take into account boundary conditions. To guarantee the existence of a continuous value function that satisfies the boundary conditions, it is important to impose Assumption 3.2 that verifies that the boundary is reachable. We discuss three different cases.

Case 1 If the system exits at any boundary point $x \in \partial O$ once the boundary is reached, *e.g.* it is the case when the exit reward $R(x)$ is sufficiently high to avoid leaving the area, in particular when $R(x') = R(x) \geq r$ for any $x, x' \in \partial O$ and $r \geq r(\bar{x}, u(\bar{x}))$ for any $\bar{x}, u(\bar{x})$,

$$V(x) - R(x) = 0 \quad \forall x \in \partial O. \tag{8}$$

The uniqueness result holds due to the comparison principle that states that under Assumptions 3.1 and provided that W and V are bounded and uniform continuous functions, if W and V are viscosity subsolution and supersolution respectively, then $\sup_{x \in \bar{O}}(W(x) - V(x)) \leq \sup_{x \in \partial O}(W(x) - V(x))$. As mentioned previously, the existence is assured with Assumption 3.2-(ii).

Case 2 If the system never exits the control domain. Let us denote the external normal vector at point $x \in \partial O$ as $\eta(x)$, then this boundary can be expressed as $f(x, u^*(x))^T \eta(x) \leq 0$ for any $x \in \partial O$. The optimal policy is not known a priori and thus it is hard to verify this constraint. In [9, 28], it was shown that it can be reformulated as:

$$-H(x, W, \nabla_x W + \alpha \eta(x)) \leq 0 \quad \forall \alpha \leq 0, x \in \partial O. \tag{9}$$

This allows to extend Definition 3.1.

Definition 3.2 (Constrained viscosity solution) *$W \in C(\bar{O})$ is called a constrained viscosity solution of the HJB equation* (5) *if it is a viscosity subsolution in O and a viscosity supersolution in $\bar{O}$, i.e. if $\forall \phi \in C^1(\bar{O})$ and $\forall x \in \bar{O} \cup \arg\min\{(W - \phi)(x) : x \in \bar{O}\}$ with $W(x) = \phi(x)$, we have:*

$$H(x, \phi(x), \nabla_x \phi(x)) \geq 0.$$

It is also possible to prove that there exists a continuous value function provided that Assumption 3.2(i) holds and the set of admissible actions is not empty for any state of the system. Under additional Assumption 3.3, there exists a unique constrained viscosity solution (see [28]).

Case 3 If there exists a subset of points of the boundary at which the system exits the control domain. This implies the boundary:

$$R(x) - V(x) \leq 0 \quad \forall x \in \partial O. \tag{10}$$

However, this boundary is not sufficient to have uniqueness, therefore we redefine viscosity for this inequality constraint (10).

Definition 3.3 (Viscosity solution with the boundary condition (10)**)**

- *$W \in C(\bar{O})$ is a viscosity subsolution of the HJB equation in O with the boundary condition* (10) *if it is a viscosity subsolution in O and $\forall \phi \in C^1(\bar{O})$ and $\forall x \in \partial O$ local maximum of $W - \phi$ such that $W(x) = \phi(x)$, we have:*

$$\min\{H(x, \phi(x), \nabla_x \phi(x)), R(x) - W(x)\} \leq 0$$

- *$W \in C(\bar{O})$ is a viscous supersolution of the HJB equation in O with the boundary condition* (10) *if it is a viscosity supersolution and $\forall \phi \in C^1(O)$ and $\forall x \in \partial O$ local minimum of $W - \phi$ such that $W(x) = \phi(x)$, we have:*

$$\max\{H(x, \phi(x), D\phi(x)), R(x) - W(x)\} \geq 0$$

- *If $W \in C(\bar{O})$ is a viscous subsolution and a supersolution with the boundary condition* (10) *then it is a viscous solution with the boundary condition* (10).

It is easy to check that when Eq. (10) is verified then a viscosity subsolution $W(x)$ in O is a viscosity subsolution with the boundary condition (10). However, when $W(x) > R(x)$ for some point $x \in \partial O$ then definition 3.3 imposes an additional constraint that $W(x)$ should be a viscosity supersolution at such boundary points. Thus similarly to Case 2 boundary condition (9) should be also satisfied, which can be interpreted as the system not being able to exit at those points. Similarly to Case 2, there is a uniqueness result:

Theorem 3.3 *Let us assume that Assumptions 3.1-3.3 hold, then the value function V is in $C(\bar{O})$ and it is the unique viscous solution of the HJB equation in O with the boundary condition* (10).

