Physics-Informed Neural Network Policy Iteration: Algorithms, Convergence, and Verification

Yiming Meng [*1] **Ruikun Zhou** [*2] **Amartya Mukherjee** [2] **Maxwell Fitzsimmons** [2] **Christopher Song** [2] **Jun Liu** [2]

Abstract

Solving nonlinear optimal control problems is a challenging task, particularly for high-dimensional problems. We propose algorithms for model-based policy iterations to solve nonlinear optimal control problems with convergence guarantees. The main component of our approach is an iterative procedure that utilizes neural approximations to solve linear partial differential equations (PDEs), ensuring convergence. We present two variants of the algorithms. The first variant formulates the optimization problem as a linear least square problem, drawing inspiration from extreme learning machines (ELM) for solving PDEs. This variant efficiently handles low-dimensional problems with high accuracy. The second variant is based on a physics-informed neural network (PINN) for solving PDEs and has the potential to address high-dimensional problems. We demonstrate that both algorithms outperform traditional approaches, such as Galerkin methods, by a significant margin. We provide a theoretical analysis of both algorithms in terms of convergence of neural approximations towards the true optimal solutions in a general setting. Furthermore, we employ formal verification techniques to demonstrate the verifiable stability of the resulting controllers.

1. Introduction

Reinforcement learning in discrete environments has achieved remarkable success over the past decade, from AlphaGo (Silver et al., 2017) to the recent breakthrough of GPT-3 (Ouyang et al., 2022), which uses reinforcement learning (Schulman et al., 2017) from human feedback to fine-tune large language models. However, reinforcement learning in continuous environments, where states and actions evolve continuously in both space and time, remains a challenge (Duan et al., 2016). Theoretically speaking, when considering continuous-time scenarios, the discrete-time Bellman equation is replaced by a nonlinear partial differential equation (PDE) known as the Hamilton–Jacobi–Bellman (HJB) equation. Solving and analyzing this equation, in general, becomes a complex task due to its intricate nature. One major challenge arises from the possibility that the optimal cost function may not be differentiable, even for relatively straightforward problems (Bertsekas, 2015). In such cases, one has to resort to viscosity solutions (Crandall et al., 1984) to study HJB equations.

A rich literature exists on policy iteration techniques for obtaining suboptimal solutions to the HJB equations (Leake & Liu, 1967; Saridis & Lee, 1979; Beard, 1995; Beard et al., 1997; 1998). One notable approach is to construct successive approximations to solutions of the so-called Generalized Hamilton-Jacobi-Bellman (GHJB) equation, which is a linear PDE and potentially easier to solve. Galerkin approximations for solving the GHJB are proposed in (Beard et al., 1997; 1998) and have proven effective for solving low-dimensional problems. However, such approaches do not scale well to high-dimensional problems. Indeed, Galerkin methods are known to suffer from the curse of dimensionality.

Motivated by recent successes in solving PDEs using neural networks (Raissi et al., 2019; Huang et al., 2006; Chen et al., 2022; Han et al., 2018; Sirignano & Spiliopoulos, 2018; Weinan et al., 2021) and the potential of neural networks to overcome the curse of dimensionality (Poggio et al., 2017), we set out to revisit the policy iteration approach by solving GHJB equations using neural networks. The main goal is to answer the following questions:

1. Can neural approximations of the solutions to GHJB converge to the viscosity solution of the HJB equation?

2. Can neural approximations efficiently compute solutions of the HJB with high accuracy?

[*]Equal contribution [1]Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Champaign–Urbana, Illinois, United States [2]Department of Applied Mathematics, University of Waterloo, Waterloo, Canada. Correspondence to: Jun Liu <j.liu@uwaterloo.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

3. Can neural policy iteration overcome the curse of dimensionality?

4. Can neural approximations be guaranteed to lead to stabilizing controllers?

The answers to these questions are all positive to some degree. The main contributions of this paper are as follows:

1. We prove that policy iteration indeed converges to viscosity solutions of the HJB equation.

2. We propose two variants of neural policy iteration. The first, inspired by the Extreme Learning Machine (Huang et al., 2006) and termed ELM-PI, can achieve remarkable accuracy and efficiency on low-dimensional problems. The second, based on Physics-Informed Neural Networks (PINN) (Raissi et al., 2019) for solving PDEs, has been shown to scale better than ELM-PI as dimensions increase.

3. We formulate formal verification problems for the resulting controllers to verify their stability. We show with a simple example that seemingly convergent results can lead to unstable controllers, which necessitate the use of formal verification when safety is a concern.

Related work: (1) The idea of policy iteration in the context of designing optimal stabilizing controllers has a long history. For linear systems, this reduces solving an algebraic Riccati equation (ARE), which is quadratic in the unknown matrix, into a sequence of Lyapunov equations, which are linear and easier to solve. The algorithm is known as Kleinman's algorithm (Kleinman, 1968). In the nonlinear case, this procedure reduces the nonlinear HJB equation to a sequence of linear PDEs that characterize the value (and Lyapunov) functions for the stabilizing controllers obtained at each iteration of policy evaluation. This result dates back at least to 1960s (Milshtein, 1964; Vaisbord, 1963) and followed by (Leake & Liu, 1967; Saridis & Lee, 1979; Beard, 1995; Jiang & Jiang, 2017; Bhasin et al., 2013; Vrabie & Lewis, 2009; Jiang & Jiang, 2012; 2014) and many others. To the best knowledge of the authors, however, none of these works establish the convergence of policy iteration to viscosity solutions (Crandall et al., 1984) of the HJB equation, especially when function approximators are involved. Furthermore, classical computational approaches, such as Galerkin methods, for solving PDEs often do not scale well.

(2) We draw significant inspiration from recent work on neural networks for solving PDEs (Raissi et al., 2019; Huang et al., 2006; Chen et al., 2022; Han et al., 2018; Sirignano & Spiliopoulos, 2018) (see the recent survey (Weinan et al., 2021) and discussions on the potential for machine learning to overcome the curse of dimensionality when solving PDEs). More relevantly, the authors in (Furfaro et al., 2022) solved HJB equation directly using PINN to tackle optimal control problems for aerospace systems. To the best of our knowledge, no previous work has reported the use of neural PDE solving for policy iterations in the context of nonlinear optimal control on benchmark problems. We address these gaps in this paper.

2. Problem formulation

We consider a class of optimal control problems subject to control-affine dynamical systems of the form

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable vector field and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is smooth, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input. We also assume that $f(\mathbf{0}) = \mathbf{0}$.

We are interested in the case where the maximal interval of existence is $[0, \infty)$ for admissible controls. For simplicity, in the context of infinite-horizon trajectory, we overload the notation u as the control signal, i.e. $u : [0, \infty) \to \mathbb{R}^m$. Subject to the control u, the unique solution starting from x_0 is denoted by $\phi(t; x_0, u)$. We may also write the solution as $\phi(t)$ or ϕ if the rest of the arguments are not emphasized.

Introduce $L(x, u) = Q(x) + \|u\|_{\mathrm{R}}^2$, where $Q : \mathbb{R}^n \to \mathbb{R}$ is a positive definite function, and $\|u\|_{\mathrm{R}} = u^T Ru$, where $R : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ is a symmetric and positive definite matrix. The associated cost is then commonly defined as:

$$J(x, u) = \int_0^\infty L(\phi(s; x, u), u(s))ds. \tag{2}$$

Definition 2.1 (Admissible Controls)**.** *Given a subset $\Omega \subseteq \mathbb{R}^n$ containing the origin. A control $u : \Omega \to \mathbb{R}^m$ is admissible on Ω, denoted as $u \in \mathcal{U}(\Omega)$ or simply $u \in \mathcal{U}$, if (1) u is Lipschitz continuous on Ω; (2) $u(\mathbf{0}) = \mathbf{0}$; (3) u is a stabilizing control, i.e., $\lim_{t\to\infty} |\phi(t; x_0, u)| = 0$ for all $x_0 \in \Omega$; and (4) $J(x_0, u) < \infty$.*

Let $V : \mathbb{R}^n \to \mathbb{R}$ be the value function for this problem, i.e.,

$$V(x) := \inf_{u\in\mathcal{U}} J(x, u). \tag{3}$$

We aim to find V as well as the associated optimal control u^*. If we introduce

$$G(x, u, p) := L(x, u) + p \cdot (f(x) + g(x)u), \tag{4}$$

where $p \in \mathbb{R}^n$, and define the Hamiltonian

$$H(x, p) = \sup_{u\in\mathbb{R}^m} -G(x, u, p), \tag{5}$$

then V is generally a viscosity solution (see Appendix A for a formal definition) within $C(\Omega)$ to the HJB equation

$$H(x, DV(x)) = 0. \tag{6}$$

We show this in the proof of Proposition 2.4.

Remark 2.2. *The basic properties of viscosity solutions are discussed in Appendix A. The concept of viscosity solution relaxes C^1 solutions. Note that, at differentiable points, $DV(x)$ exists and $\{DV(x)\} = \partial^+ V(x) = \partial^- V(x)$. In this case, to justify a viscosity solution, we can simply substitute $DV(x)$ and check if $F(x, V(x), DV(x)) = 0$ pointwise, where F is defined in Definition A.1. If V is not differentiable at a given point, then we have to go through the conditions in Definition A.1 to verify that V is a viscosity solution. A classical example is that $V(x) = 1 - |x|$, $x \in \mathbb{R}$, is a viscosity solution to $|DV| - 1 = 0$ with boundary conditions $V(-1) = V(1) = 0$. To verify this, we only have to check if (1) and (2) in Definition A.1 are satisfied at 0.* □

We also provide the following nice properties and complete the proofs in Appendix B.

Proposition 2.3 (Dynamic Programming Principle)**.** *For all $x \in \Omega \subseteq \mathbb{R}^n$ and $t > 0$,*

$$V(x) = \inf_{u \in \mathcal{U}} \left\{ \int_0^t L(\phi(s; x, u), u(s)) ds + V(\phi(t; x, u)) \right\}. \tag{7}$$

Proposition 2.4 (Uniqueness of Viscosity Solution)**.** *The V defined in (3) is the unique viscosity solution of $H(x, DV(x)) = 0$.*

Theorem 2.5 (Optimal Feedback Control)**.** *Let $\kappa : \Omega \to \mathbb{R}^m$ be locally Lipschitz continuous. Suppose that $u^*(\cdot) := \kappa(\phi(\cdot))$ and $u^* \in \mathcal{U}$. If V is the viscosity solution of $G(x, \kappa(x), DV(x)) = 0$, then $J(x, \kappa(\phi(\cdot))) = V(x)$.*

Note that $G(x, u, p)$ is minimized given that $u(x) = -\frac{1}{2} R^{-1} g^T(x) p^T$. With this, the HJB equation reduces to

$$\begin{aligned} H(x, DV(x)) = &- Q(x) - DV(x) \cdot f(x) \\ &+ \frac{1}{4} DV(x) g(x) R^{-1} g^T(x) (DV(x))^T. \end{aligned} \tag{8}$$

We can either numerically solve the nonlinear PDE (8) or use policy iteration to approximate V and the optimal controller. The conventional policy iteration (Bardi et al., 1997; Jiang & Jiang, 2017) assumes that $V \in C^1(\Omega)$ and seeks C^1 solutions V_i to the GHJBs $G(x, u_i, DV_i(x)) = 0$ for each $i \in \{0, 1, \cdots\}$, where $u_i = -\frac{1}{2} R^{-1} g^T(x) DV_i(x)$ for $i \in \{1, 2, \cdots\}$ and $u_0 \in \mathcal{U}$. The convergence value function V_∞ is expected to solve (8) and $u_i \to u^*$ at least pointwise (Theorem 3.1.4, Jiang & Jiang, 2017). The numerical solution of GHJBs, which are linear, is commonly believed to be achieved more easily.

However, the continuous differentiability (on Ω) of $\{V_i\}$ and V are assumed without justification (Bardi et al., 1997; Jiang & Jiang, 2017), leading to uncertainty regarding the applicability of the obtained results. Even though $V_i \in C^1(\Omega)$ for all i, the limit V_∞ w.r.t. the uniform norm in (Theorem 3.1.4, Jiang & Jiang, 2017) may not be continuously differentiable, and hence may not be the approximation of $V \in C^1(\Omega)$. Motivated by this, we characterize solutions to GHJB equations and demonstrate in Section 3.1 that the exact policy iteration based on viscosity solutions converges to the viscosity solution of the HJB. The convergence analysis differs from the conventional case. Based on this analysis, we show in Section 3 that the neural policy iteration algorithms, ELM-PI and PINN-PI, also converge to viscosity solutions under less restrictive assumptions. The algorithms will be presented in Section 3, and the convergence analysis will be discussed in Section 4.

3. Algorithms

3.1. Exact policy iteration

To begin, we provide an overview of the theoretical foundation of policy iteration. Policy iteration (PI) originates optimal control of Markov decision processes (MDP) (Bellman, 1957; Howard, 1960) (additional references can be found in recent texts and monographs (Bertsekas, 2012; 2019)). In this section, we present a fundamental version of PI for the system (1) with the cost (2). The algorithm dates back to the 1960s (Leake & Liu, 1967; Milshtein, 1964; Vaisbord, 1963; Kleinman, 1968), and its convergence is established in various sources (Saridis & Lee, 1979; Beard, 1995; Milshtein, 1964; Vaisbord, 1963; Jiang & Jiang, 2017; Farsi & Liu, 2023). However, all these proofs rely on strong assumptions on the smoothness of the optimal value function, which may or may not be satisfied in general by the solutions to the HJB equation associated with the optimal control problem. To illustrate this, consider the bilinear scalar problem $\dot{x} = xu$, with $Q(x) = x^2$ and $R = 1$. The optimal value function is $V(x) = 2|x|$, which fails to be differentiable at $x = 0$. We provide a regularity analysis in the general setting where viscosity solutions are allowed (see Section 4).

We now define policy iteration with exact solutions to PDEs, which we refer to as exact-PI. This process begins with an initial policy $u = \kappa_0(x)$, where $\kappa_0(\mathbf{0}) = \mathbf{0}$. This initial policy is assumed to be an admissible controller. For each $i \geq 0$, exact-PI performs the following two steps iteratively:

1. (**Policy evaluation**) Compute a value function $V_i(x)$ at all $x \in \Omega \setminus \{\mathbf{0}\}$ for the policy κ_i by solving the GHJB

$$\begin{aligned} G(x, \kappa_i(x), DV_i(x)) :=& Q(x) + \kappa_i^T(x) R(x) \kappa_i(x) \\ &+ DV_i(x)(f(x) + g(x)\kappa_i(x)) \\ =& 0. \end{aligned} \tag{9}$$

   We set $V_i(\mathbf{0}) = 0$ for all $i \geq 0$.

