

DEEP LEARNING FOR HIGH-DIMENSIONAL CONTINUOUS-TIME STOCHASTIC OPTIMAL CONTROL WITHOUT EXPLICIT SOLUTION

Jean-Loup Dupret, Donatien Hainaut

REPRINT | 2026 / 35

ISBA

Voie du Roman Pays 20 - L1.04.01

B-1348 Louvain-la-Neuve

Email : lidam-library@uclouvain.be

<https://uclouvain.be/en/research-institutes/lidam/isba/publication>

Deep learning for high-dimensional continuous-time stochastic optimal control without explicit solution

Jean-Loup Dupret^{a,*}, Donatien Hainaut^{b,**}

^a*Department of Mathematics and RiskLab, ETH Zurich, Rämistrasse 101, Zurich, 8092, Switzerland*

^b*LIDAM-ISBA, Université Catholique de Louvain, Voie du Roman Pays 20, Louvain-La-Neuve, 1348, Belgium*

Abstract

This paper introduces the GPI-PINN algorithm, a novel numerical scheme for solving continuous-time stochastic optimal control problems in high dimensions when the optimal control does not admit an explicit solution. The proposed iterative deep-learning algorithm leverages Physics-Informed Neural Networks on the Hamilton–Jacobi–Bellman residuals in an Actor-Critic fashion, which is built from the Generalized Policy Iteration technique. It employs two separate neural networks to approximate both the value function and the multidimensional optimal control, achieving a global approximation of the optimal solution across all time and space that can be evaluated online rapidly. Theoretical guarantees on convergence and optimality are provided, while its accuracy and efficacy are empirically validated through two important numerical examples from operations research. In particular, we generalize the standard Almgren-Chriss model arising from optimal liquidation in finance by allowing for a price impact model with fully nonlinear temporary and permanent impact functions and by considering a multidimensional setting with numerous co-integrated assets.

Keywords: Machine learning, Stochastic optimal control, Deep learning, Physics-Informed Neural Networks, Optimal liquidation

1. Introduction

A lot of literature has emerged recently for solving high-dimensional partial differential equations based on deep learning, see *e.g.* Han et al. (2018), Sirignano and Spiliopoulos (2018), Han et al. (2017), Beck et al. (2021), Blechschmidt and Ernst (2021). These algorithms can therefore be used to solve continuous-time (CT) stochastic (Gaussian) optimal control problems whenever the optimal control admits an explicit solution in terms of the value function. Indeed, in such cases, the well-known Hamilton–Jacobi–Bellman (HJB) equation reduces to a parabolic second-order PDE. One of the most promising numerical schemes is based on Physics-Informed Neural Networks (PINNs), as introduced in Raissi et al. (2017, 2019), and is quite different from standard supervised machine learning techniques since instead of relying purely on data, it uses the physical properties of the PDE itself to guide the training process. PINNs indeed generate approximate solutions to PDEs by training a neural network to minimize a loss function consisting of terms representing the misfit of the initial and boundary conditions along points sampled from the boundary of

*Corresponding author, Email address: jeanloup.dupret@math.ethz.ch

**Email address: donatien.hainaut@uclouvain.be

the space-time domain, as well as the PDE residual at selected points in the interior domain. We refer in particular to Sirignano and Spiliopoulos (2018) and Blechschmidt and Ernst (2021), where PINNs are applied for solving CT stochastic optimal control problems in high dimensions with explicit solution.

On the opposite, when the optimal control does not admit an explicit representation, we cannot plug it back anymore inside the HJB equation, and we then have to simultaneously solve an optimization problem with respect to the control as well as a (second-order) PDE with respect to the value function, see Section 2.2. These implicit optimal control problems cannot be solved anymore using directly the deep learning techniques mentioned above and therefore require a specifically-designed iterative approximation algorithm. For low-dimensional implicit optimal controls, a classical solution from the reinforcement learning (RL) literature is to use the (Generalized) Policy Iteration (GPI) methods, which are a class of iterative approximation algorithms for both the value function and the optimal control (Jacka and Mijatović, 2017, Sutton and Barto, 2018), see Algorithm 1. In particular, we highlight the Actor-Critic methods from Werbos (1992), which have a separate memory structure to explicitly represent the optimal control independently of the value function and which rely on this (G)PI approach. However, these numerical schemes based on (G)PI usually scale poorly with the problem’s dimensionality as the optimization and the PDE both have to be discretized and solved for every point in a finite grid. This raises the need of meshfree alternative numerical techniques based on deep learning for solving implicit CT stochastic control in high dimensions and continuous action space.

Several (deep) local approaches based on a time discretization of the associated state dynamics have thus been explored in Han and E (2016), Zhou et al. (2021), Nüsken and Richter (2021), Huré et al. (2021), Bachouch et al. (2022), Ji et al. (2022), Li et al. (2024), Domingo-Enrich et al. (2024a,b). These methods typically lead to *local solutions* that are only valid at a fixed, pre-specified set of time-space points from the domain. Among such time-discretized algorithms, we also mention the global-in-time approaches (Hu and Lauriere, 2023, Li and Forsyth, 2019, Fecamp et al., 2020, Warin, 2023, Han and Hu, 2021, van Staden et al., 2023) based on a single (global) network for the control, which takes both time and state variables as input and which is trained by direct maximization of the empirical objective function. As discussed in Section 2.2 and confirmed numerically in Section 5, a key limitation of the global-in-time approach is that the control network tends to learn the optimal policy only along the optimal states trajectories. This phenomenon is sometimes referred to in the literature as semi-local approaches, see Nakamura-Zimmerer et al. (2021). Indeed, as training progresses and the global control network converges to the optimal policy, fewer sample paths deviate from the optimal ones, leading to poor approximations for time-space points far from the optimal trajectory.

Alternatively, meshfree algorithms for implicit control based on deep learning have also gained popularity through the use of deep RL techniques. These include policy gradient methods such as A2C/A3C (Mnih

et al., 2016), PPO (Schulman et al., 2017) and TRPO (Duan et al., 2016); Q-learning type algorithms with DQN (Mnih et al., 2013) or C51 (Bellemare et al., 2017); and hybrid approaches with DDPG (Lillicrap et al., 2019) or SAC (Haarnoja et al., 2018). However, these discrete-time model-free RL algorithms cannot leverage explicitly the known CT internal dynamics of the optimal stochastic control problem. We hence propose instead a deep model-based deterministic on-policy approach that includes explicitly the given internal state dynamics of our control problem thanks to the CT version of the HJB equation (2.6). Moreover, leveraging this CT HJB equation through PINNs eliminates the need to simulate the associated state dynamics, thereby avoiding discretization errors. It also allows us to solve for a global solution in very high dimensions rather than on a constrained time-grid.

Therefore, the numerical scheme proposed in this paper aims at solving CT stochastic optimal control in high dimensions, even when the control cannot be expressed explicitly, and is inspired by both the deep learning and the reinforcement learning theory. More precisely, the so-called GPI-PINN algorithm is obtained by combining PINNs on the HJB residuals together with an Actor-Critic structure built upon the GPI algorithm. This iterative GPI-PINN scheme approximates both the value function and the high-dimensional optimal control using two separate NNs, which are trained on sampled data from the space-time domain. This provides a global approximation of the control and value function available across all time and space, which can then be evaluated online very fast. By imposing an early termination condition in the training process of the GPI-PINN algorithm, we establish both optimality and convergence guarantees of our numerical scheme to the true value function and optimal control. Finally, based on two detailed numerical examples, we show empirically the generality, accuracy and efficacy of the proposed GPI-PINN algorithm for solving implicit stochastic optimal control problems in high dimensions. In the first example, we compare and confirm the performance and accuracy of our GPI-PINN algorithm on a classical linear-quadratic Gaussian control example for which the solution is known up to Monte-Carlo simulations. The second example arises from optimal liquidation in finance where an agent has to sell a large amount of shares and yet minimize adverse price movements generated by her own trades. In a first step, we generalize the standard Almgren and Chriss (2001) and Almgren (2003) models by assuming nonlinear (power-law) temporary and permanent impact functions as it prevents from having an explicit solution for the optimal control. We then extend this fully nonlinear optimal liquidation setting in higher dimensions by considering a large number of co-integrated assets, which interact and influence each other. This indeed leads to a high-dimensional CT implicit stochastic optimal control problem that can be solved efficiently and accurately using our GPI-PINN Algorithm.