The proof of this theorem can be found in [9, 3]. Some of the assumptions can be relaxed and it is possible to obtain more general uniqueness results (see [9, 3, 13]). However, the assumptions mentioned earlier are verified for the large class of control problems that appear in practice, like classical control or MuJoCo [30].

4 Possible Approaches

In this section we discuss several ways of solving the HJB equation (5). In what follows, we assume that the dynamics of the state determined by $f(x, u)$ is known and the control space is discrete, the case of unknown $f(x, u)$ and the continuous control space is left for future work. In regard to boundary conditions, we illustrate further methods for solving HJB under boundary conditions from Case 2 (see Eq. (9)), the extension to other cases is straightforward.

4.1 Dynamic Programming

Several solvers exist such as Finite Difference method (FD) or Finite Element Method (FEM). These methods require the discretization of the domain (a grid for FD or a triangulation for FEM). Moreover, the dynamics of the system has to be available to compute the solution of the PDE.

The work of [25] establishes the connection between solving the HJB equation and the classical reinforcement learning framework by deriving a Markov Decision Process (MDP) from the discretization of the HJB equation, using either FD or FEM schemes.

We show the perfomance of this approach in Section 5. However, despite many efforts, we were not able to make the algorithm based on FD work in our experiments. Therefore, we only present the results of dynamic programming based on FEM. This dynamic programming is presented in Appendix A.1.1.

Despite convergence guarantees provided in [25], the main problem of these methods is that they are mesh-dependent. Thus, they suffer from the curse of dimensionality, *i.e.* the computantional cost increases exponentially with the dimension of the problem. For example, if the state space of the environment is a subdomain of $\mathbb{R}^4$ then a naive approach that divides all axes uniformly in n parts results in n^4 states to handle. If $n = 32$, which may not be sufficient to solve the problem, it leads to 2^{20} states for the algorithm to process. In our experiments performed on a single CPU, we are not able to go beyond 20 points per axis for $O \subseteq \mathbb{R}^4$. There exist smarter ways of defining the grid such as a sparse grid [14], which may help to use the available compute and memory budget more efficiently, but we leave sparse grids usage out of scope of this paper.

4.2 Approach Based PINNs

Let $W(\cdot,\theta)$ is a neural network that computes the solution of Eq. (5) with θ being its parameters. In this section, we describe how to train $W(\cdot,\theta)$ in PINNs-like manner. The idea of PINNs was proposed in [27] and applied to some simple PDEs like one-dimensional nonlinear Schrödinger equation, but not for optimal control (see Appendix A.1.2 for a quick introduction to PINNs). To apply it to the HJB equation, it is necessary to search among viscosity solutions to avoid non-uniqueness issues (see Section 3). Checking the conditions of Definition 3.1 is not feasible in practice. Instead, we use the next property of viscosity solutions.

Lemma 4.1 (Stability) *Let W^ϵ be a viscosity subsolution (or a super solution) of*

$$W^\epsilon(x) + F^\epsilon(x, W^\epsilon(x), \nabla_x W^\epsilon(x), \nabla_x^2 W^\epsilon(x)) = 0$$

in O. Suppose that F^ϵ converges to F uniformly on every compact subset of its domain, and W^ϵ converges to W uniformly on compact subsets of $\bar{O}$. Then W is a viscosity subsolution (or a supersolution, respectively) of the limiting equation.

This lemma is proven in [9]. In our case, we are interested in equation:

$$H(x, W^\epsilon(x), \nabla_x W^\epsilon(x)) = \epsilon \nabla_x^2 W^\epsilon(x), \tag{11}$$

where the left hand side is the same as in Eq. (5), while the right hand side depends linearly on $\epsilon > 0$. In [9], it is shown that Eq. (11) has a unique smooth solution $W^\epsilon(x)$, *i.e.* it admits a classical solution, which is a viscosity solution at the same time. Therefore, if $W^\epsilon(x)$ converges uniformly to $W(x)$ then $W(x)$ is a viscosity solution of the original HJB equation (5).