2. (**Policy improvement**) Update the policy

$$\kappa_{i+1}(x) = \begin{cases} -\frac{1}{2}R^{-1}g^T(x)(DV_i(x))^T, & \text{if } x \neq \mathbf{0}; \\ \mathbf{0} \text{ otherwise.} \end{cases} \tag{10}$$

The exact-PI algorithm is impractical because the exact solution of the linear PDE (9) is generally unavailable. In the following sections, we propose two algorithms for neural policy iterations by solving this PDE iteratively using function approximators.

3.2. ELM-PI via linear least squares

The first algorithm uses a one-layer function of the form

$$\hat{V}(x) := V(x;\beta) = \beta^T\sigma(Wx+b), \tag{11}$$

where $\beta \in \mathbb{R}^m$, $W \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function applied element-wise. It can be easily verified that the gradient of V with respect to x takes the form

$$DV(x;\beta) = \beta^T \text{diag}(\sigma'(Wx+b))W, \tag{12}$$

where diag maps the m-vector $\sigma'(Wx+b)$ to an $m \times m$ diagonal matrix and $\sigma'(\cdot)$ is the derivative of σ applied element-wise.

We will briefly describe how to solve a PDE via optimization in this paragraph. Suppose we would like to solve a PDE $H(x, DV(x)) = 0$. A general idea for solving a PDE via optimization is to collect a number of collocation points $\{x_s\}_{s=1}^N$, on which we evaluate the derivative $DV(x;\beta)$ of a parameterized, potential solution $V(x;\beta)$ and formulate the residual loss with mean squared error (MSE) as

$$\begin{aligned}&\text{Loss}(\beta)\\ =&\frac{1}{N}\sum_{s=1}^N H(x_s, DV(x_s;\beta))^2 + \lambda\sum_{p=1}^{N_b}(V(y_p;\beta) - \hat{V}(y_p))^2,\end{aligned} \tag{13}$$

where $\lambda > 0$ is a weight parameter, the points $\{y_p\}_{p=1}^{N_b}$ are boundary points and $\hat{V}(y_p)$ describes the boundary value at these points.

To efficiently solve (9) by (11), the main idea is to randomize W and b and then fix them when optimizing Loss(β) defined by (13). Due to the linearity of $V(x;\beta)$ in β, the linearity of the PDE (9), and definition of Loss(β) by (13), we obtain a linear least square optimization problem, which can be solved efficiently and accurately for moderate-sized problems. We call this ELM-PI and describe it in Algorithm 1. Here, the boundary condition[1] is simply $V(\mathbf{0}) = 0$, because the the origin is an equilibrium point without $u = 0$.

[1]Alternatively, in this setting, we can simply set $V(\mathbf{0}) = 0$ by subtracting a nonzero $V(\mathbf{0})$ as a bias term. This does not affect the subsequent controller.

From (9), (11), and (12), the loss function of β reduces to

$$\begin{aligned}&\text{Loss}(\beta)\\ =&\frac{1}{N}\sum_{s=1}^N H(x_s, \beta^T\text{diag}(\sigma'(Wx+b))W)^2 + \lambda(\beta^T\sigma(b))^2,\end{aligned} \tag{14}$$

where H is given by the left-hand side of (9) and linear in DV. Hence minimizing (14) with β is a linear least square problem.

From our experiments, it appears immaterial whether Steps 2 and 3 of Algorithm 1 are placed within or outside of the loop. In other words, we can use the same set of parameters W and b as well as the set of collocation points for all iterations.

3.3. PINN-PI via physics-informed neural network

Physics-informed neural networks (PINN) (Raissi et al., 2019) are a popular method for solving PDEs. We propose a variant for performing physics-informed neural policy iteration in this subsection.

For each i, instead of assuming that a solution $V_i(x)$ to Equation (9) takes the form (11), we consider a more general approach by assuming it to be a neural network function:

$$\hat{V}_i(x) := V_{i,\text{NN}}(x;\theta), \tag{15}$$

where $V_{i,\text{NN}}$ represents a feedforward neural network with potentially multiple layers and nonlinear activation functions. In this formulation, θ represents the parameters of the neural network, allowing for a flexible and adaptable representation of the solution $V_i(x)$. Even with just one hidden layer, the PINN approach would be allowed to change all parameters in the optimization process, leading to a non-convex optimization problem. Gradient descent methods are usually used to solve these large-scale non-convex optimization problems.

Similar to ELM-PI, at each iteration, we choose a set of collocation points $\{x_s\}_{s=1}^N$, evaluate the derivatives $DV_{i,\text{NN}}$ of $V_{i,\text{NN}}$ at these points using automatic differentiation, and form a residual loss

$$\text{Loss}(\theta) = \frac{1}{N}\sum_{s=1}^N H(x_s, DV_{i,\text{NN}}(x_s;\theta))^2 + \lambda(V_{i,\text{NN}}(0;\theta))^2, \tag{16}$$

which is in general a non-convex function of θ.

We describe PINN-PI in Algorithm 2.

Algorithm 1 Extreme Learning Machine Policy Iteration (ELM-PI)

Require: f, g, Q, R, k_0, Ω, N, m
repeat
  Generate random W and b
  Generate random $\{x_s\}_{s=1}^N \subseteq \Omega$
  Finding β that minimizes (14) to form V_i from (11)
  Update κ_{i+1} according to (10)
  $i = i + 1$
until desired accuracy or max iterations reached

Algorithm 2 Physics-Informed Neural Network Policy Iteration (PINN-PI)

Require: f, g, Q, R, k_0, Ω, $V_{i,\text{NN}}(x;\theta)$
repeat
  Generate random $\{x_s\}_{s=1}^N \subseteq \Omega$
  repeat
    Run gradient descent on θ with (16)
  until desired accuracy or max epochs reached
  Form $V_i(x)$ from $V_{i,\text{NN}}(x;\theta)$
  Update κ_{i+1} according to (10)
  $i = i + 1$
until desired accuracy or max iterations reached

A natural question to ask is when to terminate the algorithm. Clearly, there is no guarantee that gradient descent will find a global minimum θ^* for (16). Even if it does, the resulting $V_{i,\text{NN}}(x;\theta^*)$ will not satisfy (9) precisely. What one can hope for is that when the observed loss is sufficiently small and the number of iterations become large, $V_{i,\text{NN}}(x;\theta^*)$, where θ^* is a returned minimizer, can approximate the optimal solution to an arbitrary precision. In Section 4, we provide a convergence analysis and further discussion on this issue. In practice, the algorithm will terminate when either a desired accuracy is reached or a predetermined number of iterations is completed. In such cases, due to the approximation errors, it is unclear whether the resulting controller is stabilizing. We provide a verification framework to address this issue, which will be discussed in Section 3.5.

Remark 3.1. *In this work, we introduce ELM-PI and PINN-PI as two different approaches for solving the linear PDE* (9) *for policy iteration. Both methods are physics-informed, capturing the same PDE, but they differ in their formulation and optimization. With a one hidden-layer neural network, PINN-PI has the same architecture as ELM-PI, but ELM-PI uses linear least squares (for optimizing the output-layer weight only) while PINN-PI uses gradient descent (for solving a non-convex optimization problem). ELM-PI is more computationally efficient (see Table 1), but PINN-PI can handle higher-dimensional systems effectively (due to the expressiveness of deeper networks; see Section 5.3).*

3.4. Loss term to ensure local stability is preserved across iterations

Based on our observations, training optimal controllers for high-dimensional systems remains *extremely* challenging. In fact, most state-of-the-art reinforcement learning algorithms, whether model-free or model-based, struggle to solve the benchmark control problems we chose with stability guarantees, even in a small region around the equilibrium point to be stabilized (such as cartpole and quadrotors).

Through our extensive testing (see Table 2 in Appendix E), we found that ELM-PI excels in solving low-dimensional problems with high accuracy and fast solver time. However, PINN-PI scales better with state dimensions. Hence, we focus on the PINN-PI algorithm for high-dimensional control problems.

When naively implemented, PINN-PI can also lead to unstable controllers. This is because the loss function (16) does not capture the stabilization requirement of the resulting controller

$$\hat{\kappa}_{i+1}(x) = -\frac{1}{2}R^{-1}g^T(x)(DV_{i,\text{NN}}(x))^T. \tag{17}$$

To overcome this issue, we draw inspiration from classical control theory. When $f(x) = Ax$ and $g(x) = B$, the exact-PI algorithm is nothing but a sequence of Lyapunov equations that can be used to iteratively solve the algebraic Riccati equation. Given the assumptions on f and g, locally (1) is approximated by $\dot{x} = Ax + Bu$, where $A = Df(0)$ and $g(0) = B$.

We examine the linear approximation of $\hat{\kappa}$ and quadratic approximation of $V_{i,\text{NN}}(x)$ around the origin. Assume $\nabla^2 Q(0) = \hat{Q} > 0$ and let $\hat{R} = R(0)$. Furthermore, suppose that, for each $i \geq 0$, the controller $\hat{k}_i$ is exponentially stabilizing and denote $\hat{K}_i := D\hat{\kappa}_i(0)$. Write $\hat{A}_i = A + B\hat{K}_i$. Since $\hat{A}_i$ is Hurwitz, by linear system theory, there exists a quadratic function $\hat{P}$ that solves the Lyapunov equation

$$\hat{P}_i\hat{A}_i + \hat{A}_i^T\hat{P}_i = -\hat{Q} - \hat{K}_i^T\hat{R}\hat{K}_i. \tag{18}$$

Comparing this with (9), we expect the quadratic part of $V_{i,\text{NN}}(x)$ to be approximated by $x^T\hat{P}x$ near the origin and the next neural controller $\hat{\kappa}_{i+1}(x)$ is well approximated by a linear controller $\hat{K}_{i+1}x$ near the origin with

$$\hat{K}_{i+1} = -R^{-1}B^T\hat{P}_i, \tag{19}$$

which is precisely the gain update required for policy improvement for linear systems. Hence we expect that

$$\hat{K}_{i+1} = D\hat{\kappa}_{i+1}(0). \tag{20}$$

In view of (17), this can be easily encoded as a loss term

$$\left|\frac{\partial}{\partial x}(-\frac{1}{2}R^{-1}g^T(x)(DV_{i,\text{NN}}(x))^T)\big|_{x=0} - \hat{K}_{i+1}\right|_{\text{F}} \tag{21}$$

where $|\cdot|_{\mathrm{F}}$ is the Frobenius norm and K_{i+1} is solved by (19) and (18). This loss term plays a significant role in stabilizing the training process of PINN-PI for high-dimensional systems.

3.5. Verification of stability via neural Lyapunov functions

Upon termination of Algorithms 1 or 2, we obtain an approximation $\hat{V}(x)$ of the optimal value function. A corresponding approximate optimal control is given by

$$u = \hat{\kappa}(x) = \begin{cases} -\frac{1}{2}R^{-1}g^T(x)(D\hat{V}(x))^T, & \text{if } x \neq \mathbf{0}; \\ \mathbf{0} \text{ otherwise.} \end{cases} \tag{22}$$

For either Algorithms 1 or 2, $D\hat{V}$ can be readily computed, and is a function involving nonlinear activation functions and possible compositions of them when using multi-layer neural networks in Algorithm 2.

Suppose that the algorithms terminate perfectly as in the exact-PI case, we have $\hat{V}_{i+1}(x) = \hat{V}_i(x) = V(x)$ and $\hat{\kappa}_{i+1}(x) = \hat{\kappa}_i(x) = \kappa(x)$. We obtain from (9) that

$$\begin{aligned} &DV(x)(f + g(x)\kappa(x)) \\ =& - Q(x) - \kappa^T(x)R(x)\kappa(x) < 0, \quad x \neq \mathbf{0}, \end{aligned} \tag{23}$$

provided that $Q(x)$ is positive definite. However, because of the use of function approximators, we cannot obtain (23). Instead, we use a satisfiability modulo theories (SMT) solver (Gao et al., 2013) to verify the following nonlinear inequality

$$D\hat{V}(x)(f + g(x)\hat{\kappa}(x)) \leq -\mu, \quad x \in \Omega \setminus U_\varepsilon, \tag{24}$$

where U_ε is a small neighborhood around the origin of radius $\varepsilon > 0$, and $\mu > 0$ is a small constant. While checking the exact satisfaction of inequality (23) is in general undecidable, there exist delta-complete SMT solvers (Gao et al., 2013) that can either verify the inequality or falsify a δ-weakened version of it, where $\delta > 0$ can be any arbitrary precision parameter. To use such tools, it is necessary to exclude a small neighborhood of the origin (Chang et al., 2019; Zhou et al., 2022), as (23) turns into an equation at the origin. It is worth noting that, with additional assumptions, one may be able to verify exact stability including the origin through examination of the derivatives of the vector fields (Liu et al., 2023b). In this paper, we verify the stability given the controllers generated from Algorithms 1 and 2 using (24) for a U_ε with some small $\varepsilon > 0$, which ensures that solutions are attracted to any prescribed small neighborhood of the origin. Note that, by the continuity (or smoothness) of the approximators, for sufficiently small $\varepsilon > 0$, we can also have

$$D\hat{V}(x)(f+g(x)\hat{\kappa}(x)) \leq -\mu+\mathcal{O}(\varepsilon) < 0, \quad x \in U_\varepsilon \setminus \{\mathbf{0}\}, \tag{25}$$

where $\mathcal{O}(\varepsilon) \to 0$ as $\varepsilon \to 0$. Assuming that a suitable value for ε can be chosen, such that both (24) and (25) hold, the stability can be verified using neural Lyapunov functions.

4. Convergence analysis

We state the main regularity and convergence results in this section. The proofs can be found in Appendix C.

4.1. Convergence analysis for exact-PI

In view of Proposition 2.4, we expect that each policy evaluation in exact-PI has a unique solution so that the algorithm eventually yields a meaningful outcome. We first establish that each GHJB in exact-PI possesses a unique viscosity solution characterized by a specific pattern.