The contributions of this article are therefore threefold. We first introduce a new iterative deep-learning approximation algorithm able to solve high-dimensional CT stochastic optimal control problems across all time and space when the control does not admit an explicit solution. We then prove theoretically the uniform

convergence of this so-called GPI-PINN algorithm toward the true optimal solutions. Finally, we show empirically the efficacy and accuracy of the proposed numerical scheme on several examples. In particular, we generalize the standard Almgren-Chriss model arising from optimal liquidation problems by allowing for fully nonlinear temporary and permanent impact functions and by considering a multidimensional setting with numerous co-integrated assets. The organization of the paper is as follows. In Section 2, we review the literature and the theoretical foundations of explicit (Section 2.1) and implicit (Section 2.2) stochastic optimal control problems. In Section 3, we introduce the GPI-PINN algorithm with early termination and establish theoretical convergence guarantees to the optimal solution. Section 4 lays out the neural network implementation used in the rest of the paper. Section 5 performs a comparative example for which a solution is known up to Monte-Carlo simulations and Section 6 shows how the GPI-PINN algorithm can be used to solve complex optimal liquidation strategies in finance. Section 7 concludes.

2. Stochastic optimal control

Let $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t \geq 0}, \mathbb{P})$ be a complete filtered probability space on which is defined a d -dimensional Brownian motion $W = (W_t)_{t \geq 0}$. We fix $T < \infty$ and define $Q = [0, T) \times \mathbb{X}$ with state space $\mathbb{X} \subseteq \mathbb{R}^n$ and $\bar{Q} = [0, T] \times \mathbb{X}$. We consider a fixed final-time CT stochastic (Gaussian) optimal control problem of the form

$$\begin{cases} \text{minimize}_{u \in \mathcal{A}} & V^u(t, x) = \mathbb{E} \left[\phi(X_T^u) + \int_t^T \ell(\tau, X_\tau^u, u_\tau) d\tau \mid X_t^u = x \right], \\ \text{subject to} & dX_\tau^u = f(\tau, X_\tau^u, u_\tau) d\tau + \sigma(\tau, X_\tau^u, u_\tau) dW_\tau, \end{cases} \quad (2.1)$$

for all $(t, x) \in Q$. Here $X^u : [0, T) \times \Omega \rightarrow \mathbb{X}$ is the state process, $u : [0, T) \times \Omega \rightarrow \mathbb{U} \subseteq \mathbb{R}^m$ is the control process and $\mathbb{U}$ is a closed subset of $\mathbb{R}^m$. $\mathcal{A}$ denotes the set of admissible controls, which consists of all progressively measurable m -dimensional processes $u = (u_t)_{t \in [0, T)}$ with finite $\mathbb{L}^2(\Omega \times [0, T))$ -norm. The drift and diffusion coefficients $f : Q \times \mathbb{U} \rightarrow \mathbb{R}^n$ and $\sigma : Q \times \mathbb{U} \rightarrow \mathbb{R}^{n \times d}$ are assumed to be continuous and satisfy the Lipschitz conditions

$$\begin{aligned} |f(t, x, u) - f(t, y, u)| + |\sigma(t, x, u) - \sigma(t, y, u)| &\leq K|x - y|, \\ |f(t, x, u)| + |\sigma(t, x, u)| &\leq K(1 + |x| + |u|), \end{aligned} \quad (2.2)$$

for K independent of (t, x, y, u) . From Touzi (2012), for each admissible control $u \in \mathcal{A}$, the controlled SDE (2.1) has then a unique $\mathbb{F}$ -adapted solution X^u , for all $X_0 \in \mathbb{L}^2(\mathcal{F}_0, \mathbb{P})$ such that for some constant C ,

$$\mathbb{E} \left[\sup_{0 \leq s \leq t} |X_s^u|^2 \right] < C (1 + \mathbb{E} [|X_0|^2]) e^{Ct} \quad \text{for all } t \in [0, T).$$

Finally, $\phi : \mathbb{X} \rightarrow \mathbb{R}$ is the terminal cost at maturity and $\ell : Q \times \mathbb{U} \rightarrow \mathbb{R}$ is a continuous running cost. They are assumed to be both measurable and satisfy the following quadratic growth condition for some constant K independent of (t, x, u) , $|\ell(t, x, u)| + |\phi(x)| \leq K(1 + |u|^2 + |x|^2)$. This quadratic growth on ℓ and ϕ ensures that $V^u(t, x)$ is finite and well-defined for all $u \in \mathcal{A}$. We consider in this work Markovian

controls, *i.e.* processes $u \in \mathcal{A}$ that can be written in the form $u_t = u(t, X_t^u)$ for each $t \in [0, T)$ and for some measurable map u from Q into $\mathbb{U}$. We also denote $u_t^* = u^*(t, X_t^{u^*})$ the optimal closed-loop feedback control minimizing the cost V^u (2.1), with $u^* : Q \rightarrow \mathbb{U}$ measurable. Standard notation in dynamic programming defines the value function $V : \bar{Q} \rightarrow \mathbb{R}$ as the optimal cost-to-go (2.1) starting at $(t, x) \in Q$,

$$V(t, x) := \inf_{u \in \mathcal{A}} V^u(t, x) = \mathbb{E} \left[\phi(X_T^{u^*}) + \int_t^T \ell(\tau, X_\tau^{u^*}, u^*(\tau, X_\tau^{u^*})) d\tau \mid X_t^{u^*} = x \right], \quad (2.3)$$

with terminal condition $V(T, x) = \phi(x)$. Let $\mathcal{S}_n$ the set of all $n \times n$ symmetric matrices with real coefficients. We next define the Hamiltonian by the map $\mathcal{H} : Q \times \mathbb{R}^n \times \mathcal{S}_n \times \mathbb{U} \rightarrow \mathbb{R}$,

$$\mathcal{H}(t, x, p, \gamma, u) := \ell(t, x, u) + p^\top f(t, x, u) + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x, u) \gamma \right]. \quad (2.4)$$

The minimized Hamiltonian H with respect to the control is then defined as

$$H(t, x, p, \gamma) := \inf_{u \in \mathbb{U}} \mathcal{H}(t, x, p, \gamma, u), \quad (2.5)$$

and we have the following crucial theorem related to the value function V , see the review of Touzi (2012).

Theorem 2.1. *Assume the value function $V \in C^{1,2}(Q)$, let the minimized Hamiltonian H in (2.5) be continuous in (t, x) with $H(\cdot, \cdot, V_x, V_{xx}) < \infty$ and let $\ell(\cdot, \cdot, u)$ be continuous in (t, x) for all fixed $u \in \mathbb{U}$, then the value function V solves the following HJB PDE for every $(t, x) \in Q$,*

$$\begin{cases} V_t(t, x) + \inf_{u \in \mathbb{U}} \left\{ \ell(t, x, u) + [V_x(t, x)]^\top f(t, x, u) + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x, u) V_{xx}(t, x) \right] \right\} = 0 \\ V(T, x) = \phi(x), \end{cases} \quad (2.6)$$

where $V_t := \partial V / \partial t$, $V_x := (\partial V / \partial x_1, \dots, \partial V / \partial x_n)^\top$ the gradient of V and V_{xx} the Hessian of V .