To ensure the uniform convergence of Eq. (11) to Eq. (5), we need to define a sequence of $\{\epsilon_n\}_{n=0}^\infty$, such that $\epsilon_n \to 0$ and $\|\epsilon_n \nabla_x^2 W^\epsilon(x)\| \to 0$ uniformly for all $x \in \bar{O}$ when $n \to \infty$. Let us denote $\delta(\epsilon,\theta) = \frac{1}{N_S}\sum_i \left\|\epsilon \nabla_x^2 W^\epsilon(x_i,\theta)\right\|^2$. Further, we use the following scheme to generate a sequence $\{\epsilon\}_{n=0}^\infty$. Given the first element of the sequence ϵ_0, we get all the consecutive elements with the next update rule

$$\epsilon_{n+1} = \begin{cases} \frac{k_\epsilon \delta(\epsilon_n,\theta_{n-1})}{\delta(\epsilon_n,\theta_n)}\epsilon_n & \text{if } k_\epsilon \delta(\epsilon_n,\theta_{n-1}) \le \delta(\epsilon_n,\theta_n), \\ & \text{and } \mathcal{L}(\theta_i) \ge \mathcal{L}(\theta_{i-1})\ \forall i : n - n_\epsilon + 1 \le i \le n \\ \epsilon_n & \text{otherwise,} \end{cases} \tag{12}$$

where n corresponds to the current iteration number and $k_\epsilon \in (0,1)$ is a coefficient defining the speed with which $\|\epsilon_n \nabla_x^2 W^\epsilon(x)\|^2$ should decrease. Moreover, n_ϵ serves as a number of epochs since θ_i does not improve the loss, *i.e.* it also specifies the minimum number of epochs with fixed ϵ to obtain $W^\epsilon(x,\theta) \approx W^\epsilon(x)$ that solves Eq. (11). We also tried more simple update rules like a linear update rule $\epsilon_{n+1} = k_\epsilon \epsilon_n$, but gave much worse results, while our scheme results in $\|\epsilon_n \nabla_x^2 W^\epsilon(x)\|$ steadily going towards 0.

Finally, we define losses corresponding to Eq. (11) and Eq. (9). Let us assume that $\mathcal{S}_O \sim \mathcal{U}(O, N_F)$, a sample of points drawn uniformly from O of size N_F, and $\mathcal{S}_{\partial O} \sim \mathcal{U}(\partial O, N_B)$. Further, if ϵ and θ are indicated without an iteration number, then they correspond to the current iteration n. We have:

$$\mathcal{L}_O(\theta, \mathcal{S}_O) = \frac{1}{N_F}\sum_{i=1}^{N_F}\left(H(x_i, W^\epsilon(x_i,\theta), \nabla W^\epsilon(x_i,\theta)) - \epsilon \nabla^2 W^\epsilon(x_i,\theta)\right)^2 \qquad x_i \in \mathcal{S}_O, \tag{13}$$

$$\mathcal{L}_{\partial O}(\theta, \mathcal{S}_{\partial O}) = \frac{1}{N_B}\sum_{i=1}^{N_B}\left([-H(x_i, W^\epsilon(x_i,\theta), \nabla W^\epsilon(x_i,\theta) + \alpha\eta(x_i))]^+\right)^2 \qquad x_i \in \mathcal{S}_{\partial O}, \tag{14}$$

where $[f(x)]^+ = \max\{f(x), 0\}$. In addition to PDE-related and boundary-related losses, we introduce MSE regularization loss that should encourage uniform convergence:

$$\mathcal{L}_R(\theta, \mathcal{S}_O) = \frac{1}{N_F}\sum_{i=1}^{N_F}\left(W^\epsilon(x_i,\theta) - W^{\epsilon_{n-1}}(x_i, \theta_{\epsilon_{n-1}})\right)^2 \quad x_i \in \mathcal{S}_O \tag{15}$$

where $W^{\epsilon_{n-1}}(x, \theta_{\epsilon_{n-1}})$ is the best function obtained for ϵ_{n-1}. The final loss is:

$$\mathcal{L}(\theta, \mathcal{S}_O, \mathcal{S}_{\partial O}) = \mathcal{L}_O(\theta, \mathcal{S}_O) + \lambda \mathcal{L}_{\partial O}(\theta, \mathcal{S}_{\partial O}) + \lambda_R \mathcal{L}_R(\theta, \mathcal{S}_O). \tag{16}$$

Bringing everything together results in Algorithm 1.