Proposition 4.1. *Let $u \in \mathcal{U}$ be any (autonomous) state feedback controller so that there exists some feedback policy $\kappa : \mathbb{R}^n \to \mathbb{R}^m$ such that $u(\cdot) = \kappa(\phi(\cdot))$. Then, the equation $-G(x, u, DV(x)) = 0$ (which captures the transient transitions) has a unique positive definite viscosity solution within the space $C(\Omega) \cap C^1(\Omega \setminus \{\mathbf{0}\})$.*

Corollary 4.2. *For each $i \geq 0$, the GHJB $G(x, \kappa_i(x), DV_i(x)) = 0$ has a unique positive definite viscosity solution V_i, which belongs to $C(\Omega) \cap C^1(\Omega \setminus \{\mathbf{0}\})$.*

Remark 4.3. *Note that in the situation where the state feedback controller u is not necessarily stabilizing, but (1)(2)(4) of Definition 2.1 still hold and $f(x) + g(x)u$ has countable zeros, the above existence and uniqueness of viscosity solution still follow. However, this discussion is beyond the scope of this paper.* □

The following theorem states that exact-PI converges to the true solution to (8). The proof can be found in Appendix C. This result plays an important role in analyzing the convergence of neural approximations in the next subsection.

Theorem 4.4 (Convergence of Successive Approximations of Viscosity Solution)**.** *For each $i \geq 0$, let $u_i(\cdot) = \kappa_i(\phi(\cdot))$, where κ_i is defined in (10). Suppose that $u_0 \in \mathcal{U}$, then,*

(1) $u_i \in \mathcal{U}$ for all $i \in \{0, 1, \cdots\}$.

(2) $V^* \leq V_{i+1} \leq V_i$ for all $x \in \Omega$ and for all $i \in \{0, 1, \cdots\}$, where V_i is the viscosity solution to $G(x, \kappa_i(x), DV_i(x)) = 0$ and V^* is the viscosity solution to (8).

(3) $V_i \to V^*$ uniformly on Ω as $i \to \infty$ given the compactness of Ω.

4.2. Convergence analysis for policy iteration using neural approximations

The main idea is to formalize properties of the loss function that capture the desired convergence of neural approximations to true solutions, which in this context are the viscosity

solutions to the GHJB and HJB equations. We expect that when the training error (or the loss function in (14) or (16)) is small, the generalization error is also small. In practice, this requires that the number of collocation points chosen from Ω, at which the residual of each GHJB G is evaluated, be sufficiently large.

However, based on Corollary 4.2, the viscosity solution for each iteration does not exhibit uniform differentiability across the entire domain of Ω. In addition, most convergence results for data-driven methods are typically based on a compact subset of the state space. Therefore, direct consideration of C^1-uniform convergence on $\Omega \setminus \{\mathbf{0}\}$ is not feasible. Instead, we achieve the C^1-uniform convergence on $\Omega \setminus U_\varepsilon$ and a weaker convergence on U_ε (asymptotically as $\varepsilon \to 0$), where U_ε is some open set centered at $\mathbf{0}$ of arbitrarily small radius $\varepsilon > 0$.

To circumvent complex notation, let us consider the general case for any GHJB $G(x, \kappa(x), DV(x)) = 0$ with admissible κ as in Proposition 4.1 to illustrate the idea. Focusing on $\Omega \setminus U_\varepsilon$, we consider the space of continuously differentiable functions $\mathcal{G} = C^1(\Omega \setminus U_\varepsilon, \mathbb{R})$ equipped with the C^1-uniform norm, $|V|_{C^1} := \sup_{x\in\Omega\setminus U_\varepsilon} |V(x)| + \sup_{x\in\Omega\setminus U_\varepsilon} |DV(x)|$. We consider a training error $E_{T,N} : \mathcal{G} \to [0, \infty)$ of the following form (e.g. the loss function in (14) and (16)),

$$E_{T,N}(V) = \frac{1}{N}\sum_{k=1}^{N} |G(x_k, \kappa(x_k), DV(x_k))|^2 + |V(\mathbf{0})|^2 ,$$

where $N \in \mathbb{N}$ is associated with the number of collocation points chosen from Ω. We seek approximations $\{\hat{V}_N\}_{N\in\mathbb{N}} \subseteq \mathcal{F}$ of the unique viscosity solution V in some function space $\mathcal{F} \subseteq \mathcal{G}$, for instance, the space of functions representable by a one hidden-layer network. Then, we aim to determine whether $E_{T,N}(\hat{V}_N) \to 0$ implies $\hat{V}_N \to V$ in $\mathcal{G}$.

Continuing the above settings, in the following proposition, we state that by incorporating additional assumptions, a convergence result can be obtained on $\Omega \setminus U_\varepsilon$.

Hypothesis 4.5. *For any Lipschitz continuous function h and its smooth neural approximations $\{\hat{h}_N\}_{N\in\mathbb{N}}$, the Lipschitz constant of $\hat{h}_N$ converges to the true Lipschitz constant as $\frac{1}{N}\sum_{k=1}^{N} \left|\hat{h}(x_k) - h(x_k)\right|^2$ converges to 0.*

Remark 4.6. *This phenomenon has been thoroughly investigated by (Khromov & Singh, 2023). For low-dimensional systems, it is possible to also directly penalize the Lipschitz constant of the residual and achieve higher accuracy (see the proof of Proposition 4.7 for details). In contrast, when the Lipschitz constant of the residual is difficult to verify, it is reasonable to assume Hypothesis 4.5.* □

Proposition 4.7. *Let $\varepsilon > 0$ any arbitrarily small number and U_ε be an open set centered at $\mathbf{0}$ of radius ε. Let $\mathcal{F} \subseteq \mathcal{G}$ be a subspace with uniformly bounded Lipschitz constant on $\Omega \setminus U_\varepsilon$. Suppose that $\{x_k\}_{k\in\mathbb{N}}$ is a sequence dense on $\Omega \setminus U_\varepsilon$ with the additional requirement that, for all $N \in \mathbb{N}$, $\delta_N = \inf\left\{\delta > 0 : \Omega \setminus U_\varepsilon \subseteq \bigcup_{k=1}^{N} \mathcal{B}_\delta^{\Omega\setminus U_\varepsilon}(x_k)\right\}$ and $C = \sup\left\{N\mu(\mathcal{B}_{\delta_n}^{\Omega\setminus U_\varepsilon}(x_1)) : N \in \mathbb{N}\right\} < \infty$ where μ is the Lebesgue measure and $\mathcal{B}_\delta^{\Omega\setminus U_\varepsilon}(x)$ is the open ball of radius $\delta > 0$ centered at x in $\Omega \setminus U_\varepsilon$.*

Suppose that Hypothesis 4.5 holds and the training error $E_{T,N}(\hat{V}_N)$ can be arbitrarily small for sufficiently large N. Then, the neural network $\hat{V}_N \to V$ in $\mathcal{G}$.

Remark 4.8. *The additional requirement on $\{x_k\}_{k\in\mathbb{N}}$ indicates that the smallest volume of the "finite coverings" on $\Omega \setminus U_\varepsilon$ is finite. As a technical matter, instead of using the dense set $\{x_k\}$, we can use a sequence of finite sets $A_N := \{x_k\}_{k=1}^{N}$ (as in Algorithm 2) that is "eventually dense" in $\mathbb{R}^n$, i.e. $A_N \to X$ in the Hausdorff metric. This would allow one to use new training points, as long as the finite sets are good approximations of $\mathbb{R}^n$.* □

By patching the above sound approximation on $\Omega \setminus U_\varepsilon$ for any small $\varepsilon > 0$ and the asymptotic approximation within U_ε, one can obtain the following convergence guarantee.

Theorem 4.9. *Given κ_0, let $\{V_i\}$ and $\{\kappa_{i+1}\}$ be updated by exact-PI. Let $\{\hat{V}_i\}$ and $\{\hat{\kappa}_{i+1}\}$ be updated by ELM-PI or PINN-PI with $\hat{\kappa}_0 = \kappa_0$. Let the conditions in Proposition 4.7 hold. Then, for any $i \geq 0$ and $\vartheta > 0$, we can choose a sufficiently dense set of collocation points $\{x_k\}_{k=1}^{N}$ such that*

$$|\hat{V}_i(x) - V_i(x)| \leq \vartheta,\ |\hat{\kappa}_{i+1}(x) - \kappa_{i+1}(x)| \leq \vartheta,\ \ x \in \Omega.$$

Remark 4.10. *We would like the readers to be aware that the approximating neural network sequence may not exhibit certain equicontinuity, implying that the limit may not be the (unique) true solution within the same function space. Instead of merely stating that convergence can be guaranteed when the training error is arbitrarily small, we have explicitly demonstrated in the proof how the difference between $\hat{V}_i$ and V_i is proportional to the training error. The theoretical results provide the conditions under which the demonstrated convergence can occur, ensuring that any further effort to minimize the training error is not in vain.* □

5. Numerical experiments

In this section, we present numerical examples to evaluate the performance of the proposed algorithms. We aim to accomplish three goals: 1) Evaluate the performance characteristics of ELM-PI and PINN-PI, ranging from low to high-dimensional systems; 2) Compare with approaches in classical control literature and demonstrate the superior performance of ELM-PI in solving low-dimensional systems and highlight the importance of formal verification; 3)

Table 1: Comparison of ELM-PI, PINN-PI, and SGA on the inverted pendulum example: we run ELM-PI and PINN-PI with $m = 50$, $m = 100$, $m = 200$, and $m = 400$ and SGA (Beard et al., 1997) with polynomial bases of order 2, 4, 6, 8. We record the training/computational time and whether the resulting controller is verifiably stabilizing.

SGA			ELM-PI			PINN-PI		
Order	Time (s)	Verified?	m	Time (s)	Verified?	m	Time (s)	Verified?
2	4.80	Yes	50	**0.11**	Yes	50	255.15	Yes
4	19.37	Yes	100	**0.24**	Yes	100	256.53	Yes
6	66.52	Yes	200	**0.71**	Yes	200	258.89	Yes
8	212.42	Yes	400	**2.92**	Yes	400	256.52	Yes

Demonstrate the superior capabilities of PINN-PI in solving high-dimensional benchmark control problems and compare it with state-of-the-art model-free and model-based reinforcement learning algorithms. The code of these experiments can be found at https://git.uwaterloo.ca/hybrid-systems-lab/lyznet/.

5.1. Synthetic n-dimensional nonlinear control

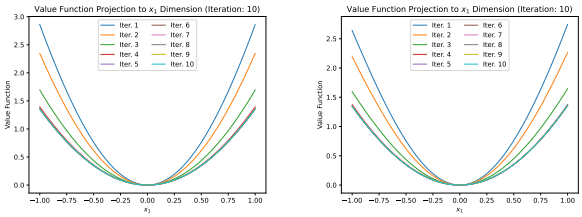
Consider the nonlinear control problem given by $f_i(x) = x_i^3 + u_i$, where $i = 1, 2, \ldots, n$. The cost is defined by $Q(x) = \sum_{i=1}^n (x_i^2 + 2x_i^4)$ and $R = I_n$. From optimal control theory, the optimal value function is obtained by solving the HJB equation, giving $V^*(x) = \sum_{i=1}^n \left(\frac{1}{2}x_i^4 + \frac{1}{2}(x_i^2+1)^2 - \frac{1}{2}\right)$. We run ELM-PI and PINN-PI and compare their performance for various dimensions n in terms of computational time and maximum testing error relative to the true optimal value function. The results are summarized in Table 2 in Appendix E.

From the experiments, we see that for low-dimensional problems ($n \leq 3$), ELM-PI outperforms PINN-PI in terms of both computational efficiency and approximation accuracy. As the dimension increases, more computational units (m) are required for ELM-PI to achieve a higher accuracy. For example when $n = 4$, to achieve 10^{-3} accuracy, it requires $m = 3200$, and for 10^{-5}, $m = 6400$. The complexity of solving linear least square is $O(mN^2)$, provided that $N > m$. In our experiments, we set $N = d * m$. Hence, the time complexity is $O(d^2m^3)$, which roughly captures the increase in computational time reported in Table 2 as m and d increase. For $n \geq 5$, it is evident that ELM-PI becomes inefficient. In comparison, PINN-PI can achieve 10^{-2} to 10^{-3} accuracy across all dimensions within a reasonable amount of computational time. In fact, we were able to obtain 10^{-2} error with $m = 800$ across all dimensions. The accuracy, however, does not seem to improve significantly as m, N, or the number of steps increase in training PINN-PI.

Based on these evaluations, we recommend to use ELM-PI for low-dimensional problems and PINN-PI for high-dimensional problems.

5.2. Inverted pendulum and comparison with successive Galerkin approximations

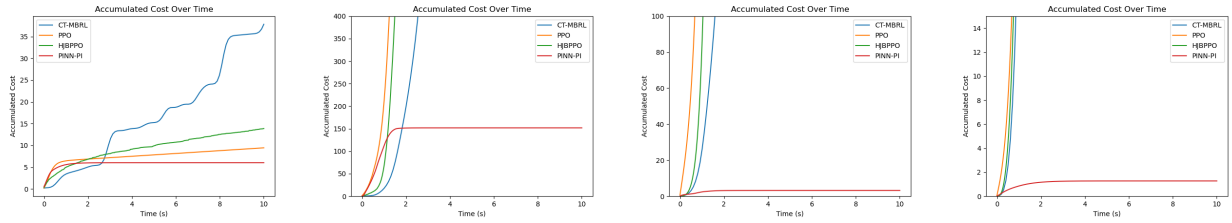
We run both ELM-PI and PINN-PI to compute the optimal control and policy for the inverted pendulum (see Section E.3 in the Appendix for more details). Figure 1 displays the results of implementing ELM-PI on an inverted pendulum. The value functions, projected onto x_1, are plotted for each iteration. The left panel represents the scenario with $m = 100$ and $N = 200$, while the right panel corresponds to $m = 50$ and $N = 100$. Despite the visual similarity and apparent convergence after five iterations, it is surprising that the controller obtained from $m = 50$ does not actually stabilize the system. Conversely, we have verified that the controller derived from $m = 100$ is indeed stabilizing using dReal (Gao et al., 2013). This example demonstrates the justification for employing formal verification alongside policy iteration to attain both optimality and stability, particularly in safety-critical scenarios.



(a) $m = 100,\ N = 200$ (b) $m = 50,\ N = 200$

Figure 1: ELM-PI on inverted pendulum: despite visual similarity and apparent convergence, the controller obtained from $m = 50$ fails to stabilize the system, while the one from $m = 100$ can be formally verified to be stabilizing.