If V satisfies Theorem 2.1, then it is said to be a classical solution of the HJB equation (2.6). We directly see that we can rewrite this HJB equation (2.6) as

$$\begin{cases} V_t(t, x) + H(t, x, V_x(t, x), V_{xx}(t, x)) = 0, \\ V(T, x) = \phi(x), \end{cases} \quad (2.7)$$

We now give a crucial verification theorem, the proof of which is outlined in Touzi (2012).

Theorem 2.2. *Let $\widehat{V} \in C^{1,2}(Q) \cap C(\bar{Q})$ be a solution of (2.7). If there exists $\hat{u}(t, x)$ for each $(t, x) \in Q$ such that¹*

$$\hat{u}(t, x) \in \arg \min_{u \in \mathbb{U}} \mathcal{H}(t, x, \widehat{V}_x(t, x), \widehat{V}_{xx}(t, x), u), \quad (2.8)$$

for which the process $\hat{u}_s := \hat{u}(s, X_s^{\hat{u}})$ is a well-defined control in $\mathcal{A}$ and such that the SDE

$$dX_s^{\hat{u}} = f(s, X_s^{\hat{u}}, \hat{u}(s, X_s^{\hat{u}}))ds + \sigma(s, X_s^{\hat{u}}, \hat{u}(s, X_s^{\hat{u}}))dW_s, \quad t \leq s < T, \quad X_t^{\hat{u}} = x,$$

¹or similarly such that $\widehat{V}_t(t, x) + \mathcal{H}(t, x, \widehat{V}_x(t, x), \widehat{V}_{xx}(t, x), \hat{u}(t, x)) = 0$, see (2.7).

admits a unique solution for each initial data $x \in \mathbb{X}$, then $\widehat{V}(t, x) = V(t, x)$ for all $(t, x) \in \bar{Q}$ and $\hat{u}(t, x)$ must be an optimal (Markov) control $u^*(t, x)$ for all $(t, x) \in Q$.

From this verification theorem, we propose in this paper as candidate $\widehat{V}$ a NN-parametrized value function $V(\cdot; \theta)$ on $\bar{Q}$ chosen to be $C^{1,2}(Q) \cap C(\bar{Q})$ and whose neural weights θ are estimated in such a way that the PDE (2.7) is satisfied. We then seek a control candidate $\hat{u}$ minimizing (2.8). This control $\hat{u}$ will also be parameterized by a (second) NN $u(\cdot; \omega)$ on Q , with weights ω trained to minimize (2.8) on average over Q . We first note that selecting two-times differentiable activation functions in our NN ensures that $V(\cdot; \theta)$ is $C^2(\bar{Q})$, as it is the composition of two-times differentiable activation functions. Activation functions like the tanh, sigmoid, softplus or linear thus imply that the value NN $V(\cdot; \theta)$ is smooth and can therefore solve for a classical solution of the PDE (2.6). In the case $\bar{Q}$ compact, the choice of a NN approximation for the value function can also be justified by the universal approximation theorem (see Lemma 3.2 below). Note that under the assumption that the coefficients $f(\cdot)$, $\sigma(\cdot)$ and $\ell(\cdot)$ are C^1 in u and if $\hat{u}(\cdot)$ belong to the interior of $\mathbb{U}$, equation (2.8) implies

$$\mathcal{H}_u(t, x, V_x(t, x), V_{xx}(t, x), \hat{u}(t, x)) := \left. \frac{\partial}{\partial u} \mathcal{H}(t, x, V_x(t, x), V_{xx}(t, x), u) \right|_{u=\hat{u}(t, x)} = \mathbf{0}_m. \quad (2.9)$$

Therefore, finding a $\hat{u}(\cdot)$ satisfying (2.9) provides a good candidate for the optimal control, even if this condition is not sufficient for ensuring the optimality of our candidate. The following notation $\tilde{\mathcal{H}}$ for the so-called *modified Hamiltonian* will also prove to be particularly useful throughout the manuscript

$$\tilde{\mathcal{H}}(t, x, V_x(t, x), V_{xx}(t, x), u(t, x)) := \mathcal{H}(t, x, V_x(t, x), V_{xx}(t, x), u(t, x)) + V_t(t, x), \quad (t, x) \in Q. \quad (2.10)$$

2.1. Explicit optimal control

For certain CT stochastic optimal control problems, the minimization problem (2.8) in u can be simplified so as to express the optimal control u^* explicitly as a function of t, x, V_x and V_{xx} . Therefore, in this case, the minimized Hamiltonian $H(t, x, V_x, V_{xx})$ can also be obtained explicitly as a function of (t, x, V_x, V_{xx}) , as shown *e.g.* in Example 1 below with equation (2.13). The PDE (2.7) is then directly solved for V since it does not depend anymore on the optimal control. This value function $V(t, x)$ obtained from the HJB equation is next plugged back into the formula for $u^*(t, x)$ so as to also derive the optimal control explicitly. This two-step method (with first the derivation of V and then of u^*) is illustrated in the following example.

Example 1 : We consider a simple case with an affine stochastic control and convex running cost in $u \in \mathbb{U} = \mathbb{R}^m$. The system dynamic is given by

$$dX_t^u = f(t, X_t^u)u_t dt + \sigma(t, X_t^u)dW_t, \quad X_0 = x_0,$$

where $f : Q \rightarrow \mathbb{R}^{n \times m}$, $\sigma : Q \rightarrow \mathbb{R}^{n \times d}$. The running cost is assumed to be of the form $\ell(t, x, u) = h(t, x) + u^\top \mathbf{R}u$, for some function $h : Q \rightarrow \mathbb{R}$ and some positive definite weight matrix $\mathbf{R} \in \mathbb{R}^{m \times m}$.

Then, the Hamiltonian (2.4) becomes

$$\mathcal{H}(t, x, p, \gamma, u) = h(t, x) + u^\top \mathbf{R}u + p^\top f(t, x)u + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x) \gamma \right]. \quad (2.11)$$

The minimization problem (2.8) for u^* with $p = V_x$ amounts to compute the gradient of $\mathcal{H}$ with respect to u and set it to zero

$$\mathbf{0}_m = \mathcal{H}_u(t, x, V_x(t, x), V_{xx}(t, x), u^*(t, x)) = 2\mathbf{R}u^*(t, x) + f^\top(t, x)V_x(t, x).$$

Solving for u^* yields the optimal feedback control policy in explicit form

$$u^*(t, x) = -\frac{1}{2}\mathbf{R}^{-1}f^\top(t, x)V_x(t, x), \quad (2.12)$$

and substituting u^* back in (2.11), we also obtain the minimized Hamiltonian explicitly

$$H(t, x, V_x(t, x), V_{xx}(t, x)) = h(t, x) - \frac{1}{4}V_x^\top(t, x)f(t, x)\mathbf{R}^{-1}f^\top(t, x)V_x(t, x) + \frac{1}{2}\text{Tr} \left[\sigma \sigma^\top(t, x)V_{xx}(t, x) \right]. \quad (2.13)$$

We then obtain for all $(t, x) \in Q$ the following HJB PDE satisfied by the value function V :

$$V_t(t, x) + h(t, x) - \frac{1}{4}V_x^\top(t, x)f(t, x)\mathbf{R}^{-1}f^\top(t, x)V_x(t, x) + \frac{1}{2}\text{Tr} \left[\sigma \sigma^\top(t, x)V_{xx}(t, x) \right] = 0, \quad (2.14)$$

with terminal condition $V(T, x) = \phi(x)$. We hence see that this HJB PDE (2.14) can be solved independently of the control u . Once the value function V is solved, we can plug its space derivative into (2.12) and obtain the optimal control u^* .