Algorithm 1 ϵ-HJBPINNs

Set $\epsilon = \epsilon_0$, $\theta = \theta_0$, initialize $W^\epsilon(\cdot, \theta)$
for epoch n in $\{1, \ldots, \text{NB_ITER}\}$ **do**
  Generate datasets $\mathcal{D}_O(N_D)$ and $\mathcal{D}_{\partial O}(N_D)$ of N_D states uniformly sampled from O and ∂O respectively
  for batches $\mathcal{S}_O \in \mathcal{U}(\mathcal{D}_O, N_S)$ and $\mathcal{S}_{\partial O} \in \mathcal{U}(\mathcal{D}_{\partial O}, N_S)$ **do**
    Update $\theta_n := \theta_n - \nu \nabla_\theta \mathcal{L}(\theta, \mathcal{S}_O, \mathcal{S}_{\partial O})$, where $\mathcal{L}(\theta, \mathcal{S}_O, \mathcal{S})$ is computed with Eq. (16)
  end for
  Compute ϵ_n using Eq. (12)
  Set $\epsilon = \epsilon_n$ and $\theta = \theta_n$
end for

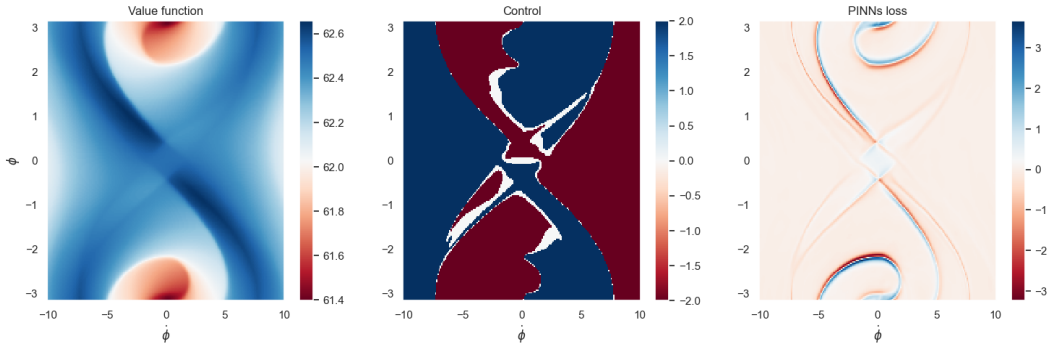


Figure 1: Value function, control and loss maps obtained with PINNs after 1000 epochs.

5 Experimental Results

In this section we evaluate the performance of methods from Section 4 on the inverted pendulum, which is a classical RL control task. We use a continuous-time adaptation of the inverted pendulum[1]. The state space consists of angle ϕ and angular speed $\dot{\phi}$. We consider the domain $O = \mathbb{R} \times [-20, 20]$. The set of actions is $\{-2, 0, 2\}$. In this paper, we consider only the boundary condition from Case 2 in Section 3, described with Eq. (9).

5.1 PINNs Results

In this section, we give the results of the training with Algorithm 1. We tried several types of architectures, such as Multi-Layer Perceptron (MLP) and Fourier-Feature Network (FFN) introduced in [33] with different number of layers and neurons. We have obtained the best performances with FFN, consisting of 3 layers, where the first layer is of size $d \times 40$ (this size of the first layer was recommended in the original article, where d is the dimensionality of state space) and other layers are 100 neurones each. The best performing activation function is `tanh`. It is important to use smooth activation functions for solutions of Eq. (11), using non-smooth activations like `relu` causes the training to fail. The hyperparameters for Algorithm 1 are given in Appendix A.2.2 together with the discussion on how they are chosen.

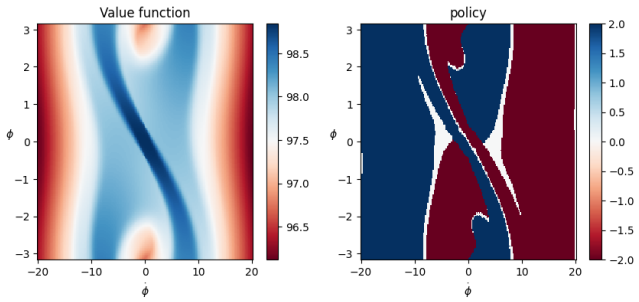
We have executed a training for 1000 iterations. In Figure 1, we present the value function, control and loss maps that we have obtained after the training. We observe that PINNs are able to find a general structure of value function, though it is less precise to approximate the non-smooth zones because of the smooth activation functions used in the neural networks. Thus, further research on how to improve this approach is necessary.