We also compare ELM-PI and PINN-PI with successive Galerkin approximations (SGA) for solving GHJB (Beard et al., 1997) and provide further verification results. All



(a) Inverted Pendulum (b) Cartpole (c) 2D Quadrotor (d) 3D Quadrotor

Figure 2: Plots of accumulated costs over time for the four environments

algorithms are run with 10 iterations. The results are summarized in Table 1. While SGA can also effectively solve low-dimensional problems, ELM-PI is significantly superior in terms of solver time. To demonstrate the performance of the obtained controllers, we simulate trajectories resulting from different controllers with random initial conditions with the same random seed for different controllers) and record the cost averaged over 50 trajectories in Figure 3 in Appendix E.3. It can be seen that high-order SGA achieves the same cost as ELM-PI with different m values. PINN-PI achieves similar costs but requires longer training time as expected. Furthermore, as demonstrated in the next section, PINN-PI can solve high-dimensional problems that are beyond the reach of SGA.

5.3. Comparison with reinforcement learning algorithms

We compare PINN-PI against well-established reinforcement learning (RL) algorithms, including Proximal Policy Optimization (PPO) (Schulman et al., 2017), Hamilton Jacobi Bellman PPO (HJBPPO) (Mukherjee & Liu, 2023), and Continuous Time Model-Based Reinforcement Learning (CT-MBRL) (Yildiz et al., 2021). We train each algorithm in benchmark control environments (inverted pendulum, cartpole, 2D quadrotor, and 3D quadrotor) and compare their control costs.

PPO is a model-free actor-critic algorithm that uses a clipped objective function to limit policy updates, ensuring incremental learning steps. It consists of an actor-network $\pi_\theta(a|s)$ that takes the state s as input and outputs a distribution over actions a, and a value network $V_\phi(s)$ that takes the state s as input and outputs the expected return. HJBPPO is an extension of PPO that uses the continuous-time HJB equation as a loss function, instead of the discrete-time Bellman optimality equation. CT-MBRL introduces a model-based approach, employing continuous-time dynamics for more precise control and prediction in RL tasks. While we are aware there are other RL algorithms available in the literature, the rationale for choosing these RL algorithms for comparison is given in Section F.1 in the Appendix.

We compare PINN-PI with three reinforcement learning (RL) algorithms in Figure 2 by comparing their accumulated control costs over time. As shown in the plots, PINN-PI significantly outperforms the rest of the algorithms in the inverted pendulum environment and the remaining higher-dimensional environments. Furthermore, the simulated trajectories from different initial conditions under the learned optimal controllers are shown in Figure 5 in Appendix F.

While the RL algorithms exhibit comparable performance to PI-Policy in the inverted pendulum environment, they struggle to achieve stability in higher-dimensional environments, such as the cartpole and quadrotor, leading to their accumulated control cost diverging. PINN-PI, on the other hand, demonstrates convergence to equilibrium in less than two seconds, marking a significant improvement. This may appear striking at first, but being able to encode local asymptotic stability as a loss (21) in training plays an important role in achieving an asymptotically stabilizing optimal controller, whereas other RL algorithms typically use episodic training over a finite time horizon. This discrepancy explains the superior performance of PINN-PI in problems where asymptotic stability is closely tied to the performance criteria (think of LQR as a special case).

6. Conclusions

We propose two algorithms for conducting model-based policy iterations to solve nonlinear optimal control problems. The first algorithm leverages the linearity of the PDE that defines the policy value and utilizes linear least squares to obtain the approximation. This approach proves to be highly efficient and accurate for low-dimensional problems. The second approach employs physics-informed neural networks and demonstrates better scalability for high-dimensional problems. We emphasize the importance of incorporating formal verification on top of policy iterations to achieve both optimality and stability when safety is a concern. We provide theoretical analysis that shows policy iterations (both exact and approximate) converge to the true optimal solutions in general settings. Limitations of this work and potential future work are discussed in Appendix G.

Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canada Research Chairs program. This research was enabled in part by computing resources provided by the Digital Research Alliance of Canada (alliance.ca).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bardi, M., Dolcetta, I. C., et al. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer, 1997.

Beard, R. W. *Improving the closed-loop performance of nonlinear systems*. PhD thesis, Rensselaer Polytechnic Institute, 1995.

Beard, R. W., Saridis, G. N., and Wen, J. T. Galerkin approximations of the generalized hamilton-jacobi-bellman equation. *Automatica*, 33(12):2159–2177, 1997.

Beard, R. W., Saridis, G. N., and Wen, J. T. Approximate solutions to the time-invariant Hamilton–Jacobi–Bellman equation. *Journal of Optimization theory and Applications*, 96:589–626, 1998.

Bellman, R. E. *Dynamic Programming*. Princeton University Press, 1957.

Bertsekas, D. P. *Dynamic Programming and Optimal Control: Volume I*, volume 1. Athena Scientific, 2012.

Bertsekas, D. P. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE transactions on neural networks and learning systems*, 28(3): 500–509, 2015.

Bertsekas, D. P. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

Bhasin, S., Kamalapurkar, R., Johnson, M., Vamvoudakis, K. G., Lewis, F. L., and Dixon, W. E. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica*, 49(1):82–92, 2013.

Camilli, F., Grüne, L., and Wirth, F. A generalization of zubov's method to perturbed systems. *SIAM Journal on Control and Optimization*, 40(2):496–515, 2001.

Chang, Y.-C., Roohi, N., and Gao, S. Neural Lyapunov control. *Advances in Neural Information Processing Systems*, 32, 2019.

Chen, J., Chi, X., Yang, Z., et al. Bridging traditional and machine learning-based algorithms for solving PDEs: The random feature method. *arXiv preprint arXiv:2207.13380*, 2022.

Crandall, M. G., Evans, L. C., and Lions, P.-L. Some properties of viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 282 (2):487–502, 1984.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Evans, L. C. *Partial Differential Equations*, volume 19. American Mathematical Society, 2010.

Farsi, M. and Liu, J. *Model-Based Reinforcement Learning: From Data to Continuous Actions*. Wiley-IEEE Press, 2023.

Furfaro, R., D'Ambrosio, A., Schiassi, E., and Scorsoglio, A. Physics-informed neural networks for closed-loop guidance and control in aerospace systems. In *AIAA SCITECH 2022 Forum*, pp. 0361, 2022.

Gao, S., Kong, S., and Clarke, E. M. dreal: An smt solver for nonlinear theories over the reals. In *Proceedings of 24th International Conference on Automated Deduction*, pp. 208–214. Springer, 2013.

Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510, 2018.

Howard, R. A. *Dynamic Programming and Markov Process*. MIT Press, 1960.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, 70 (1-3):489–501, 2006.

Jiang, Y. and Jiang, Z.-P. Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica*, 48(10):2699–2704, 2012.

Jiang, Y. and Jiang, Z.-P. Robust adaptive dynamic programming and feedback stabilization of nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):882–893, 2014.

Jiang, Y. and Jiang, Z.-P. *Robust Adaptive Dynamic Programming*. John Wiley & Sons, 2017.

Khromov, G. and Singh, S. P. Some fundamental aspects about lipschitz continuity of neural network functions. *arXiv preprint arXiv:2302.10886*, 2023.

Kleinman, D. On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1):114–115, 1968.

Leake, R. and Liu, R.-W. Construction of suboptimal control sequences. *SIAM Journal on Control*, 5(1):54–63, 1967.

Liu, J., Meng, Y., Fitzsimmons, M., and Zhou, R. Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification. *arXiv preprint arXiv:2312.09131*, 2023a.

Liu, J., Meng, Y., Fitzsimmons, M., and Zhou, R. Towards learning and verifying maximal neural Lyapunov functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 8012–8019. IEEE, 2023b.

Meng, Y., Zhou, R., and Liu, J. Learning regions of attraction in unknown dynamical systems via zubov-koopman lifting: Regularities and convergence. *arXiv preprint arXiv:2311.15119*, 2023.

Milshtein, G. N. On an iterative technique for Riccati equation computations. *Automation and Remote Control*, 25 (3):298–306, 1964.

Mukherjee, A. and Liu, J. Bridging physics-informed neural networks with reinforcement learning: Hamilton-Jacobi-Bellman proximal policy optimization (HJBPPO). *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Saridis, G. N. and Lee, C.-S. G. An approximation theory of optimal control for trainable manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(3):152–159, 1979.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Vaisbord, E. M. Concerning an approximate method for optimum control synthesis. *Avtomatika i Telemekhanika*, 24(12):1626–1632, 1963.

Vrabie, D. and Lewis, F. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks*, 22 (3):237–246, 2009.

Weinan, E., Han, J., and Jentzen, A. Algorithms for solving high dimensional PDEs: from nonlinear monte carlo to machine learning. *Nonlinearity*, 35(1):278, 2021.

Yildiz, C., Heinonen, M., and Lähdesmäki, H. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

Zhou, R., Quartz, T., De Sterck, H., and Liu, J. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. *Advances in Neural Information Processing Systems*, 2022.

A. Basic properties of viscosity solutions

The definition of viscosity solutions is given below.

Definition A.1. *Define the superdifferential and the subdifferential sets of V at x respectively as*

$$\partial^+ V(x) = \left\{ p \in \mathbb{R}^n : \limsup_{y \to x} \frac{V(y) - V(x) - p \cdot (y - x)}{|y - x|} \leq 0 \right\}, \tag{26a}$$

$$\partial^- V(x) = \left\{ q \in \mathbb{R}^n : \liminf_{y \to x} \frac{V(y) - V(x) - q \cdot (y - x)}{|y - x|} \geq 0 \right\}. \tag{26b}$$

A continuous function V of a PDE of the form $F(x, V(x), DV(x)) = 0$ (possibly encoded with boundary conditions) is a viscosity solution if the following conditions are satisfied:

(1) (viscosity subsolution) $F(x, V(x), p) \leq 0$ for all $x \in \mathbb{R}^n$ and for all $p \in \partial^+ V(x)$.

(2) (viscosity supersolution) $F(x, V(x), q) \geq 0$ for all $x \in \mathbb{R}^n$ and for all $q \in \partial^- V(x)$.

The following lemma (Lemma 1.7, Lemma 1.8, Chapter I, Bardi et al., 1997) provides some insights on $\partial^+ V(x)$ and $\partial^+ V(x)$ for some $V \in C(\Omega)$.

Lemma A.2 (Sub- and Supperdifferential)**.** *Let $V \in C(\Omega)$. Then*

(1) $p \in \partial^+ V(x)$ if and only if there exists $\psi \in C^1(\Omega)$ such that $D\psi(x) = p$ and $u - \psi$ has a local maximum at x;

(2) $q \in \partial^- V(x)$ if and only if there exists $\psi \in C^1(\Omega)$ such that $D\psi(x) = q$ and $u - \psi$ has a local minimum at x;

(3) if for some x both $\partial^+ V(x)$ and $\partial^- V(x)$ are nonempty, then $\partial^+ V(x) = \partial^- V(x) = \{DV(x)\}$;

(4) the sets $\{x \in \Omega : \partial^+ V(x) \neq \emptyset\}$ and $\{x \in \Omega : \partial^- V(x) \neq \emptyset\}$ are dense.

In view of (1) and (2) in Lemma A.2, (1) and (2) in Definition A.1 are equivalent as

(1) for any $\psi \in C^1$, if x is a local maximum for $V - \psi$, then $F(x, \psi(x), d\psi(x)) \leq 0$;

(2) for any $\psi \in C^1$, if x is a local minimum for $V - \psi$, then $F(x, \psi(x), d\psi(x)) \geq 0$.

Definition A.3. *Suppose V is locally Lipschitz. Define the classical upper and lower Dini (directional) derivatives, respectively, as*

$$D^+ V(x; p) = \limsup_{t \to 0^+} \frac{V(x + tp) - V(x)}{t}, \tag{27a}$$

$$D^- V(x; q) = \liminf_{t \to 0^+} \frac{V(x + tq) - V(x)}{t}. \tag{27b}$$

The following theorem (Theorem 2.40, Chapter III, Bardi et al., 1997) states the equivalence of Dini solutions to viscosity solutions to GHJBs. We rephrase the theorem as follows. We omit the proof due to the similarity.

Theorem A.4. *Suppose Ω is a bounded open set, and $V \in C(\bar{\Omega})$. For GHJB $G(x, u, DV(x)) = 0$ with a fixed $u \in \mathcal{U}$, the following statements are equivalent:*

(1) $-G(x, u, p) \leq 0$ for all $x \in \mathbb{R}^n$ and for all $p \in \partial^+ V(x)$ (respectively, ≥ 0);

(2) $-L(x, u) - D^+ u(x; f(x) + g(x)u) \leq 0$ for all $x \in \mathbb{R}^n$ (respectively, ≥ 0).

B. Proofs in Section 2

Proof of Proposition 2.3: Note that for all $t > 0$ and $u \in \mathcal{U}$, we have

$$\begin{aligned} J(x,u) &= \int_0^t L(\phi(s;x,u),u(s))ds + \int_t^\infty L(\phi(s;x,u),u(s))ds \\ &= \int_0^t L(\phi(s;x,u),u(s))ds + \int_0^\infty L(\phi(s+t;x,u),u(s+t))ds \\ &= \int_0^t L(\phi(s;x,u),u(s))ds + J(\phi(t;x,u),u') \\ &\geq \inf_{u\in\mathcal{U}} \left\{ \int_0^t L(\phi(s;x,u))ds + V(\phi(t;x,u)) \right\}, \end{aligned}$$

where the controller u' is defined as $u'(s) := u(s+t)$ for all $s > 0$. Taking the infimum over $\mathcal{U}$, we have

$$V(x) \geq \inf_{u\in\mathcal{U}} \left\{ \int_0^t L(\phi(s;x,u),u(s))ds + V(\phi(t;x,u)) \right\}.$$

Not we fix a $u \in \mathcal{U}$, an $\varepsilon > 0$, and choose a $u' \in \mathcal{U}$ such that

$$V(\phi(t;x,u)) \geq J(\phi(t;x,u),u') - \varepsilon.$$

Let the controller u'' be such that

$$u''(s) = \begin{cases} u(s), & s \leq t, \\ u'(s-t), & s > t. \end{cases}$$

Then,

$$\begin{aligned} V(x) \leq & J(x,u'') \\ &= \int_0^t L(\phi(s;x,u),u(s))ds + \int_t^\infty L(\phi(s;x,u''),u''(s))ds \\ &= \int_0^t L(\phi(s;x,u),u(s))ds + \int_0^\infty L(\phi(s;\phi(t;x,u),u'),u'(s))ds \\ &\leq \int_0^t L(\phi(s;x,u),u(s))ds + V(\phi(t;x,u)) + \varepsilon. \end{aligned}$$

Since u and ε are given arbitrarily, by sending $\varepsilon \to 0$ and taking the infimum over $\mathcal{U}$, we have

$$V(x) \leq \inf_{u\in\mathcal{U}} \left\{ \int_0^t L(\phi(s;x,u),u(s))ds + V(\phi(t;x,u)) \right\},$$

which completes the proof. □

Proof of Proposition 2.4: We first show that V is a viscosity solution using the equivalent conditions introduced in Appendix A. Let $\psi \in C^1$ and x be a local maximum point of $V - \psi$. Then

$$V(x) - V(z) \geq \psi(x) - \psi(z), \quad \forall z \in \mathcal{B}(x,r),$$

where $\mathcal{B}(x,r)$ denotes the set $\{z \in \mathbb{R}^n : |z-x| < r\}$. Consider any constant control signal u. For t sufficiently small, we have $\phi(t;x,u) \in \mathcal{B}(x,r)$. Therefore,

$$\begin{aligned} \psi(x) - \psi(\phi(t;x,u)) \leq & V(x) - V(\phi(t;x,u)) \\ &\leq \int_0^t L(\phi(s;x,u),u(s))ds + V(\phi(t;x,u)) - V(\phi(t;x,u)), \end{aligned} \tag{28}$$

where the second line is in virtue of Proposition 2.3. Considering the infinitesimal transition behavior (as $t \downarrow 0$) on both sides of (28), we have

$$-D\psi(x) \cdot (f(x) + g(x)u) \leq L(x, u),$$

which implies that $H(x, D\psi(x)) \leq 0$.