2.1.1. Literature review

Solving the parabolic PDE (2.14) for the value function arising from such explicit stochastic control problem often requires numerical methods, especially in higher dimensions. Recently, many numerical schemes based on deep learning have emerged and have been proposed for solving such kind of PDEs since they limit the increased complexity when considering high-dimensional control problems, see *e.g.* Hutzenthaler et al. (2020). Moreover, they offer the possibility to build complex functions $f(t, x; \theta)$ that can directly approximate the value function and/or the optimal control across all time and space. Once the set of parameters θ is estimated offline, these functions can then be rapidly evaluated online for every (t, x) of the time-space domain. Such deep learning techniques can be divided into two main approaches, as investigated in the review paper of Blechschmidt and Ernst (2021). First, we have the Physics-Informed Neural Networks (PINNs) introduced in Raissi et al. (2019) which use the physical properties of the PDE itself to guide the training process. They indeed generate approximate solutions to PDEs by training a neural network to minimize a loss function consisting of terms representing the misfit of the initial and boundary conditions along points sampled from the boundary of the space-time domain, as well as the PDE residual at selected points in the interior of the domain. More precisely, the PINN loss function is defined

$$\mathcal{L}(\theta) = \left\| \partial_t V(t, x; \theta) + H(t, x, V_x(t, x; \theta), V_{xx}(t, x; \theta)) \right\|_{Q, \mu_1}^2 + \left\| V(T, x; \theta) - \phi(x) \right\|_{\mathbb{X}, \mu_2}^2, \quad (2.15)$$

where $\|g(y)\|_{\mathcal{Y},\mu}^2 = \int_{\mathcal{Y}} |g(y)|^2 \mu(y) dy$ and $\mu(y)$ a positive probability density on $\mathcal{Y}$, and where $V(\cdot; \theta)$ is a NN parameterized by θ which approximates the value function (2.3). As explained in the DGM algorithm of Sirignano and Spiliopoulos (2018), the loss function $\mathcal{L}(\theta)$ measures how well the function $V(\cdot; \theta)$ satisfies the HJB equation (2.7)². If $\mathcal{L}(\theta) = 0$, then $V(\cdot; \theta)$ is solution of the HJB equation. Therefore, the goal is to find a set of parameters θ such that the function $V(\cdot; \theta)$ minimizes the error $\mathcal{L}(\theta)$ and leads to a reduced-form model which approximates $V(t, x)$ for all $(t, x) \in \bar{Q}$. Estimating θ by minimizing directly $\mathcal{L}(\theta)$ is infeasible when the dimension n is large since the integral over Q and $\mathbb{X}$ is intractable computationally. However, using stochastic gradient descent (SGD) with machine learning approach limits the curse of dimensionality by drawing a sequence of time and space points according to some suitably chosen sampling measures μ_1 on Q and μ_2 on $\mathbb{X}$, while satisfying the dynamics (2.1). In this explicit case, once V is approximated by the NN $V(\cdot; \theta)$, the optimal control can then be derived directly from this approximated value function, see *e.g.* equation (2.12). The strength of PINNs lies in their flexibility in that they can be applied to a great variety of challenging PDEs, see *e.g.* Sirignano and Spiliopoulos (2018) with the DGM algorithm, Berg and Nyström (2018) or Mowlavi and Nabi (2023), see also Bai et al. (2022) or Hainaut (2023) in finance. Moreover, PINNs provide a global solution of the PDE across all time and space (t, x) in the domain $\bar{Q}$, which is much more suitable for real-time applications.

The second set of methods consists in algorithms based on the Feymann-Kac formula for solving (Kolmogorov) linear parabolic second-order PDEs, as introduced in Beck et al. (2018) and Berner et al. (2020). Such Feymann-Kac approach connects the solution of the Komogorov PDE to the expectation of a stochastic process, which appears to work extremely well in high dimensions. This approach can also be extended to semi-linear (or even fully nonlinear) PDEs in high dimensions through a nonlinear Feymann-Kac representation, which allows to make the correspondence between nonlinear PDEs and backward stochastic differential equations (BSDEs), see the deep BSDE solver of Han et al. (2017) and its use in Han et al. (2018) for solving high-dimensional (explicit) HJB PDE. Note that this second set of methods provides local solutions and hence focus on computing the value of the PDE solution on a time-grid (at pre-specified times), on the contrary to the global solution provided by PINNs (see Section 5).

2.2. Implicit optimal control

Unlike the previous section on explicit controls, we are especially interested in this paper in the case where the optimal control u^* cannot be obtained explicitly as a function of (t, x, V_x, V_{xx}) . Instead, it is only available implicitly as soon as the minimization (2.8) does not admit an analytical solution. In this case, we know from Theorem 2.2 that the optimal control $u^*(t, x)$ should minimize the Hamiltonian

²Note that potential boundary conditions on the HJB equation inferred from the physics of the problem can also be considered easily by adding a third residual term $\left\| V(t, x; \theta) - g(t, x) \right\|_{\partial Q, \mu_3}^2$ in (2.15), defined on the boundary $\partial Q := [0, T) \times \partial \mathbb{X}$, see Section 6 on optimal liquidation problems.

$\mathcal{H}(t, x, V_x(t, x), V_{xx}(t, x), u)$ for $u \in \mathbb{U}$ while the value function should satisfy the HJB equation (2.6). We then have to solve simultaneously this HJB PDE for the value function V and the minimization (2.8) for the optimal control u^* . Numerical methods exist based on recursive schemes for approximating u^* (see e.g. Algorithm 1 below), but these schemes are already computationally intensive even for low-dimensional problems. Therefore, the curse of dimensionality that typically appears for high-dimensional PDEs is exacerbated by the extra simultaneous minimization step required when no explicit expression is available for the optimal control u^* . This raises the need of alternative numerical techniques based on deep learning for tackling implicit CT stochastic optimal control problems, especially in high dimensions, as shown in the following example.

Example 2 : We now consider the following stochastic dynamic for the state

$$dX_t^u = f(t, X_t^u) \sin(u_t) dt + \sigma(t, X_t^u) dW_t, \quad X_0 = x_0,$$

for some function $f : Q \rightarrow \mathbb{R}^{n \times m}$, $\sigma : Q \rightarrow \mathbb{R}^{n \times d}$ and where the running cost $\ell(t, x, u) = h(t, x) + u^\top \mathbf{R}u$, as in Example 1. The Hamiltonian then becomes

$$\mathcal{H}(t, x, p, \gamma, u) = h(t, x) + u^\top \mathbf{R}u + p^\top f(t, x) \sin(u) + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x) \gamma \right]. \quad (2.16)$$

The minimization problem (2.8) for u^* with $p = V_x$ and $\gamma = V_{xx}$ implies

$$\mathbf{0}_m = \mathcal{H}_u(t, x, V_x(t, x), V_{xx}(t, x), u^*(t, x)) = 2\mathbf{R}u^*(t, x) + V_x^\top(t, x) f(t, x) \cos(u^*(t, x)), \quad (2.17)$$

which does not admit an explicit solution for u^* . Therefore, we cannot proceed as in Example 1 and we must solve the optimal control problem numerically since equation (2.17) and the HJB PDE (2.7) should now be solved simultaneously.

2.2.1. Literature review

The stochastic control literature does not provide lots of insights for solving implicit control problems of the form (2.1) when the dimension is high. Indeed, most of the available results based on deep learning are related to explicit control problems, as seen in the previous section. Fortunately, we can borrow from the literature on reinforcement learning (mainly developed for discrete-time problems) some insights on how to solve these implicit CT stochastic optimal control problems. One of the most classical ways is to use the Howard or Policy Iteration (PI) algorithm, originally derived in Howard (1960) for discrete-time reinforcement learning applications. This algorithm defines a successive approximation algorithm that can also be applied for CT stochastic optimal control problems, as highlighted in Algorithm 1 below. As in equation (2.1), the approximated value function $V(t, x)$ for every $(t, x) \in \bar{Q}$ is denoted at step k of the Algorithm 1 by $V^{u^{(k)}}(t, x)$, highlighting its dependence on the control $u^{(k)}(t, x)$ computed at the previous iteration. It can be shown from Peyrl et al. (2005) and Jacka and Mijatović (2017) that $V^{u^{(k+1)}}(t, x) \leq V^{u^{(k)}}(t, x)$ with $\lim_{k \rightarrow \infty} V^{u^{(k)}}(t, x) = V(t, x)$ for all $(t, x) \in \bar{Q}$ and that $\lim_{k \rightarrow \infty} u^{(k)}(t, x) = u^*(t, x)$ for all

Algorithm 1 PI for CT stochastic optimal control

Choose an initial admissible control policy $u^{(0)}(\cdot) \in \mathcal{A}$.