[1]We used a slightly modified environment taken from `https://github.com/cagatayyildiz/oderl`

5.2 Comparison with DTRL and Dynamic Programming Algorithms

Finally, we evaluate the above methods on discrete time inverted pendulum with small $dt = 0.001$[2] to compare their performance with PPO and A2C, which are state-of-the-art DTRL algorithms. A complete discussion about our experiments on dynamic programming algorithm can be found in A.2.1. To test the algorithms, we run them for 100 episodes, each episode has 5000 steps. In case of dynamic programming, we also vary the grid size N to test its performance (then the discretization step is $\delta = 1/N$). We report in Figure 2 (right) the mean and the standard deviation. We also present the value function and the policy obtained by the dynamic programming algorithm with a resolution of 200 by 200 ($N = 200$) in Figure 2 (left).

As expected for the dynamic programming algorithm, the cumulative reward grows when N is increased. However, from 30 by 30, the gain in perfomance is marginal with respect to the increase of computational cost. Conversely, PINNs is performing worse than VI as it struggles to reach the optimal control in the area where the value function is non-smooth, but it still performs much better than PPO or A2C that are unable to learn the optimal policy function for such a small dt.



method	N	mean	std
VI	5	159.56	287.12
VI	10	1621.81	963.94
VI	15	2010.99	485.91
VI	20	3924.99	637.41
VI	30	3802.49	552.67
VI	124	4006.83	468.13
VI	200	4133.71	433.02
PINNs	NA	3127.02	660.04
PPO	NA	1660.92	999.80
A2C	NA	868.33	450.69

Figure 2: The value function and the optimal policy for the inverted pendulum (left). Mean and standard deviation of the cumulative reward for different methods and several grid sizes for VI (right).

6 Conclusion

In this article, we consider the problem of finding the value function of the optimal policy in the context of continuous time for deterministic dynamical system relying on the HJB equation. There are multiple ways for solving this equation, but one should make sure that the final solution is the viscosity solution to get the value function. We propose methods to compute viscosity solutions, making a particular focus on NN-based approaches. We compare some of those methods on a control task such as the inverted pendulum. On the one hand, our results have shown that the dynamic programming algorithm is efficient for getting a precise solution once the discretization time step is small enough. However, scaling this algorithm to more difficult problems is limited as the complexity grows exponentially with respect to the dimensionality of the state space. On the other hand, PINNs method based on neural networks can be applied to high dimensional problems, but performance depends on the optimization methods that do not always converge to the global minimum. The PINNs algorithm proposed in this paper is designed to find the viscosity solution by approximating it with smooth solutions of other PDE equations (11). Even though, it is able to approximate well the solution where it is smooth, it still struggles to match precisely the solution at the points of non-smoothness, which causes a bad control at those points. Possible improvements include considering more sophisticated neural network architectures such as Mixture of Experts or Physics Informed Neural Operators, augmenting the PINNs loss with the data driven loss (*e.g.* by integrating the discrete time solutions into the observational loss) or considering more advanced sampling techniques.

[2]Note that in the gym inverted pendulum environment dt is set to $1/20$

References

[1] Dipak M Adhyaru, IN Kar, and Madan Gopal. Bounded robust control of nonlinear systems using neural network–based hjb solution. *Neural Computing and Applications*, 20:91–103, 2011.

[2] Bastian Alt, Matthias Schultheis, and Heinz Koeppl. Pomdps in continuous time and discrete spaces. *Advances in Neural Information Processing Systems*, 33:13151–13162, 2020.

[3] Piermarco Cannarsa, Fausto Gozzi, and Halil Mete Soner. A boundary-value problem for hamilton-jacobi equations in hilbert spaces. *Applied Mathematics and Optimization*, 24(1):197–220, 1991.

[4] Tao Cheng, Frank L Lewis, and Murad Abu-Khalaf. Fixed-final-time-constrained optimal control of nonlinear systems using neural network hjb approach. *IEEE Transactions on Neural Networks*, 18(6):1725–1737, 2007.

[5] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. Theses, Institut National Polytechnique de Grenoble - INPG, June 2002.

[6] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

[7] Jérôme Darbon, Peter M Dower, and Tingwei Meng. Neural network architectures using min-plus algebra for solving certain high-dimensional optimal control problems and hamilton–jacobi pdes. *Mathematics of Control, Signals, and Systems*, 35(1):1–44, 2023.

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

[9] Wendell H Fleming and Halil Mete Soner. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

[10] Christian Grossmann, Hans-Görg Roos, and Martin Stynes. *Numerical treatment of partial differential equations*, volume 154. Springer, 2007.