Now we verify the case when x is a local minimum of $V - \psi$. For each $\varepsilon > 0$ and $t > 0$, by the second part of Proposition 2.3, there exists a $u'' \in \mathcal{U}$ such that

$$\begin{aligned} V(x) &\geq \int_0^t L(\phi(s; x, u''), u''(s))ds + V(\phi(t; x, u)) - t\varepsilon \\ &\geq \int_0^t L(x, u'')ds + V(\phi(t; x, u)) - t\varepsilon + \mathcal{O}(t), \end{aligned}$$

where the second line is by the Lipschitz continuity of L and ϕ, and $\mathcal{O}(t)/t \to 0$ as $t \to 0$. Therefore, by the local minimum property of $V - \psi$,

$$\begin{aligned} \psi(x) - \psi(\phi(t; x, u'')) &\geq V(x) - V(\phi(t; x, u'')) \\ &\geq \int_0^t L(x, u'')ds - t\varepsilon + \mathcal{O}(t). \end{aligned}$$

Note that u'' is selected based on some arbitrary ε and t. Now one can use a similar infinitesimal argument as the first part and obtain $H(x, D\psi(x)) \geq 0$.

It can be easily verified that $|H(x, 0)| \leq M$ for some constant M, and, for every r, there exists an L_r such that $|H(x, p) - H(y, p) \leq L_r|x - y|(1 + |p|)$ for $|x|, |y| \leq r$. The uniqueness argument follows (Theorem 1, Section 10.2, Evans, 2010), (Section VI.3, Bardi et al., 1997), and (Camilli et al., 2001, Section 3). □

Proof of Theorem 2.5: Let $x_i := \phi(t_i; x, \kappa)$ and $p_i \in \partial^+ V(x_i)$. Then, by Lemma A.2, there exists a $\psi_i \in C^1$ such that $D\psi_i(x_i) = p_i$, $V(x_i) = \psi_i(x_i)$, and $V \leq \psi$ in the neighborhood of x_i. Since $\{x \in \Omega : \partial^+ V(x) \neq \emptyset\}$ is dense, setting $t_0 = 0$ and $t_k \to \infty$ as $k \to \infty$, we can patch up J in the following sense given that $t_i - t_{i-1} > 0$ is sufficient small for all $i \in \{1, 2, \cdots\}$:

$$\begin{aligned} J(x, \kappa(\phi(\cdot))) &= \lim_{k\to\infty} \left\{ \sum_{i=0}^{k-1} \int_{t_i}^{t_{i+1}} L(\phi(s; \phi(t_i; x, \kappa), \kappa(\phi(s))ds + J(\phi(t_k; x, \kappa), \kappa) \right\} \\ &\geq \psi(x) - \lim_{k\to\infty} \psi(\phi(t_k; x, \kappa)) \geq V(x). \end{aligned} \tag{29}$$

Note that in (29), the first inequality is obtained in a similar way as in (28), except that the ψ_i's are indexed, and the intermediate ψ_i's cancel each other out. The reason that the sequence of inequalities, as in (28), works out is due to the autonomous system's 'Markov' property, i.e., whenever we halt the trajectory and start fresh, the new trajectory has no dependence on the past.

The opposite inequality can be shown in a similar manner. □

C. Proofs in Section 4

C.1. Results in Section 4.1

Proof of Proposition 4.1: The existence and uniqueness of ϕ follows by the basic assumptions on (1). Now we define

$$V(x) = \int_0^\infty L(\phi(s; x, u), u(\phi(s; x, u)))ds. \tag{30}$$

Then V is positive definite, and, for all $t > 0$, we have

$$\begin{aligned} V(x) &= \int_0^t L(\phi(s;x,u),u(\phi(s;x,u))ds + \int_t^\infty L(\phi(s;x,u),u(\phi(s;x,u))ds \\ &= \int_0^t L(\phi(s;x,u),u(\phi(s;x,u))ds + \int_0^\infty L(\phi(s;\phi(t;x,u),u),u(\phi(s;\phi(t;x,u),u))ds \\ &= \int_0^t L(\phi(s;x,u),u(\phi(s))ds + V(\phi(t;x,u)). \end{aligned}$$

To verify V is a viscosity solution, we use a similar method as in the proof of Proposition 2.4. Let $\psi \in C^1$ and x be a local maximum point of $V - \psi$. Then, for t sufficiently small, we have $\phi(t;x,u) \in \mathcal{B}(x,r)$. Therefore,

$$\begin{aligned} \psi(x) - \psi(\phi(t;x,u)) \leq & V(x) - V(\phi(t;x,u)) \\ = & \int_0^t L(\phi(s;x,u),u(\phi(s))ds + V(\phi(t;x,u)) - V(\phi(t;x,u)). \end{aligned} \tag{31}$$

Considering the infinitesimal behavior on both sides of (31), we have

$$-D\psi(x)\cdot(f(x)+g(x)u) \leq L(x,u),$$

which implies that $G(x,u,D\psi(x)) \leq 0$. The other side of the comparison falls in the same procedure.

To validate the uniqueness, we notice that for a stabilizing state feedback control $u = \kappa(x)$, the Lipschitz continuous function $f(x) + g(x)\kappa(x)$ can only have a zero at $\mathbf{0}$. Given that $V(0) = 0$, and suppose that $\Omega \subseteq \mathbb{R}$, the uniqueness is followed by (Liu et al., 2023a, Proposition 2) and (Meng et al., 2023, Theorem 19). In addition, we notice that the quantity $DV(x) = -L(x,\kappa(x))/(f(x)+g(x)\kappa(x))$ is differentiable continuous solution other than 0. For higher dimensional cases, we address the problem using the well-known method of characteristics. The solvability of C^1 solution depends on the non-singularity of the Jacobian matrix, which is only problematic at $\mathbf{0}$. By the continuity of viscosity solution, V is uniquely defined on Ω. □

Remark C.1. *During the policy iteration, we cannot guarantee an everywhere C^1 property of the solutions to GHJBs. Revisiting the example in Section 3.1, the corresponding GHJB is given by $x^2 + DV(x)\cdot(xu) = 0$. One can simply initialize with an admissible controller $u = \kappa_0(x) = -\frac{1}{2}|x|$. Then, $V_0(x) = 2|x|$ uniquely solves the GHJB only in the viscosity sense, which fails to be everywhere C^1 in the first iteration. In view of the last part of the above proof, the non-differentiable points are only decided by the zeros of $f(x) + g(x)\kappa(x)$. In addition, to resolve Remark 4.3, one can simply follow the exact argument.*

Proof of Theorem 4.4:

(1) By Proposition 4.1, the function $V \in C^1(\Omega \setminus \mathbf{0}) \cap C(\Omega)$. Apart from $\mathbf{0}$, the exact-PI provides an admissible controller for each iteration, which follows the exact procedure as in (Lemma 5.2.4, Beard, 1995). At $\mathbf{0}$, by exact-PI, the state feedback controller returns a $\mathbf{0}$. In addition, the upper and lower Dini derivatives $D^+V(x;p)$ and $D^-V(x;p)$ for any p exist and are bounded. The Lipschitz continuity of u_i at $\mathbf{0}$ also follows by the definition.

(2) Note that, at differentiable points, the proof follows exactly as (Theorem 3.1.4, Jiang & Jiang, 2017). However, in line with the concept of viscosity solutions, we provide a general proof. To begin with, along the trajectory subject to the controller u_{i+1}, we have, for each x,

$$\begin{aligned} & V_{i+1} - V_i \\ \leq & \int_0^\infty D^+(V_{i+1} - V_i)(\phi(s); f(\phi(s)) + g(\phi(s))u_{i+1})ds \\ \leq & \int_0^\infty (D^+V_{i+1} - D^-V_i)(\phi(s); f(\phi(s))ds + \int_0^\infty g(\phi(s))u_{i+1})ds, \end{aligned} \tag{32}$$

where the existence of Dini derivatives are granted by the Lipschitz continuity of V_i and V_{i+1}. On the other hand, V_i and V_{i+1} are, respectively, the unique viscosity solution to $G(x,u_i,DV_i)$ and $G(x,u_{i+1},DV_{i+1}) = 0$. By Theorem

A.4, and plugging the directional Dini derivative $D^+V_{i+1}(\phi(s); f(\phi(s)) + g(\phi(s))u_{i+1})$ and $D^-V_i(\phi(s); f(\phi(s)) + g(\phi(s))u_i$ into (32), one can obtain that

$$\begin{aligned} & V_{i+1}(x) - V_i(x) \\ \leq & -\int_0^\infty \|u_i\|_{\mathrm{R}}^2 + \|u_{i+1}\|_{\mathrm{R}}^2 - 2u_{i+1}^T R u_i ds \\ \leq & -\int_0^\infty \|u_{i+1} - u_i\|_{\mathrm{R}}^2 \, ds \leq 0. \end{aligned} \tag{33}$$

(3) It is a well-known result that a monotonic sequence of functions $\{V_i\}_{i\geq 0}$ that is bounded from below converges pointwise to a function V_∞. In view of Dini's theorem (see also the proof of (Theorem 5.3.1, Beard, 1995)), the sequence also converges uniformly provided that Ω is compact. In addition, it can be verified that $\{V_i\}$ has a uniformly bounded Lipschitz constant on Ω and forms a compact subspace in $C(\Omega)$, which implies the Lipschitz continuity of V_∞. The pointwise convergence of $\{u_i\}_{i\geq 0}$ also follows (Theorem 3.1.4, Jiang & Jiang, 2017).

It suffices to show that V_∞ is a viscosity solution to (8). On $\Omega \setminus \{\mathbf{0}\}$, DV_∞ exists uniquely almost everywhere (a.e.) and is Lebesgue integrable. Based on the definition of G_i as well as the pointwise convergence of $\{u_i\}$, we have that DV_i is Lebesgue integrable for each i and converges pointwise. Combining the fact that $V_i \to V_\infty$ uniformly, we have $\lim_{i\to\infty} DV_i = DV_\infty$ a.e. by Radon-Nikodym theorem. However, it can be verified that $\lim_{i\to\infty} DV_i$ solves (8) pointwise on $\Omega \setminus \{\mathbf{0}\}$. It follows that DV_∞ solves (8) a.e. on $\Omega \setminus \{\mathbf{0}\}$. At $\mathbf{0}$, we have that $V_\infty(\mathbf{0}) = \lim_{i\to\infty} V_i(\mathbf{0}) = 0$, which may not be differentiable. However, it is clear that $H(x, p)$ is convex in p for each fixed x, and $H(x, DV_\infty(x)) = 0$ a.e. on Ω. By (Proposition 5.2, Chapter II, Bardi et al., 1997), V_∞ is a viscosity solution on Ω. In virtue of Proposition 2.4, V_∞ should also be the unique viscosity solution. Therefore, $V_\infty = V^*$. □

Remark C.2. *It is worth noting that (3) of (Theorem 3.1.4, Jiang & Jiang, 2017) is based on the assumptions $V^*, V_\infty \in C^1(\Omega)$. However, both assumptions are not necessarily guaranteed. As pointed out in Section 3.1, the HJB $-x^2 + \frac{1}{4}(DV(x))^2x^2 = 0$ has a unique viscosity solution $V^*(x) = 2|x|$, which fails to be differentiable at 0. As for V_∞, it is the limit of $\{V_i\}$ only w.r.t. the uniform norm rather than the C^1-norm. The C^1 property of V_∞ on $\Omega \setminus \{\mathbf{0}\}$ is not even guaranteed.*

The convergence of $DV_i \to DV_\infty$ (if exists) is also not clear in (Theorem 3.1.4, Jiang & Jiang, 2017) and (Theorem 5.3.1, Beard, 1995) relying only on the uniform convergence of $\{V_i\}$. To ensure that V_∞ solves (8), the work (Theorem 3.2, Farsi & Liu, 2023) made a strong assumption on the uniform convergence of $\{DV_i\}$, which cannot be guaranteed in practice. In this view, it is necessary to consider the convergence and solutions in the viscosity sense for general cases. The above proof takes advantage of the convexity of $H(x, p)$ in p such that only the property of $\lim_{i\to\infty} DV_i = DV_\infty$ a.e. is needed. □

C.2. Results in Section 4.2

In order to ensure the coherence of the proofs within this subsection, we recall the following notation. Given κ_0, $\{V_i\}$ and $\{\kappa_{i+1}\}$ are updated by exact-PI. In other words, for each $i \geq 0$, V_i is the unique viscosity solution to

$$\begin{aligned} & G(x, \kappa_i, DV_i(x)) \\ =& Q(x) + \|\kappa_i(x)\|_{\mathrm{R}}^2 + DV_i(x) \cdot (f(x) + g(x)\kappa_i(x)) \\ =& 0, \end{aligned} \tag{34}$$

and κ_{i+1} is updated by (10). In addition, $\{\hat{V}_i\}$ and $\{\hat{\kappa}_{i+1}\}$ are updated by PINN-PI with $\hat{\kappa}_0 = \kappa_0$. For each $i \geq 0$, we also denote $\tilde{V}_i$ as the true viscosity solutions to

$$\begin{aligned} & G(x, \hat{\kappa}_i, D\tilde{V}_i(x)) \\ =& Q(x) + \|\hat{\kappa}_i(x)\|_{\mathrm{R}}^2 + D\tilde{V}_i(x) \cdot (f(x) + g(x)\hat{\kappa}_i(x)) \\ =& 0. \end{aligned} \tag{35}$$

Accordingly, we also set $\tilde{\kappa}_{i+1}(x) = -\frac{1}{2}R^{-1}(x)g(x)D\tilde{V}_i(x)$ if $x \neq \mathbf{0}$, and $\tilde{\kappa}_{i+1}(\mathbf{0}) = \mathbf{0}$.