Set the tolerance $\varepsilon > 0$ and let $k = 0$.

repeat

Step 1. Find $V^{u^{(k)}}(t, x)$ for all $(t, x) \in \bar{Q}$ by solving the following PDE for the known control $u^{(k)}$ at step k :

$$\begin{aligned} V_t^{u^{(k)}}(t, x) + \ell(t, x, u^{(k)}(t, x)) + \left[V_x^{u^{(k)}}(t, x) \right]^\top f(t, x, u^{(k)}(t, x)) \\ + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x, u^{(k)}(t, x)) V_{xx}^{u^{(k)}}(t, x) \right] = 0 \end{aligned} \quad (2.18)$$

with terminal condition $V^{u^{(k)}}(T, x) = \phi(x)$.

Step 2. Compute the control $u^{(k+1)}(t, x)$ at step $k + 1$ from the known functional $V^{u^{(k)}}(t, x)$ by solving for every $(t, x) \in Q$:

$$u^{(k+1)}(t, x) = \arg \min_{u \in \mathbb{U}} \left\{ \ell(t, x, u) + \left[V_x^{u^{(k)}}(t, x) \right]^\top f(t, x, u) + \frac{1}{2} \text{Tr} \left[\sigma \sigma^\top(t, x, u) V_{xx}^{u^{(k)}}(t, x) \right] \right\} \quad (2.19)$$

$k \leftarrow k + 1$

until $\max_{(t,x) \in \bar{Q}} \left| V^{u^{(k)}}(t, x) - V^{u^{(k-1)}}(t, x) \right| \leq \varepsilon$

return $V^{u^{(k^*)}}$ and $u^{(k^*)}$ with $k^* \leftarrow k$.

$(t, x) \in Q$. For applying this iterative algorithm, we need to solve the PDE (2.18), using *e.g.* finite-difference schemes. From this finite grid, the optimization problem (2.19) then needs to be solved for every grid point using approximation methods, which thus scale poorly with the problem's dimension: BFGS and its extended techniques, gradient descent methods, Sequential least squares programming, and so on (see the review of Greif (2017)). In the original PI Algorithm 1, we have to completely solve (2.18) before fully evaluating (2.19). This is however not necessary. Indeed, PI belongs to the Generalized Policy Iteration (GPI) techniques, which is a family of RL techniques introduced in Sutton and Barto (2018) with PI at one extreme and value iteration at the other extreme. In GPI, at each step, one does not completely evaluate the value $V^{u^{(k)}}$ for a given control $u^{(k)}$, but only updates the current cost estimate towards that value. Likewise, one does not fully update the control policy to the greedy control defined as $u^{(k+1)}(t, x) := \arg \min_{u \in \mathbb{U}} \mathcal{H}(t, x, V_x^{u^{(k)}}(t, x), V_{xx}^{u^{(k)}}(t, x), u)$ for the new value estimate at step k , but only updates the control towards it. Numerous other RL algorithms (SARSA, Q-learning, TD(λ), etc.) are presented in Sutton and Barto (2018) or in Bertsekas (2023) and can be used for solving discrete-time (deterministic and stochastic) optimal control problems. Among these algorithms, we highlight the Actor-Critic methods from Werbos (1992), which have a separate memory structure to explicitly represent the optimal control u^* independently of the value function V . Most of Actor-Critic methods then rely on a (G)PI scheme as in Algorithm 1 and were initially developed for low-dimensional discrete-time stochastic

control problems.

Therefore, these classical RL algorithms do not solve the fact mentioned above that implicit control problems scale poorly when increasing the number of dimensions n (or m) and are then especially prone to suffer from the curse of dimensionality. This can be overcome in discrete-time by combining these RL algorithms with function approximations based on deep learning, *i.e.* *deep RL*, see the Introduction 1 and the reviews of Li (2017), Fujimoto et al. (2018). However, these discrete-time model-free deep RL algorithms cannot leverage explicitly the known CT internal dynamics of the optimal stochastic control problem (2.1), as confirmed numerically in Section 5, Figure 5.3. In this paper, we then instead propose a deep model-based deterministic on-policy approach that includes explicitly the given internal state dynamics of our control problem (2.1) thanks to the CT version (2.7) of the HJB equation.

Finally, we also mention that several local approaches based on a time-discretization of the state dynamics (2.1) have been explored in Han and E (2016), Nüsken and Richter (2021), Huré et al. (2021), Bachouch et al. (2022), Ji et al. (2022), Li et al. (2024), Domingo-Enrich et al. (2024a,b). These methods then lead to local solutions at a restricted set of pre-specified points but fail to provide a global solution across the entire time-space domain, *i.e.* for any $(t, x) \in Q$. Similarly, semi-local (or global-in-time) approaches (see *e.g.* Remark 2.2 of Hu and Lauriere (2023)) tend to only learn the optimal control along the optimal state trajectories $(X_t^{u^*})_{t \in [0, T]}$, ultimately learning solely the policies $u^*(t, X_t^{u^*})$, $t \in [0, T]$. Indeed, as the training progresses and the NN $u(\cdot; \omega)$ converges to the optimal u^* , fewer sample paths tend to deviate from the optimal ones. In contrast, leveraging the CT HJB equation (2.7) eliminates the need to simulate the associated state dynamics (2.1), thereby avoiding discretization errors and leading to a global solution in very high dimensions, rather than on a constrained time-grid or along the optimal state trajectories. This will also be confirmed numerically in Section 5, Figure 5.3.

To summarize, we now aim at developing a deep learning algorithm able to approximate both the optimal control and the value function in our CT stochastic optimal control problem (2.1) with finite horizon, which will be specifically designed to perform well in a high-dimensional setting with implicit control. Our method, inspired by both the reinforcement learning theory and the PINN literature, will provide a global approximation of the (m -dimensional) optimal control and of the value function available across all time and space, which can then be evaluated online very fast. This will be achieved by combining PINNs on the HJB residuals in an Actor-Critic fashion, based on the (G)PI scheme of Algorithm 1. We therefore first propose in Section 3.1 the PI-PINN Algorithm 2, which relies on the PI recursive scheme. The PI-PINN Algorithm 2 is then enhanced in Section 3.2 with the GPI-PINN Algorithm 3, which instead implements a GPI scheme with early termination, enabling faster numerical convergence. We also derive theoretical convergence guarantees for this improved algorithm with Theorem 3.4.