[11] Igor Halperin. Distributional offline continuous-time reinforcement learning with neural physics-informed pdes (sciphy rl for doctr-l). *arXiv preprint arXiv:2104.01040*, 2021.

[12] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[13] Hitoshi Ishii. Uniqueness of unbounded viscosity solution of hamilton-jacobi equations. *Indiana University Mathematics Journal*, 33(5):721–748, 1984.

[14] Wei Kang and Lucas C. Wilcox. Mitigating the curse of dimensionality: Sparse grid characteristics method for optimal feedback control and hjb equations, 2016.

[15] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

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

[17] Jeongho Kim and Insoon Yang. Hamilton-jacobi-bellman equations for maximum entropy optimal control. *arXiv preprint arXiv:2009.13097*, 2020.

[18] Derong Liu, Ding Wang, Fei-Yue Wang, Hongliang Li, and Xiong Yang. Neural-network-based online hjb solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE transactions on cybernetics*, 44(12):2834–2847, 2014.

[19] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[20] Michael Lutter, Boris Belousov, Kim Listmann, Debora Clever, and Jan Peters. Hjb optimal feedback control with deep differential value functions and action constraints. In *Conference on Robot Learning*, pages 640–650. PMLR, 2020.

[21] Michael Lutter, Shie Mannor, Jan Peters, Dieter Fox, and Animesh Garg. Robust value iteration for continuous control tasks. *arXiv preprint arXiv:2105.12189*, 2021.

[22] Michael Lutter, Shie Mannor, Jan Peters, Dieter Fox, and Animesh Garg. Value iteration in continuous actions, states and time. *arXiv preprint arXiv:2105.04682*, 2021.

[23] Amartya Mukherjee and Jun Liu. Bridging physics-informed neural networks with reinforcement learning: Hamilton-jacobi-bellman proximal policy optimization (hjbppo). *arXiv preprint arXiv:2302.00237*, 2023.

[24] R. Munos, L.C. Baird, and A.W. Moore. Gradient descent approaches to neural-net-based solutions of the hamilton-jacobi-bellman equation. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 3, pages 2152–2157 vol.3, 1999.

[25] Remi Munos. A Study of Reinforcement Learning in the Continuous Case by the Means of Viscosity Solutions. *Machine Learning*, 40:265–299, 2000.

[26] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.

[27] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[28] Halil Mete Soner. Optimal control with state-space constraint i. *SIAM Journal on Control and Optimization*, 24(3):552–561, 1986.

[29] Yuval Tassa and Tom Erez. Least squares solutions of the hjb equation with neural network value-function approximators. *IEEE transactions on neural networks*, 18(4):1031–1041, 2007.

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[31] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *The Journal of Machine Learning Research*, 21(1):8145–8178, 2020.

[32] Harley E Wiltzer, David Meger, and Marc G Bellemare. Distributional hamilton-jacobi-bellman equations for continuous-time reinforcement learning. In *International Conference on Machine Learning*, pages 23832–23856. PMLR, 2022.

[33] Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In *International Conference on Learning Representations*, 2022.

[34] Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning, 2021.

A Appendix

A.1 Possible Approaches. Supplementary

A.1.1 Dynamic Programming

In the FEM case, we use a triangulation Σ^δ to cover the state space. It is also possible to discretize the control space, denoted by U^δ. The vertices of the triangulation Σ^δ are denoted $\{\xi_1, \xi_2, ..., \xi_{N_\delta}\}$

with $N_\delta \in \mathbb{N}$. In this setting, V is approximated by a piecewise linear function V^δ. Thus, for $x \in Simplex(\xi_0, .., \xi_d)$, we have

$$V^\delta(x) = \sum_{i=0}^{d} \lambda_{\xi_i}(x) V^\delta(\xi_i)$$

where $\lambda_{\xi_i}(x)$ is the barycentric coordinates inside the simplex $(\xi_0, ..., \xi_d)$.