Before proving Theorem 4.9, we look at a lemma that entails the expected convergence result.

Lemma C.3. *Suppose for each $i \geq 0$, we have*

$$\sup_{x\in\Omega}\left|\hat{V}_i(x)-\tilde{V}_i(x)\right|+\sup_{x\in\Omega}|\hat{\kappa}_{i+1}(x)-\tilde{\kappa}_{i+1}(x)|\to 0 \tag{36}$$

Then, PINN-PI can guarantee that

$$\sup_{x\in\Omega}\left|\hat{V}_i(x)-V_i(x)\right|+\sup_{x\in\Omega}|\hat{\kappa}_{i+1}(x)-\kappa_{i+1}(x)|\to 0. \tag{37}$$

Proof. We prove the convergence by induction. For $i=0$, we have $\hat{\kappa}_0=\kappa_0$. Then $V_0=\tilde{V}_0$ by Corollary 4.2. By the definition of κ_1 and $\tilde{\kappa}_1$, it follows that $\kappa_1=\tilde{\kappa}_1$. We can train the neural network sufficiently well, such that $\sup_{x\in\Omega}\left|\hat{V}_0(x)-\tilde{V}_0(x)\right|+\sup_{x\in\Omega}|\hat{\kappa}_1(x)-\tilde{\kappa}_1(x)|\to 0$, which immediately implies

$$\sup_{x\in\Omega}\left|\hat{V}_0(x)-V_0(x)\right|+\sup_{x\in\Omega}|\hat{\kappa}_1(x)-\kappa_1(x)|\to 0.$$

For each $i\geq 1$, let $W_i=V_i-\tilde{V}_i$. Then, $W_i\in C^1(\Omega\setminus\{\mathbf{0}\}))$, $DW_i\in C(\Omega\setminus\{\mathbf{0}\}))$, and $W_i(\mathbf{0})=0$. Since on $\Omega\setminus\{\mathbf{0}\}$, V_i and $\tilde{V}_i$ solves (34) and (35) in the conventional sense. A direct comparison of (34) and (35) gives that

$$\begin{aligned}&DW_i(x)\cdot(f(x)+g(x)\hat{\kappa}_i(x))\\=&-g^T(x)DV_i(x)\cdot(\hat{\kappa}_i(x)-\kappa_i(x)))-\|\hat{\kappa}_i(x)\|_{\mathrm{R}}^2+\|\kappa_i(x)\|_{\mathrm{R}}^2.\end{aligned}$$

However, $g^TDV_i=-2R\kappa_i+2R(\kappa_i-\kappa_{i+1})$. Therefore, on $\Omega\setminus\{\mathbf{0}\}$,

$$\begin{aligned}&DW_i(x)\cdot(f(x)+g(x)\hat{\kappa}_i(x))\\=&2R(x)\kappa_i(\hat{\kappa}_i(x)-\kappa(x))-2R(x)(\kappa_i(x)-\kappa_{i+1}(x))(\hat{\kappa}_i(x)-\kappa_i(x))-\hat{\kappa}_i^T(x)R(x)\hat{\kappa}_i(x)+\kappa_i^T(x)R(x)\kappa_i(x)\\=&-\|\hat{\kappa}_i(x)-\kappa_i(x)\|_{\mathrm{R}}^2-2R(x)(\kappa_i(x)-\kappa_{i+1}(x))(\hat{\kappa}_i(x)-\kappa_i(x)),\end{aligned}$$

and

$$\begin{aligned}&|DW_i(x)\cdot(f(x)+g(x)\hat{\kappa}_i(x))|\\\leq&\|\hat{\kappa}_i(x)-\kappa_i(x)\|_{\mathrm{R}}^2+2\|R(x)\|\|\kappa_i(x)-\kappa_{i+1}(x)\|\hat{\kappa}_i(x)-\kappa_i(x)|.\end{aligned} \tag{38}$$

For simplicity, we define an intermediate Hamiltonian

$$\begin{aligned}&F_i(x,DW_i(x))\\:=&|DW_i(x)\cdot(f(x)+g(x)\hat{\kappa}_i(x))|-\|\hat{\kappa}_i(x)-\kappa_i(x)\|_R^2-2\|R\|\|\kappa_i(x)-\kappa_{i+1}(x)\|\hat{\kappa}_i(x)-\kappa_i(x)|.\end{aligned} \tag{39}$$

Note that for each $i\geq 1$, the mapping $p\mapsto F_i(x,p)$ is convex for any fixed x. By (Proposition 5.1, Chapter II, Bardi et al., 1997), W_i is the viscosity solution to $F_i(x,DW_i(x))=0$ on Ω. Given that $\hat{\kappa}_i$ converges to κ_i uniformly, applying (Proposition 2.2, Chapter II, Bardi et al., 1997), W_i should uniformly converges (on Ω) to the viscosity solution of

$$|DW_i(x)|\cdot(f(x)+g(x)\hat{\kappa}_i(x))=0,$$

which is the constant 0. As a byproduct, due to the continuous differentiability on $\Omega\setminus\{\mathbf{0}\}$, one can check that $D\tilde{V}_i\to DV_i$ (or $|DW_i|\to 0$) uniformly on $\Omega\setminus\{\mathbf{0}\}$. However, by the definition of $\tilde{\kappa}_i$ and κ_i again, we have $\tilde{\kappa}_{i+1}\to\kappa_{i+1}$ in the same sense on Ω. By the hypothesis (36), and a triangle inequality argument, the uniform convergence in (37) follows. □

Remark C.4. *In the proof, leveraging the convexity of $F_i(x,DW_i(x))$ in $DW_i(x)$, the result in (Proposition 2.2, Chapter II, Bardi et al., 1997) ensures that the uniform convergence $|\tilde{V}_i-V_i|$ in C^1 norm on $\Omega\setminus\{\mathbf{0}\}$ (almost everywhere) can lead to a weaker convergence on Ω (everywhere):*

$$\sup_{x\in\Omega}\left|\tilde{V}_i(x)-V_i(x)\right|+\sup_{x\in\Omega}|\tilde{\kappa}_{i+1}(x)-\kappa_{i+1}(x)|\to 0. \tag{40}$$

In other words, we sacrifice the exact convergence of $|D\tilde{V}_i-DV_i|$ at $\mathbf{0}$ (due to the potential lack of differentiability), and use the convergence of $\sup_{x\in\Omega}|\tilde{\kappa}_{i+1}(x)-\kappa_{i+1}(x)|$ instead. Note that, the mapping g in the definition of $\tilde{\kappa}_{i+1}$ and κ_i has a smoothing effect. And this is also the reason why we can achieve the desired convergence property on the entire Ω. □

Proof of Proposition 4.7: Recall that this proposition considers the general case for any GHJB $G(x, \kappa(x), DV(x)) = 0$ with admissible κ as in Proposition 4.1. Given that $f(x) + g(x)\kappa(x)$ is bounded from above and away from $\mathbf{0}$ on $\Omega \setminus U_\varepsilon$, it can be easily shown that DV of the true solution V is also Lipschitz continuous. We first introduce short hand notations $|\cdot|_\infty := \sup_{x\in\Omega\setminus U_\varepsilon} |\cdot|$ and $\|\cdot\|_2 := \int_{\Omega\setminus U_\varepsilon} |\cdot|^2 dx$ for functions. For any Lipschitz continuous function h on $\Omega \setminus U_\varepsilon$, we define the Lipschitz constant as

$$\mathrm{Lip}(h) := \sup_{x\neq y} \frac{|h(x) - h(y)|}{|x - y|}.$$

For any $\hat{V}_N \in \mathcal{F}$, let $\mathcal{R}_N(x) := G(x, \kappa(x), D\hat{V}_N(x))$. It is clear that $\mathcal{R}_N$ is Lipschitz continuous by the definition of G.
Step 1: We first show some useful bounds. Note that, given the compactness of $\Omega \setminus U_\varepsilon$ and the continuous differentiability of $\hat{V}_N$, we have

$$\begin{aligned} |\hat{V}_N|_{C^1} &= |\hat{V}_N|_\infty + |D\hat{V}_N|_\infty \\ &\leq C_1 \cdot |D\hat{V}_N|_\infty + |\hat{V}_N(\mathbf{0})|, \end{aligned} \tag{41}$$

where $C_1 = \sup_{x\in\Omega\setminus U_\varepsilon} |x| + 1$. In addition, since $f(x) + g(x)\kappa(x)$ is bounded from above and away from $\mathbf{0}$ on $\Omega \setminus U_\varepsilon$, it can be verified that

$$\begin{aligned} C_2 \cdot |D\hat{V}_N|_\infty &\leq \sup_{x\in\Omega\setminus U_\varepsilon} |L(x, \kappa(x)) + \mathcal{R}_N(x)| \\ &\leq C_3 \cdot |DV_N|_\infty, \end{aligned} \tag{42}$$

where $C_2 = \inf_{x\in\Omega\setminus U_\varepsilon} |\min\{f(x) + g(x)\kappa(x)\}|$ and $C_3 = |f(x) + g(x)\kappa(x)|_\infty$.

Now we show the bound for $|\mathcal{R}_N(x)|_\infty$. Let x^*, x_* be the maximizer and minimizer for $|\mathcal{R}_N(x)|$, respectively. Then

$$\begin{aligned} &\int_{\Omega\setminus U_\varepsilon} |\mathcal{R}_N(x)|^2 dx \\ \geq &\mu(\Omega \setminus U_\varepsilon) \cdot |\mathcal{R}_N(x_*)|^2 \\ = &\mu(\Omega \setminus U_\varepsilon) \cdot |\mathcal{R}_N(x_*) - \mathcal{R}_N(x^*) + \mathcal{R}_N(x^*)|^2 \\ \geq &\mu(\Omega \setminus U_\varepsilon) \cdot (-2|\mathcal{R}_N(x_*) - \mathcal{R}_N(x^*)||\mathcal{R}_N(x^*)| + \mathcal{R}_N^2(x^*)), \end{aligned}$$

and consequently,

$$\begin{aligned} &|\mathcal{R}_N(x)|_\infty^2 \\ \leq &\left(\frac{1}{\mu(\Omega \setminus U_\varepsilon)} \|\mathcal{R}_N\|_2^2\right) + 2|\mathcal{R}_N(x_*) - \mathcal{R}_N(x^*)||\mathcal{R}_N(x^*)| \\ \leq &\left(\frac{1}{\mu(\Omega \setminus U_\varepsilon)} \|\mathcal{R}_N\|_2^2\right) + 4\,\mathrm{Lip}^2(\mathcal{R}_N) \cdot \sup_{x\in\Omega\setminus U_\varepsilon} |x|. \end{aligned}$$

This implies that there exists a $C_4 > 0$ such that

$$|\mathcal{R}_N|_\infty \leq C_4(\|\mathcal{R}_N\|_2 + \mathrm{Lip}(\mathcal{R}_N)). \tag{43}$$

Step 2: We show the continuous dependence of $\|\mathcal{R}_N\|_2$ on $E_{T,N}(\hat{V}_N)$. Define

$$\left\{\hat{V}_N \in \mathcal{F} : \mathrm{Lip}(\mathcal{R}_N) + \left|\hat{V}_N(\mathbf{0})\right| < r\right\} =: \mathcal{F}_r,$$

which is uniformly equicontinuous and hence compact. Pick $\vartheta > 0$. By the uniform continuity of G, there is a $\delta > 0$ such that for all $\hat{V}_N \in \mathcal{F}_r$ and every $x, z \in X$ with $|x - z| < \delta$, we have

$$|\mathcal{R}_N(x) - \mathcal{R}_N(y)| < \min\left\{\sqrt{\frac{\vartheta}{3}}, \frac{\vartheta}{6M_{\mathcal{F}_r}}\right\}$$

where $M_{\mathcal{F}_r}$ is a uniform upper bound on $|\mathcal{R}_N(x)|$ for $\hat{V}_N \in \bar{\mathcal{F}}_r$ and $x \in \Omega \setminus U_\varepsilon$ (this bound exists by compactness of $\bar{\mathcal{F}}_r$ and continuity of G).

For this δ, we can pick an $N_1 \in \mathbb{N}$ so that $\delta_{N_1} < \delta$. It follows that for all $N \geq N_1$, we have $\Omega \setminus U_\varepsilon \subseteq \bigcup_{k=1}^N \overline{\mathcal{B}^X_{\delta_N}(x_k)}$ and $\delta_N < \delta$. By Hypothesis 4.5 and the assumption that $E_{T,N}(\hat{V}_N) \to 0$, there is a $N_2 \in \mathbb{N}$ with for all $n \geq N_2$ we have $E^{\text{mod}}_{T,N}(\hat{V}_N) < \frac{\vartheta}{3}$, where

$$E^{\text{mod}}_{T,N}(\hat{V}_N) = \frac{1}{N}\sum_{k=1}^N |\mathcal{R}_N(x_k)|^2 + |\operatorname{Lip}(\mathcal{R}_N)| + \left|\hat{V}(\mathbf{0})\right|.$$

Then, for all $N \geq \max\{N_1, N_2\}$

$$\begin{aligned}
\|\mathcal{R}_N\|_2 &\leq \int_{x \in \bigcup_{k=1}^N \mathcal{B}^{\Omega\setminus U_\varepsilon}_{\delta_N}(x_k)} |\mathcal{R}_N(x)|^2\, dx \\
&\leq \sum_{k=1}^N \int_{x \in \mathcal{B}^{\Omega\setminus U_\varepsilon}_{\delta_N}(x_k)} |\mathcal{R}_N(x)|^2\, dx \\
&\leq \sum_{k=1}^N \mu\left(\mathcal{B}^{\Omega\setminus U_\varepsilon}_{\delta_N}(x_k)\right) |\mathcal{R}_N(x_k^*)|^2
\end{aligned}$$

where $x_k^* \in \mathcal{B}^{\Omega\setminus U_\varepsilon}_{\delta_N}(x_k)$ is the maximizer of $|\mathcal{R}_N(x)|$. Let $\mu_N := \mu\Big(\mathcal{B}^{\Omega\setminus U_\varepsilon}_{\delta_N}(x_k)\Big)$. Then, by uniform continuity of $\mathcal{R}_N$, we see

$$\begin{aligned}
\sum_{k=1}^N \mu_N |\mathcal{R}_N(x_k^*)|^2 \leq& \mu_N \sum_{k=1}^N \left(|\mathcal{R}_N(x_k^*) - \mathcal{R}_N(x_k)| + |\mathcal{R}_N(x_k)|\right)^2 \\
<& \mu_N \sum_{k=1}^N \left(\min\left\{\sqrt{\frac{\vartheta}{3}}, \frac{\vartheta}{6M_{\mathcal{F}_r}}\right\} + |\mathcal{R}_N(x_k)|\right)^2 \\
\leq& \mu_N \sum_{k=1}^N \left(\frac{\vartheta}{3} + \frac{\vartheta}{3}\frac{|\mathcal{R}_N(x_k)|}{M_{\mathcal{F}_r}} + |\mathcal{R}_N(x_k)|_2^2\right) \\
\leq& \mu_N \sum_{k=1}^N \left(\frac{\vartheta}{3} + \frac{\vartheta}{3}\right) + \mu_N \sum_{k=1}^N |\mathcal{R}_N(x_k)|_2^2 \\
\leq& \vartheta C_5.
\end{aligned}$$

where $C_5 := N\mu_N$. This completes the proof of Step 2.