3. Deep learning for implicit CT stochastic optimal control

In addition to the NN $V(\cdot; \theta)$ used to approximate the value function, we also introduce a second network $u(\cdot; \omega)$, parameterized by ω , which approximates the m -dimensional optimal control in the implicit setting. We then need to sequentially solve the HJB equation with respect to the weights θ of the first NN $V(\cdot; \theta)$ while keeping the weights ω of the optimal control NN constant, and then minimize the Hamiltonian with respect to the weights ω of the second NN $u(\cdot; \omega)$ while keeping the weights θ of the value NN constant. The (G)PI Algorithm 1 with an Actor-Critic structure built upon the two NNs $V(\cdot; \theta)$ and $u(\cdot; \omega)$ will therefore provide the perfect framework for this iterative optimization scheme. When working with the NNs $V(\cdot; \theta)$ and $u(\cdot; \omega)$, we simplify from now on the notation of the Hamiltonian (2.4)–and its modified counterpart (2.10)–using

$$\mathcal{H}(t, x, \theta, \omega) := \mathcal{H}(t, x, V_x(t, x; \theta), V_{xx}(t, x; \theta), u(t, x; \omega)), \quad (t, x) \in Q.$$

3.1. PI-PINN Algorithm 2

We can directly see in Algorithm 2 below that we first minimize the loss function (3.2) with respect to the value weights θ so as to enforce the HJB equation (2.7) satisfied by the value function (even if at step k the control is not yet optimal). We then minimize the second loss function (3.4) with respect to the control weights ω in order to minimize the Hamiltonian (2.4) with respect to the control u in an average sense according to μ_1 , as suggested by Theorem 2.2. Note indeed from the notation (2.10) that finding the weights ω of the optimal control NN minimizing $\mathcal{H}(\cdot)$ or $\tilde{\mathcal{H}}(\cdot)$ is equivalent, but working with $\tilde{\mathcal{H}}(\cdot)$ will prove more convenient in the proof of Theorem 3.4. Alternatively, from condition (2.9), we can impose instead of the loss (3.4) that the gradient of $\tilde{\mathcal{H}}$ with respect to u is equal to zero using the loss

$$\mathcal{L}_{2'}(\theta^{(k+1)}, \omega) = \int_Q \left\| \tilde{\mathcal{H}}_u(t, x, \theta^{(k+1)}, \omega) \right\|_2^2 d\mu_1(t, x), \quad (3.5)$$

where $\tilde{\mathcal{H}}_u(t, x, \theta, \omega) := \nabla_u \tilde{\mathcal{H}}(t, x, V_x(t, x; \theta), V_{xx}(t, x; \theta), u) \Big|_{u=u(t, x; \omega)}$. This does not ensure to find a global minimum but this may speed up convergence in some cases. We also see that we only need to train two NNs $V(\cdot; \theta)$ and $u(\cdot; \omega)$ with only two sets of parameters θ and ω so as to approximate the value function $V(t, x)$ for all $(t, x) \in \bar{Q}$ and the optimal control $u^*(t, x)$ for all $(t, x) \in Q$. This differs from Huré et al. (2021) and Bachouch et al. (2022) where two sets of parameters $(\theta^{(n)}, \omega^{(n)})$ need to be estimated at each discretized time t_n . Therefore, Algorithm 2 leads to a global approximation on the space-time domain of the value function using $V(\cdot; \theta^{(k^*)})$ and of the optimal control using $u(\cdot; \omega^{(k^*)})$, where k^* is the last epoch in Algorithm 2. Once the parameters θ and ω are estimated based on some generated dataset, we can then rapidly evaluate the optimal control and the value function online, making it suitable for real-time applications. Note that the loss functions (3.2) and (3.4) are estimated in practice by the following sample average procedure. Generate M_1 random points $\{(t_m^{(1)}, x_m^{(1)})\}_{m=1}^{M_1}$ from Q according to μ_1 and M_2 random

Algorithm 2 PI-PINN for implicit CT stochastic optimal control

Initialize admissible neural weights $\omega^{(0)}$ for the NN $u(\cdot; \omega^{(0)})$ and neural weights $\theta^{(0)}$ for the NN $V(\cdot; \theta^{(0)})$. Set the tolerance $\varepsilon > 0$ and let $k = 0$.

repeat

Step 1. Update the value NN $V(\cdot; \theta^{(k+1)})$ for the known control $u(\cdot; \omega^{(k)})$ at step k , by minimizing the following loss function $\mathcal{L}_1$ with respect to the weights θ :

$$\theta^{(k+1)} = \arg \min_{\theta} \mathcal{L}_1(\theta, \omega^{(k)}), \quad (3.1)$$

where

$$\begin{aligned} \mathcal{L}_1(\theta, \omega^{(k)}) &= \left\| \partial_t V(t, x; \theta) + \mathcal{H}(t, x, \theta, \omega^{(k)}) \right\|_{Q, \mu_1}^2 + \left\| V(T, x; \theta) - \phi(x) \right\|_{\mathbb{X}, \mu_2}^2 \\ &= \mathbb{E}_{(t,x) \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(t, x, \theta, \omega^{(k)})^2 \right] + \mathbb{E}_{x \sim \mu_2(\mathbb{X})} \left[(V(T, x; \theta) - \phi(x))^2 \right]. \end{aligned} \quad (3.2)$$

Step 2. Update the control NN $u(\cdot; \omega^{(k+1)})$ for the known value NN $V(\cdot; \theta^{(k+1)})$ by minimizing the loss $\mathcal{L}_2$ with respect to the weights ω :

$$\omega^{(k+1)} = \arg \min_{\omega} \mathcal{L}_2(\theta^{(k+1)}, \omega), \quad (3.3)$$

where

$$\mathcal{L}_2(\theta^{(k+1)}, \omega) = \int_Q \tilde{\mathcal{H}}(t, x, \theta^{(k+1)}, \omega) d\mu_1(t, x) = \mathbb{E}_{(t,x) \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(t, x, \theta^{(k+1)}, \omega) \right]. \quad (3.4)$$

$k \leftarrow k + 1$

until $\max_{(t,x) \in \bar{Q}} |V(t, x; \theta^{(k)}) - V(t, x; \theta^{(k-1)})| \leq \varepsilon$

return $V(\cdot; \theta^{(k^*)})$ and $u(\cdot; \omega^{(k^*)})$ with $k^* \leftarrow k$.

points $\{(x_m^{(2)})\}_{m=1}^{M_2}$ from $\mathbb{X}$ according to μ_2 . Then, the loss functions (3.2) and (3.4) are estimated using

$$\widehat{\mathcal{L}}_1(\theta, \omega^{(k)}) = \frac{1}{M_1} \sum_{m=1}^{M_1} \left(\tilde{\mathcal{H}}(t_m^{(1)}, x_m^{(1)}, \theta, \omega^{(k)}) \right)^2 + \frac{1}{M_2} \sum_{m=1}^{M_2} \left(V(T, x_m^{(2)}; \theta) - \phi(x_m^{(2)}) \right)^2, \quad (3.6)$$

and

$$\widehat{\mathcal{L}}_2(\theta^{(k)}, \omega) = \frac{1}{M_1} \sum_{m=1}^{M_1} \tilde{\mathcal{H}}(t_m^{(1)}, x_m^{(1)}, \theta^{(k)}, \omega). \quad (3.7)$$

These sampling measures μ_1, μ_2 are chosen according to the Residual-based Adaptive Distribution (RAD) method of Wu et al. (2023), built on Sobol sampling, as it is known to significantly improve the accuracy of PINNs. Finally, in the loss function (3.2) (and in its empirical counterpart (3.6)), we can introduce a proportionality factor between the first residual term and the second boundary term. This hyper-parameter can be fine-tuned for improving the convergence of the algorithm, see *e.g.* Wang et al. (2022).

3.2. GPI-PINN Algorithm 3

However, as noted in the PI Algorithm 1, it is not necessary to wait for full convergence of the optimizations (3.1) and (3.3) to their exact solutions. Instead, a GPI approach can be employed, wherein only a limited number of optimization steps are performed in each iteration. This strategy has been shown to accelerate convergence in various contexts, see Sutton and Barto (2018). Building on this idea, we propose the GPI-PINN Algorithm 3, which extends the PI-PINN Algorithm 2 by allowing early termination in the minimization of (3.1) and (3.3). Since parameter optimization is typically achieved in NN via gradient descent, we will use in practice SGD or mini-batch updates on the empirical losses (3.6) and (3.7), for successfully training the value network $V(\cdot; \theta)$ and the optimal control network $u(\cdot; \omega)$.