By using a FEM approximation scheme, the HJB equation is transformed into:

$$V^\delta(\xi) = \sup_{u \in U^\delta} [\gamma^{\tau(\xi,u)} V^\delta(\eta(\xi, u)) + \tau(\xi, u) r(\xi, u)]$$

where $\eta(\xi, u) = \xi + \tau(\xi, u) f(\xi, u)$ and $\tau(\xi, u)$ is a time discretization function that should satisfy:

$$\exists k_1, k_2 > 0, \forall \xi \in \Sigma^\delta, \forall u \in U^\delta, k_1\delta \leq \tau(\xi, u) \leq k_2\delta$$

If F^δ is defined as $F^\delta[\phi](\xi) = \sup_{u \in U^\delta}[\gamma^{\tau(\xi,u)} \sum_{i=0}^{d} \lambda_{\xi_i}(\eta(\xi, u))\phi(\xi_i) + \tau(\xi, u) r(\xi, u)]$, it is possible to show that F^δ satisfies a contraction property, and since $V^\delta(\xi) = F^\delta[V^\delta](\xi)$ holds, dynamic programming techniques can be applied to compute V^δ. Moreover, it can be proved that $V^\delta \underset{\delta \to 0}{\longrightarrow} V$ uniformly on any compact of the state space. With this method, one can derive algorithms that converge toward V, without even knowing the dynamics of the system (see [25] for more details).

A.1.2 PINNs

To solve a differential equation

$$F(x, W(x), \nabla_x W(x), \nabla_x^2 W(x)) = 0, \quad W : \bar{O} \to \mathbb{R}, x \in O \tag{17}$$

with K_1 equality boundary conditions

$$B_i(x, W(x), \nabla_x W(x), \nabla_x^2 W(x)) = 0, \quad x \in \partial O, i \leq K_1 \tag{18}$$

and with K_2 inequality boundary conditions

$$G_i(x, W(x), \nabla_x W(x), \nabla_x^2 W(x)) \leq 0, \quad x \in \partial O, i \leq K_2. \tag{19}$$

one can assume that $W(x)$ lies in the class of functions $\mathcal{F}_\theta = \{f_\theta(x) = NN(x, \theta) : \theta \in \Theta\}$ represented by neural networks of a fixed architecture and parametrized with weights $\theta \in \Theta$. If it is the case then there exists θ such that $W(x, \theta)$ should satisfy Eq. (17) and thus minimize the loss

$$\mathcal{L}_{PDE}(\theta) = \frac{1}{N_F} \sum_{i=1}^{N_F} \left(F(x_i, W(x_i, \theta), \nabla_x W(x_i, \theta), \nabla_x^2 W(x_i, \theta))\right)^2 \quad \forall x_i \in \mathcal{S}_u(O, N_F), \tag{20}$$

with $\mathcal{S}_u(O, N_F)$ denoting a sample of N_F points drawn uniformly from O. If the solution $W(x, \theta)$ should satisfy some additional boundary constraints then it should also minimize the boundary losses for all $k \leq K_1$ and $k' \leq K_2$

$$\mathcal{L}_{B_k}(\theta) = \frac{1}{N_B^k} \sum_{i=1}^{N_B^k} \left(B_k(x_i, W(x_i, \theta), \nabla_x W(x_i, \theta), \nabla_x^2 W(x_i, \theta))\right)^2 \quad \forall x_i \in \mathcal{S}_u(\partial O, N_B^k) \tag{21}$$

$$\mathcal{L}_{G_{k'}}(\theta) = \frac{1}{N_G^{k'}} \sum_{i=1}^{N_G^{k'}} \left(\left[G_{k'}(x_i, W(x_i, \theta), \nabla_x W(x_i, \theta), \nabla_x^2 W(x_i, \theta))\right]^+\right)^2 \quad \forall x_i \in \mathcal{S}_u(\partial O, N_G^{k'}), \tag{22}$$

where $[f(x)]^+ = \max\{f(x), 0\}$.

To put everything together, when solving a PDE in a PINNs-like manner, one should train a neural network $W(x, \theta)$ that minimizes:

$$\mathcal{L}(\theta) = \mathcal{L}_{PDE}(\theta) + \sum_{k=1}^{K_1} \lambda_k \mathcal{L}_{B_k}(\theta) + \sum_{k=1}^{K_2} \lambda'_k \mathcal{L}_{G_k}(\theta). \tag{23}$$

where $\lambda_k, \lambda'_k > 0$ are some mixing coefficients for different boundary conditions.

A.2 Experimental Results. Supplementary

A.2.1 Dynamic Programming Experimental Results

In this section, we present the results obtained with one of the algorithms proposed in [25]. First, a grid is built by dividing each axis by N points. Then, we use the Delaunay's triangulation over the grid and apply the value iteration algorithm to the FEM-MDP derived in [25].