Step 3: Now we are ready to prove the statement in this proposition. For simplicity, we write $a \lesssim b$ if there exists a constant $C > 0$, independent of a and b, such that $a \leq Cb$.

We pick sufficiently large N, M, then, by Eq. (41), (42), and (43),

$$\begin{aligned}
|\hat{V}_N - \hat{V}_M|_{C^1} &\lesssim |\mathcal{R}_N - \mathcal{R}_M|_\infty + |\hat{V}_N(\mathbf{0}) - \hat{V}_M(\mathbf{0})| \\
&\lesssim \|\mathcal{R}_N - \mathcal{R}_M\|_2 + \operatorname{Lip}(\mathcal{R}_N - \mathcal{R}_M) + |\hat{V}_N(\mathbf{0}) - \hat{V}_M(\mathbf{0})|.
\end{aligned}$$

In addition, given that $E_{T,N}(\hat{V}_N)$ can be arbitrarily small, by Step 2 and Hypothesis 4.5, it is clear that $\{\hat{V}_N\}_N$ forms a Cauchy sequence. We denote the limit as $\hat{V}^* \in \mathcal{F}$, which solves the equation $G(x, \kappa(x), D\hat{V}^*(x)) = 0$. By the uniqueness of the solution of G as in Proposition 4.1, one can conclude that $\hat{V}^*(x) = V(x)$.

Proof of Theorem 4.9: For any $\varepsilon > 0$ and for each i, by Proposition 4.7, one can guarantee a C^1-uniform convergence of $\hat{V}_i$ to $\tilde{V}_i$ on the compact set $\Omega \setminus U_\varepsilon$, where $\hat{V}_i$ plays the role of the representable functions in Proposition 4.7. In view of Remark C.4, one can obtain the convergence

$$\sup_{x\in\Omega\setminus U_\varepsilon} \left|\hat{V}_i(x) - \tilde{V}_i(x)\right| + \sup_{x\in\Omega\setminus U_\varepsilon} |\hat{\kappa}_{i+1}(x) - \tilde{\kappa}_{i+1}(x)| \to 0, \tag{44}$$

with a modified convergence region $\Omega \setminus U_\varepsilon$ instead of Ω.

On $U_\varepsilon \setminus \{\mathbf{0}\}$, by the continuous differentiability of $\hat{V}_i$, $\tilde{V}_i$, and g, one can immediately verify that

$$|\hat{V}_i(x) - \tilde{V}_i(x)| \leq \sup_{x \in U_\varepsilon} (|D\hat{V}_i(x)| + |D\tilde{V}_i(x)|) \cdot |x| \tag{45}$$

and

$$|\hat{\kappa}_{i+1}(x) - \tilde{\kappa}_{i+1}(x)| \leq C|x|, \tag{46}$$

where

$$C = \sup_{x \in U_\varepsilon \setminus \{\mathbf{0}\}} |Dg^T(x)| \cdot \sup_{x \in U_\varepsilon \setminus \{\mathbf{0}\}} (|D\hat{V}_i(x)| + |D\tilde{V}_i(x)|)$$

and $Dg^T(x)$ is the Fréchet derivative at x. Note that the quantities $\hat{V}_i$, $\tilde{V}_i$, $\hat{\kappa}_{i+1}$, and $\tilde{\kappa}_{i+1}$ are all null values at $\mathbf{0}$. In addition, since $\varepsilon > 0$ is arbitrarily small, combining Eqs. (44), (45), (46), one can have the arbitrarily smallness of $\sup_{x \in \Omega} \left|\hat{V}_i(x) - \tilde{V}_i(x)\right| + \sup_{x \in \Omega} |\hat{\kappa}_{i+1}(x) - \tilde{\kappa}_{i+1}(x)|$. By Lemma C.3, the statement in Theorem 4.9 follows immediately. □

D. Further details on verification of Lyapunov conditions

We aim to verify that a value function returned by Algorithms 1 or 2 satisfies the Lyapunov condition (24), recalled here as

$$DV(x)(f + g(x)\kappa(x)) \leq -\mu, \quad x \in \Omega \setminus U_\varepsilon. \tag{47}$$

To obtain more information, we can verify a slightly stronger set of conditions defined below to facilitate the claim for asymptotic attraction and region of attraction:

1. $DV(x)(f + g(x)\kappa(x)) \leq -\mu, \quad \forall x \in \{x \in \Omega : c_1 \leq V(x) \leq c_2\}$;
2. $\{x \in \Omega : V(x) \leq c_2\} \cap \partial\Omega = \emptyset$, where $\partial\Omega$ is the boundary of Ω;
3. $\{x \in \Omega : V(x) \leq c_1\} \subseteq U_\varepsilon$.

By these conditions, V will decrease along solutions of the closed-loop system under control $u = \kappa(x)$ and cannot escape Ω if solutions start in $\Omega_{c_2} := \{x \in \Omega : V(x) \leq c_2\}$. Furthermore, these solutions eventually reach the set $\Omega_{c_1} = \{x \in \Omega : V(x) \leq c_1\}$, which is contained in B_ε. While (12) appears to be a weaker condition than the above set of three conditions, if V is positive definite on Ω with respect to the origin, then, for any $\varepsilon > 0$, we can always choose c_1 and c_2 such that the above conditions hold. Verifying these conditions readily gives a region of attraction Ω_{c_2} and an attractive set Ω_{c_1}.

E. Numerical experiments

E.1. Implementation details

All instances of ELM-PI and PINN-PI are run with the tanh activation function, unless otherwise noted. The tanh activation function is effective in approximating smooth functions. If non-smooth functions are involved, ReLU activation might be preferred (see Section F.3.1).

The same as in Section 5 and Table 2, we run all examples with $N = m * d$, where m is the size of the network, and d is the dimension of the problem. The iteration number of PI is set to be 10. We only implemented a one-layer network for PINN-PI to draw fair comparisons with ELM-PI and basis function approaches such as successive Galerkin approximations (see Section E.3).

For both ELM-PI and PINN-PI, the number of iterations in PI is set to be 10. For PINN-PI, we train the network for 10,000 steps in each iteration with Adam. For $m \leq 1,600$, we conduct three separate runs of the experiments to report the average maximum testing errors and runtimes. For $m \geq 3,200$, the computational time is significantly larger and reported for a single run. ELM-PI experiments were run with an Intel Gold 6148 Skylake @ 2.4 GHz, and PINN-PI experiments were run with an NVidia V100SXM2 (16G memory). The errors reported are testing errors on $2 * N$ points, where N is the number of collocation points.

E.2. Verification of synthetic n-dimensional nonlinear control

The domain on which we solve the problem is set to be $\Omega = [-1, 1]^n$. Recall that the optimal value function is given by

$$V^*(x) = \sum_{i=1}^{n} \left(\frac{1}{2} x_i^4 + \frac{1}{2}(x_i^2 + 1)^2 - \frac{1}{2} \right).$$

The detailed comparisons of training ELM-PI and PINN-PI on this synthetic example are shown in Table 2.

We also conducted formal verification experiments on this example. It is easy to verify that the largest level set of V^* contained in Ω is $\{V^* \leq 2\}$. In other words, we should be able to verify conditions (1)–(3) in Section D with any $0 < c_1 < c_2 < 2$. We verified value functions returned by ELM-PI and PINN-PI against conditions (1)–(3). The results are summarized in Table 3. The parameters used for verification are $\mu = 1e-4$, $c_1 = 0.01$, $c_2 = 1.99$, and $\varepsilon = 0.1$. We employed dReal (Gao et al., 2013) for verification. It is evident from the results that dReal, which utilizes interval analysis, falls prey to the curse of dimensionality. We also noted an intriguing observation that the value functions returned by ELM-PI, despite possessing the same form and number of neurons as those from PINN-PI, appear to pose a greater challenge for verification by dReal. This peculiar observation currently lacks a robust explanation, and it may pertain to the implementation of the dReal tool (Gao et al., 2013). On potential explanation is that the value function trained with ELM tends to have large coefficients, which may pose a challenge for verification with dReal. One may mitigate the issue by adding a L2 regularization term for the coefficients, at the expense of accuracy. We plan to investigate this issue further as well as alternative tools for verification in our future work.

E.3. Inverted pendulum and comparison with successive Gakerlin approximations

The dynamics of the inverted pendulum are described by $\ddot{\theta} = \frac{mg\ell \sin\theta - \mu\dot{\theta} + u}{m\ell^2}$, where u is the control input. We consider $\ell = 0.5$, $m = 0.1$, $g = 9.8$, $\mu = 0.1$. The cost function is defined by $Q = I_2$ and $R = 2$. We run both ELM-PI and PINN-PI to compute the optimal control and policy. In this case, we do not have the analytical expression of the optimal value function as the ground truth. We extract the resulting optimal controllers and plot trajectories from random initial conditions to show the stability and performance of the controllers.

Figure 3 depicts the simulated costs averaged over 50 trajectories. We also verified the stability of the resulting controllers. In all the cases, we are able to verify the Lyapunov stability conditions outlined in Section D with $c_1 = 0.01$ and $c_2 = 0.029$. While $c_2 = 0.029$ appears to be small, it indeed gives the largest level set of the optimal value function contained in the region of interest, as shown in Figure 4.

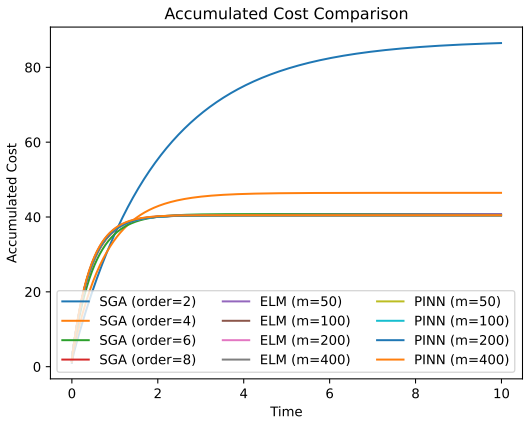


Figure 3: ELM-PI, PINN-PI, and SGA on the inverted pendulum example: it can be seen that the value returned by a high-order SGA achieves the same cost as ELM-PI with a different number of neurons, while the computational time required by ELM-PI is significantly less. In all the cases, we are able to verify the Lyapunov stability conditions outlined in Appendix D are met.

Furthermore, Figure 1 depicts the training results for ELM-PI with $m = 50$ and $m = 100$. While the plots appear similar,

Table 2: Performance of ELM-PI and PINN-PI on a synthetic n-dimensional nonlinear problem: Here, n represents the dimension of the problem, m denotes the number of hidden units used for approximation, and N indicates the number of collocation points.

Problem & model size			ELM-PI		PINN-PI	
n	m	N	Error	Time (s)	Error	Time (s)
1	50	50	1.54E-10	0.03	2.45E-03	236.69
1	100	100	1.14E-10	0.08	2.86E-03	240.79
1	200	200	1.25E-11	0.21	2.17E-03	284.02
2	50	100	1.26E-01	0.07	1.03E-02	237.31
2	100	200	8.75E-04	0.18	4.97E-03	273.05
2	200	400	8.37E-07	0.52	9.87E-03	336.04
2	400	800	1.79E-08	1.81	1.69E-02	560.14
2	800	1600	2.47E-09	25.94	3.10E-02	843.12
2	1600	3200	6.10E-10	184.98	1.53E-02	953.37
3	200	600	7.65E-02	0.82	1.95E-02	371.18
3	400	1200	7.66E-03	2.62	1.21E-02	762.42
3	800	1600	1.34E-04	33.77	1.48E-02	866.44
3	1600	4800	2.18E-06	284.53	2.55E-02	1034.34
3	3200	9600	2.58E-07	3593.61	2.81E-02	768.84
4	800	3200	2.02E-01	24.59	2.85E-02	903.55
4	1600	6400	3.29E-02	346.79	3.51E-02	1121.94
4	3200	12800	2.92E-03	4771.60	3.52E-02	984.04
4	6400	25600	4.35E-05	43052.25	4.39E-02	3304.76
5	800	4800	5.36E00	32.22	3.63E-02	770.54
5	3200	16000	3.08E-01	6303.20	5.20E-02	1204.22
5	6400	32000	6.37E-02	53479.10	9.10E-02	5906.69
6	800	4800	8.76E00	39.03	4.31E-02	800.76
6	6400	38400	2.33E00	63403.53	1.38E-01	6995.10
7	800	100000	–	–	3.74E-02	4414.28
8	800	100000	–	–	5.28E-02	4415.12
9	800	100000	–	–	4.88E-02	4422.54
10	800	100000	–	–	3.66E-02	4424.33
11	800	100000	–	–	8.29E-02	4424.00
12	800	100000	–	–	6.11E-02	4426.54

Table 3: Training and verification of ELM-PI and PINN-PI on an n-dimensional nonlinear control example: value functions returned by ELM-PI and PINN-PI are verified against the Lyapunov conditions in Section D. The experimental setup is the same as the results in Table 2. The experiments in this table were run on a MacBook Pro with a 2 GHz Quad-Core Intel Core i5. Note that the results may slightly vary due to the selection of a random seed. The symbol $\times$ indicates that verification by dReal (Gao et al., 2013) returned a counterexample for $c_1 = 0.01$ and $c_2 = 1.99$ and t.o. indicates verification was not conclusive within 1,800 (s).