Therefore, in Algorithm 3 below, the gradient steps (3.9) and (3.10) are based in practice on SGD or mini-batches using the empirical formulae (3.6) and (3.7). Indeed, multiple theoretical analyses and experimental studies like Allen-Zhu et al. (2019), Du et al. (2019) or Jentzen et al. (2021) have shown that optimization algorithms like SGD are capable of finding the global minimum of the (non-convex) training objective for feedforward NNs in polynomial time, provided that this NN is over-parameterized. This justifies the following assumption.

Assumption 3.1. *Optimization algorithms like SGD or batch-SGD can attain the global minimum of the loss functions (3.2) and (3.4) for over-parameterized NNs in polynomial time.*

We then recall the universal approximation theorem for feedforward NNs of Hornik et al. (1989).

Lemma 3.2. *For any continuous function $F(y)$ defined on a compact set $\bar{Q}$, it is possible to construct a feedforward NN with one hidden layer that uniformly approximates $F(y)$ on $\bar{Q}$ with arbitrarily small error $\varepsilon > 0$.*

This lemma justifies the following simplifying assumption, which will drastically lighten the proof of convergence of Algorithm 3 in Theorem 3.4.

Assumption 3.3. *The NN approximation errors ε from Lemma 3.2 are ignored when proving the convergence of Algorithm 3 in Theorem 3.4.*

We now present the main theoretical result of the paper showing that the value NN $V(t, x; \theta)$ and optimal control NN $u(t, x; \omega)$ both converge uniformly to an optimum. We consider a feed forward NN with one hidden layer of N neurons such that $V(t, x; \theta) = \sigma_{V_{out}} \left(\sum_{i=1}^N \beta_i^V \sigma_V \left(\alpha_{1,i}^V t + \sum_{j=1}^n \alpha_{j+1,i}^V x_j + c_i^V \right) \right) \in \mathbb{R}$, where σ_V is the value activation function, $\sigma_{V_{out}}$ the output activation function and $\theta \in \mathbb{R}^p$ the set of all parameters of the value NN with $p = N(n+2) + n$. For the control network, we assume $u(t, x; \omega) = \sigma_{u_{out}} \left(\sum_{i=1}^N \beta_i^u \sigma_u \left(\alpha_{1,i}^u t + \sum_{j=1}^n \alpha_{j+1,i}^u x_j + c_i^u \right) \right) \in \mathbb{R}^m$ with σ_u the control activation function, $\sigma_{u_{out}}$ the output activation function and $\omega \in \mathbb{R}^p$ the set of all parameters of the control NN.

Algorithm 3 GPI-PINN for implicit CT stochastic optimal control

Initialize admissible weights $\omega^{(0)}$ for the NN $u(\cdot; \omega^{(0)})$ and admissible weights $\theta^{(0)}$ for the NN $V(\cdot; \theta^{(0)})$.

Set the tolerance $\varepsilon \geq 0$ and let $k = 0$. Choose suitable learning rates $\eta_0, \eta_1, \eta_2 \geq 0$.

Phase 1 : Warm-up

Set $(\theta^{(1)}, \omega^{(1)}) \leftarrow (\theta^{(0)}, \omega^{(0)})$.

repeat

Update $(\theta^{(1)}, \omega^{(1)})$ using

$$\{\theta^{(1)}, \omega^{(1)}\} \leftarrow \{\theta^{(1)}, \omega^{(1)}\} - \eta_0 \frac{\mathbf{d}\mathcal{L}_1(\theta^{(1)}, \omega^{(1)})}{\mathbf{d}\{\theta^{(1)}, \omega^{(1)}\}} \quad (3.8)$$

until $\tilde{\mathcal{H}}(t, x, \theta^{(1)}, \omega^{(1)}) \leq 0$ for all $(t, x) \in Q$.

Phase 2 : GPI-PINN with early termination

while $\max_{(t,x) \in \bar{Q}} |V(t, x; \theta^{(k)}) - V(t, x; \theta^{(k-1)})| \geq \varepsilon$ **do**

Step 1. Update the value NN $V(\cdot; \theta^{(k+1)})$ for the known control network $u(\cdot; \omega^{(k)})$ at step k :

Set $\theta^{(k+1)} \leftarrow \theta^{(k)}$.

repeat

Update $\theta^{(k+1)}$ using

$$\theta^{(k+1)} \leftarrow \theta^{(k+1)} - \eta_1 \frac{\mathbf{d}\mathcal{L}_1(\theta^{(k+1)}, \omega^{(k)})}{\mathbf{d}\theta^{(k+1)}} \quad (3.9)$$

until $\tilde{\mathcal{H}}(t, x, \theta^{(k)}, \omega^{(k)}) \leq \tilde{\mathcal{H}}(t, x, \theta^{(k+1)}, \omega^{(k)}) \leq 0$ for all $(t, x) \in Q$.

Step 2. Update the control NN $u(\cdot; \omega^{(k+1)})$ for the known value network $V(\cdot; \theta^{(k+1)})$:

Set $\omega^{(k+1)} \leftarrow \omega^{(k)}$.

repeat

Update $\omega^{(k+1)}$ using

$$\omega^{(k+1)} \leftarrow \omega^{(k+1)} - \eta_2 \frac{\mathbf{d}\mathcal{L}_2(\theta^{(k+1)}, \omega^{(k+1)})}{\mathbf{d}\omega^{(k+1)}} \quad (3.10)$$

until $\tilde{\mathcal{H}}(t, x, \theta^{(k+1)}, \omega^{(k+1)}) \leq \tilde{\mathcal{H}}(t, x, \theta^{(k+1)}, \omega^{(k)})$ for all $(t, x) \in Q$.

$k \leftarrow k + 1$

end while

return $V(\cdot; \theta^{(k^*)})$ and $u(\cdot; \omega^{(k^*)})$ with $k^* \leftarrow k$.

Theorem 3.4. *Let us assume $\ell(t, x, u)$ and $\phi(x)$ non-negative, $\bar{Q}$ compact and that the Assumptions 3.1 and 3.3 hold. For an arbitrary control NN $u(t, x; \omega^{(0)})$ with initial weights $\omega^{(0)}$ and with a continuous discriminatory activation function $\sigma_u(\cdot)$ and an output activation function $\sigma_{u_{out}}(\cdot)$ such that $(u(t, X_t^u; \omega))_{0 \leq t \leq T} \in \mathcal{A}$ for all $\omega \in \mathbb{R}^p$, and for an arbitrary initial value NN $V(t, x; \theta^{(0)})$ with initial weight $\theta^{(0)}$ and with twice differentiable discriminatory activation functions $\sigma_V(\cdot)$ and $\sigma_{V_{out}}(\cdot)$ such that $\sigma_{V_{out}}(\cdot) \geq 0$, if these two NNs are tuned with Algorithm 3, then $V(t, x; \theta^{(k)}) \rightarrow V(t, x)$ uniformly on $\bar{Q}$ and $u(t, x; \omega^{(k)}) \rightarrow u^*(t, x)$ uniformly on Q as $k \rightarrow \infty$.*

The proof is delegated in Appendix 8.1. However, when using the sampling procedure with loss functions (3.6) and (3.7), it can be intractable to check the values of $\tilde{\mathcal{H}}(t, x, \omega, \theta)$ for every sampled points $\{(t_m^{(1)}, x_m^{(1)})\}_{m=1}^{M_1}$ when M_1 is large. In the practical applications below, we therefore use instead the expected value $\mathbb{E}_{(t,x) \sim \mu_1(Q)} [\tilde{\mathcal{H}}(t, x, \theta, \omega)]$ to verify that the early termination conditions in Algorithm 3 are satisfied, as suggested in Duan et al. (2023). This appears to provide fast and very accurate results numerically, as investigated in Section 5 and 6. As in Algorithm 2, the loss function $\mathcal{L}_{2'}$ given in (3.5) may also be used in place of (3.10) to accelerate convergence in certain empirical cases, although it does not guarantee to find the global minimizer u^* of the Hamiltonian. Moreover, Assumption 3.3 allows us to considerably simplify the exposition of the proof of Theorem 3.4. However, after closer inspection, it can be seen that Assumption 3.3 is not necessary and can be dropped from the proof straightforwardly, leading to the following Corollary 3.5. The weights $\theta_N^{(k)}$ (resp. $\omega_N^{(k)}$) now denote the NN value (resp. control) weights at step k for a feedforward NN with one hidden layer of N neurons.