We set $\delta = \frac{1}{N}$, δ being the discretization step. The stopping criterion used at the step n is $\|V_n - V_{n-1}\|_\infty \leq \epsilon$ where ϵ is a chosen tolerance. In our experiments we work with $\epsilon = 10^{-5}$.

When δ goes towards 0, our approximated value function, V^δ, converges towards the true value function. In our case, $\delta \to 0$ is equivalent to $N \to +\infty$. Empirically, we can see in Figure 3 that this property is satisfied. Indeed, as we increase N, we obtain a more accurate V^δ, and as a result, a better control that leads to a higher cumulative reward.

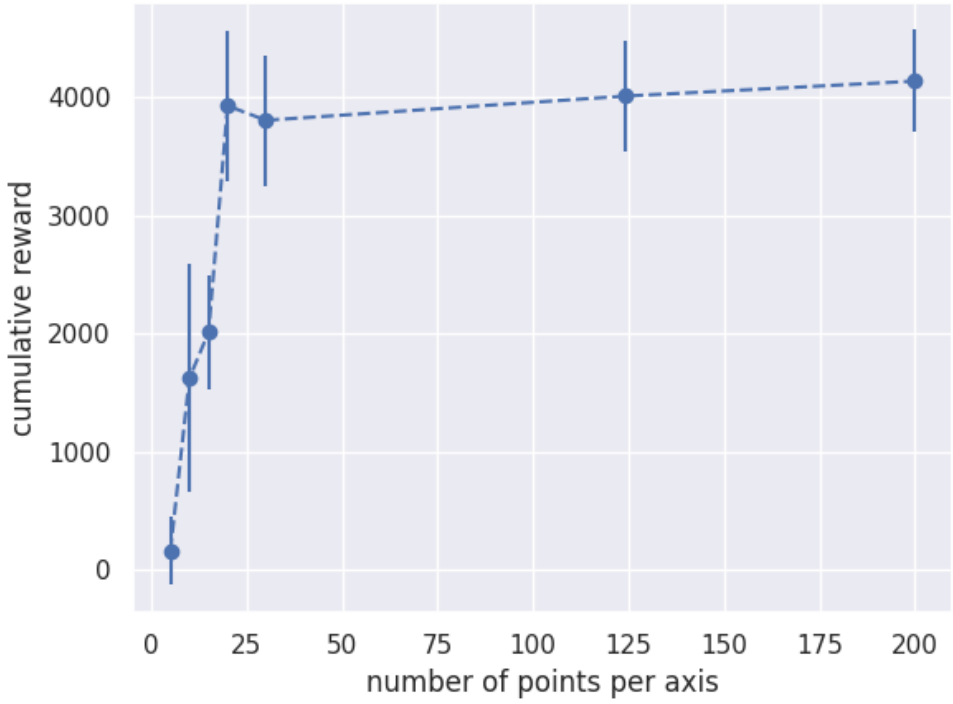


Figure 3: The cumulative reward obtained on the inverted pendulum for different grid sizes N

A.2.2 Ablation Study on Regularization

The best performing hyperparameters are gathered in Table 1.

For the inverted pendulum training, we tried several $\epsilon_0 \in (10^{-7}, 10^{-1})$ and $k_\epsilon \in \{0.9, 0.99, 0.999, 0.9999\}$, and we did not observe any significant change in performance for different set of parameters. In what follows, we keep $\epsilon_0 = 10^{-2}$ and $k_\epsilon = 0.99$.

In addition, we study how regularization term λ_R affects the training. We demonstrate the performance of Algorithm 1 with 3 metrics: PDE loss $\mathcal{L}_O$, boundary loss $\mathcal{L}_{\partial O}$ and ϵ-added term $\delta(\epsilon, \theta)$. All measurements are performed using 5 seeds. From Figure 4, one can conclude that $\lambda_R = 10^{-6}$ achieves the best performance for all metrics, though the values close to it result in the similar behaviour. From the same figure, we see that high λ_R can be too restrictive, resulting into slow convergence. However, putting λ_R too low can lead to more instabilities. Thus, $\lambda_R = 10^{-6}$ is a good trade-off, though it does not remove all the risk of instabilities.

Hyperparameters	values
N_D	200000
N_S	100
ν	0.00085
n_ϵ	7
λ	10^{-2}
λ_R	10^{-6}
ϵ_0	10^{-2}
k_ϵ	0.99

Table 1: Hyperparameters for Algorithm 1

Figure 4: PINNs loss, boundary loss and $\delta(\epsilon, \theta)$ with respect to different λ_R.