Problem & model size		ELM-PI	PINN-PI
n	m	Verify (s)	Verify (s)
1	50	6.99	0.02
1	100	2.20	0.05
1	200	4.13	0.14
2	50	t.o.	$\times$
2	100	t.o.	0.68
2	200	t.o.	1.88
2	400	t.o.	6.10

the controller obtained from $m = 50$ fails to stabilize the system, while the one from $m = 100$ can be verified to be stabilizing using an SMT solver. This highlights the importance of formal verification to ensure stability guarantees.

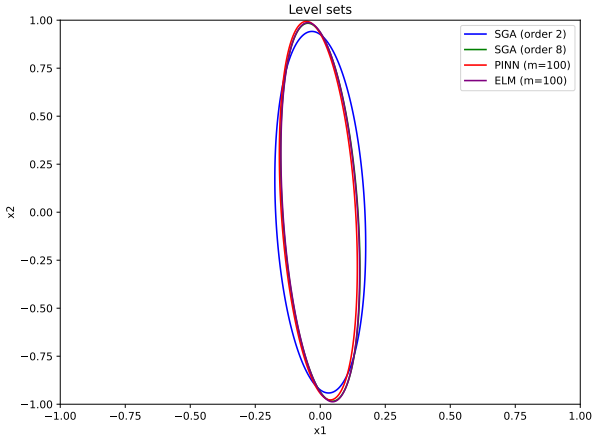
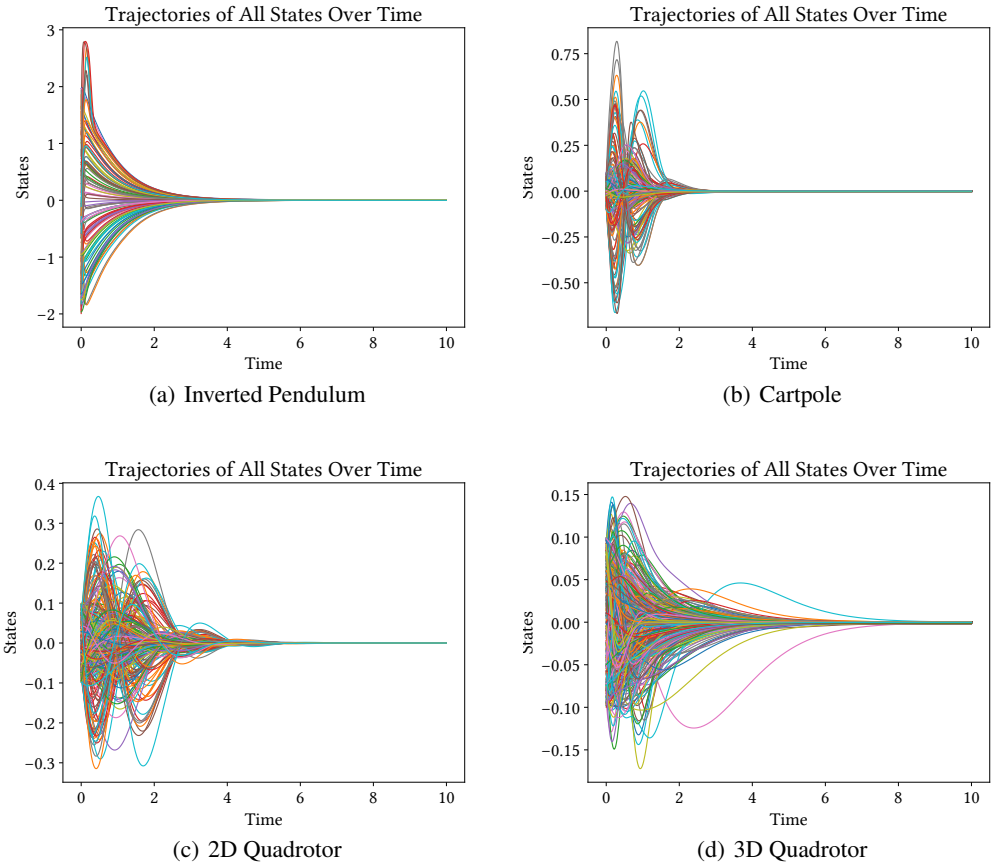


Figure 4: Certified regions of attraction by ELM-PI, PINN-PI, and SGA on the inverted pendulum example: it can be seen that for high-order SGA, PINN-PI, and ELM-PI, a region of attraction close to the boundary of the region of interest Ω can be verified using SMT solvers.

F. Comparison with reinforcement learning algorithms on benchmark nonlinear control problems

F.1. Rationale for algorithms selection in comparison

We chose three recent RL algorithms as benchmarks: two model-free ones, PPO and HJBPPO, and one model-based, CT-MBRL. We used the implementation of PPO from the stable-baselines3 library (Raffin et al., 2021), and we implemented HJBPPO by modifying this library. Since HJBPPO addresses the same problems as ours and incorporates the HJB equation to derive better policies than PPO, we compare our algorithm with both as model-free RL benchmarks. On the other hand, CT-MBRL is a state-of-the-art model-based RL algorithm that concerns similar environments to those in our numerical experiments.



(a) Inverted Pendulum (b) Cartpole

(c) 2D Quadrotor (d) 3D Quadrotor

Figure 5: Plots of trajectories starting from different initial conditions under the optimal controller learned using PINN-PI for the four environments. All trajectories converge to the origin.

It is worth mentioning that the ultimate goal of our algorithm is to devise optimal and stabilizing controllers for nonlinear systems after a few policy iterations, which can provide asymptotic stability for an infinite time horizon. This differs from the typical performance comparisons for RL algorithms, where success rate or normalized/averaged scores obtained using episodic training are used. For instance, for Cartpole, our PI-generated policy can maintain the rod in the upright position after a few seconds, while policies generated by most RL algorithms oscillate around the upright position. As long as it does not fall out of a given interval, it is regarded as a success with a reward, which means the trajectories do not asymptotically converge to the equilibrium points. In contrast, our algorithms achieve *asymptotic* stability, meaning the controller can be deployed for any duration of time with convergence guarantees.

F.2. Comparison results

Figure 5 shows the simulated trajectories from different initial conditions under the learned optimal controllers for the four systems in Section 5.3.

F.3. Additional case studies

F.3.1. 1D BILINEAR SYSTEM

Recall the bilinear scalar problem $\dot{x} = xu$, with $Q(x) = x^2$ and $R = 1$. The optimal value function is $V(x) = 2|x|$, which fails to be differentiable at $x = 0$. We show that ELM-PI can converge to the optimal value function. We choose the activation function to be ReLU and set the bias term to zero. ELM-PI achieves 1E-16 accuracy with $m \geq 3$. While this is a simple example, it demonstrates that ELM-PI can potentially achieve arbitrary accuracy, provided that the neural network is

capable of approximating the value function. A plot of the obtained value function after 10 iterations is included in Figure 6.

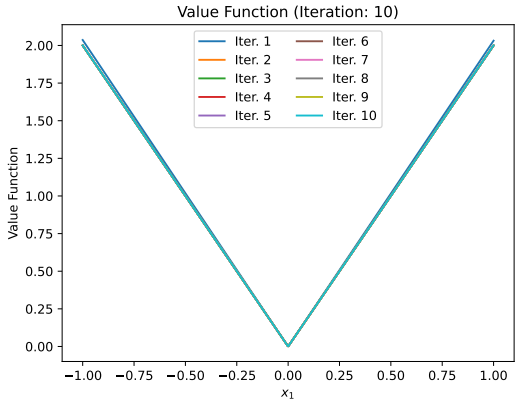


Figure 6: ELM-PI on the bilinear example with $m = 10$. The error between the optimal value and computed value function is within 1E-15 after two iterations.

F.3.2. LORENZ SYSTEM

We consider the stabilization of a chaotic system

$$\begin{aligned} \dot{x}_1 &= -10x_1 + 10x_2 + u, \\ \dot{x}_2 &= 28x_1 - x_2 - x_1x_2, \\ \dot{x}_3 &= -\frac{8}{3}x_2 + x_1x_2. \end{aligned} \tag{48}$$

Without control, the origin is a saddle equilibrium point. We would like to stabilize the system to the origin via policy iteration. We first run ELM-PI with $m = 100$, $m = 200$, $m = 400$, and $m = 800$. The computational times are 0.68, 1.55, 7.03, and 46.84 seconds, respectively. In comparison, SGA with polynomial bases of order 2, 4, and 6 takes 6.63, 69.15, and 893.28 seconds. It can be seen from numerical simulations that ELM-PI with $m = 400$ and $m = 800$ leads to stabilizing controllers, whereas $m = 100$ and $m = 200$ give unstable controllers. Figure 7 depicts 10 simulated closed-loop trajectories under the controllers returned by ELM-PI with $m = 400$ and $m = 800$. The performance of the controllers returned by ELM-PI and the initial controller obtained from eigenvalue assignment for the linearized system are shown in Figure 8 through simulated trajectories.

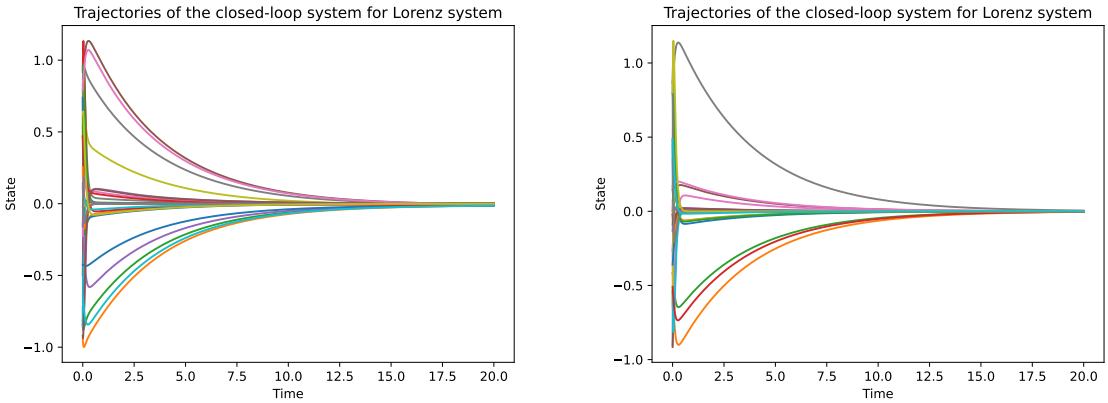


Figure 7: ELM-PI with $m = 400$ and $m = 800$ for the Lorenz system: the left panel shows the closed-loop trajectories under the controller returned by ELM-PI with $m = 400$, and the right panel for $m = 800$.

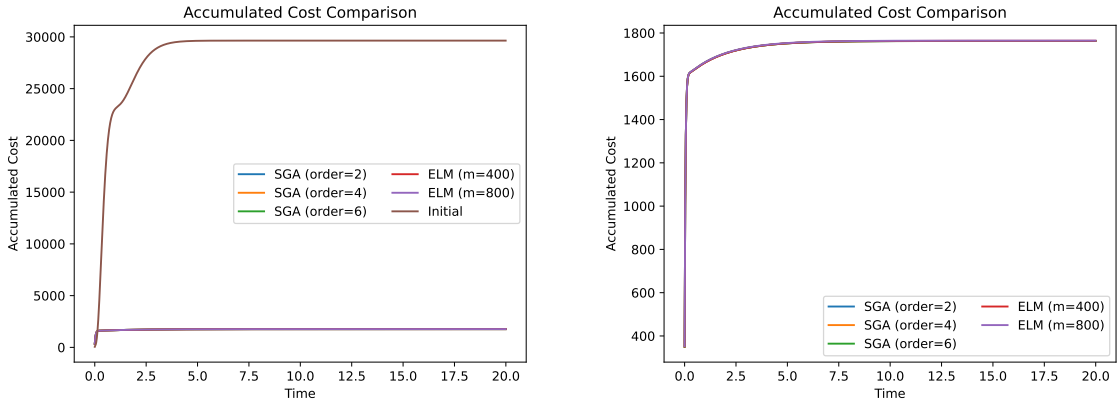


Figure 8: Simulated costs for ELM-PI with $m = 400$ and $m = 800$ for the Lorenz system, compared with the performance of the initial controller obtained from eigenvalue assignment for the linearized system and SGA with polynomial bases of orders 2, 4, and 6. It can be seen that both ELM and SGA achieve almost identical cost, while the computational time required by ELM-PI is considerably less.

G. Limitations and future work

We discuss a few limitations of the proposed work and potential future work in this section.

- **Convergence analysis:** In the convergence analysis, we established that both the exact PI and approximate PI can converge to the true optimal value and controller, provided that the training error can be made arbitrarily small and the training set forms a dense subset of the domain. While this is theoretically interesting, the results do not offer convergence rates or finite sample approximation guarantees. This could be an interesting topic for future research.
- **Initial controller and training over larger domains:** One of the main drawbacks of PI is that it requires a stabilizing controller to begin with. On the other hand, we noticed that both PINN-PI and current RL algorithms also struggle to learn a stabilizing controller over larger domains. In a small region around the equilibrium point, it is always possible to use a linear controller. Since PI requires a controlled invariant set to train the subsequent value and control functions, an interesting topic for future investigation is how to combine controllers that can guarantee to reach a small region of attraction, patched together with a local stabilizing controller, to offer opportunities for training PINN-PI over a larger domain.
- **Unknown systems:** The proposed algorithm requires the exact information of the system and a stabilizing initial controller for performing policy iteration. However, the systems typically cannot be modelled exactly in practice, and sometimes a stabilizing initial policy is not easy to obtain. In our future research, we will combine up-to-date system identification techniques with the proposed PI method to tackle this issue for unknown systems. Meanwhile, learning a stabilizing initial policy purely from data using machine learning-based methods would be an interesting direction as well.
- **Verification:** Formal verification remains challenging for high-dimensional value functions. This difficulty seems unavoidable when computing general optimal value functions. However, there might be ways to circumvent this issue by designing cost metrics that encourage compositional controllers and value functions, or one can deliberately seek compositionally verifiable controllers that are suboptimal, yet still provide satisfactory performance with stability guarantees. We also remarked that the value functions computed with ELM-PI seem harder to verify than those computed with PINN-PI. This may be due to gradient descent implicitly regularizing the functions. Future work can investigate this discrepancy in more detail. One can also incorporate probabilistic guarantees with randomized algorithms for testing.