Corollary 3.5. *Let the assumptions of Theorem 3.4 still hold except for Assumption 3.3. Then, $\forall \varepsilon > 0, \exists k^*, N^* : k > k^*, N_1 > N^*, N_2 > N^*$, one has*

$$\sup_{y \in \bar{Q}} |V(y; \theta_{N_1}^{(k)}) - V(y)| < \varepsilon \quad \text{and} \quad \sup_{y \in Q} |u(y; \omega_{N_2}^{(k)}) - u^*(y)| < \varepsilon.$$

Proof. The proof of Theorem 3.4 still holds using Lemma 3.2 when taking a sufficiently large number of neurons N_1 and N_2 with $\theta_{N_1}^{(k)}$ and $\omega_{N_2}^{(k)}$ and by introducing explicitly NN approximation errors ε . See also Theorems 2, 3, 4 of Abu-Khalaf and Lewis (2005).

4. Neural network implementation

Instead of the shallow NN of Theorem 3.4, we choose in practice the DGM architecture introduced in Sirignano and Spiliopoulos (2018) for both the optimal control NN and the value NN, as it empirically speeds up the convergence to the true solutions. It is inspired by LSTMs and Highway Networks introduced in Schmidhuber and Hochreiter (1997) and Srivastava et al. (2015), respectively. As pictured in Figure 4.1, each DGM layer takes as input the mini-batch inputs (t, x) (the set of randomly sampled time-space points from μ_1) and the output of the previous DGM layer. This DGM network then leads to a vector-valued

output consisting of the NN approximation of the desired function $V(\cdot; \theta)$ or $u(\cdot; \omega)$ evaluated at the mini-batch points. Following a similar rationale as in LSTMs, each layer produces weights based on the previous layer, determining how much information is transferred to the next layer, which allows to effectively address the issue of vanishing gradients while remaining flexible enough to model complex functions. Finally, we implement Algorithm 3 using TensorFlow and Keras, which are software libraries in Python with reverse mode automatic differentiation and which enable the exact calculation of derivatives for a broad range of functions (including the DGM network with respect to t , x , θ , and ω).

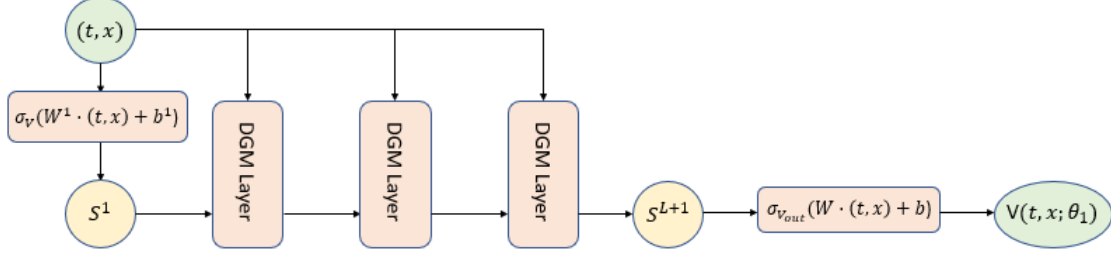


Figure 4.1: DGM architecture for the value neural network with $L = 3$ (i.e. 4 hidden layers).

More precisely, each DGM layer in Figure 4.1 for the value network is as follows

$$\begin{aligned}
 S^1 &= \sigma_V \left(W^1 \cdot (t, x) + b^1 \right), \\
 Z^\ell &= \sigma_V \left(U^{z,\ell} \cdot (t, x) + W^{z,\ell} \cdot S^\ell + b^{z,\ell} \right), \quad \ell = 1, \dots, L, \\
 G^\ell &= \sigma_V \left(U^{g,\ell} \cdot (t, x) + W^{g,\ell} \cdot S^1 + b^{g,\ell} \right), \quad \ell = 1, \dots, L, \\
 R^\ell &= \sigma_V \left(U^{r,\ell} \cdot (t, x) + W^{r,\ell} \cdot S^\ell + b^{r,\ell} \right), \quad \ell = 1, \dots, L, \\
 H^\ell &= \sigma_V \left(U^{h,\ell} \cdot (t, x) + W^{h,\ell} \cdot \left(S^\ell \odot R^\ell \right) + b^{h,\ell} \right), \quad \ell = 1, \dots, L, \\
 S^{\ell+1} &= \left(1 - G^\ell \right) \odot H^\ell + Z^\ell \odot S^\ell, \quad \ell = 1, \dots, L, \\
 V(t, x; \theta) &= \sigma_{V_{out}} \left(W \cdot S^{L+1} + b \right),
 \end{aligned}$$

where the number of hidden layers is $L + 1$, and $\odot$ denotes element-wise multiplication. The parameters are

$$\theta = \left\{ W^1, b^1, \left(U^{z,\ell}, W^{z,\ell}, b^{z,\ell} \right)_{\ell=1}^L, \left(U^{g,\ell}, W^{g,\ell}, b^{g,\ell} \right)_{\ell=1}^L, \left(U^{r,\ell}, W^{r,\ell}, b^{r,\ell} \right)_{\ell=1}^L, \left(U^{h,\ell}, W^{h,\ell}, b^{h,\ell} \right)_{\ell=1}^L, W, b \right\}.$$

The number of units in each layer is N , $\sigma_V : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is a twice-differentiable element-wise nonlinearity and $\sigma_{V_{out}} : \mathbb{R}^N \rightarrow \mathbb{R}^+$ is the output activation function (assumed to be non-negative and twice-differentiable). The control NN is designed following the same DGM architecture with $\sigma_{u_{out}} : \mathbb{R}^N \rightarrow \mathbb{R}^m$ locally bounded. In the examples below, we consider $L = 3$, i.e. 4 hidden layers, each one with $N = 50$ neurons. Unless otherwise stated, we take $M_1 = M_2 = 5000$ leading to a sample size $M = 10,000$, a batch size $b = 128$ and a maximum number of epochs $k^* = 500$. The network's parameters are updated using the Adaptive Moment Estimation with constant learning rates η_0, η_1, η_2 equal to 0.005. Weights and biases are initialized using the Glorot uniform initializer. Finally, the value activation function σ_V is the

$\tanh$ and $\sigma_{V_{out}}$ is the softplus function, as justified by Theorem 3.4. The control activation function σ_u is again the $\tanh$ and $\sigma_{u_{out}}$ the linear activation function.

5. A comparative example

We now evaluate the performance of our GPI-PINN Algorithm 3 on a test example with an analytically tractable solution. We consider the setting of Example 1 with a classical linear-quadratic Gaussian control problem in n dimensions with $Q = [0, T] \times \mathbb{R}^n$. We assume that the dimension m of the control is also n with $\mathbb{U} = \mathbb{R}^n$ and that $(W_t)_{t \in [0, T]}$ is a n -dimensional $\mathbb{F}$ -Brownian motion ($d = n$). We take as drift $f(t, x) = 2\sqrt{\lambda} I_n$ and diffusion coefficient $\sigma(t, x) = \sqrt{2} I_n$, where λ a positive constant representing the strength of the control. Finally, we suppose $R = I_n$, $h(t, x) = 0$ and $\phi(x) = \ln((1 + \|x\|^2)/2)$, so that

$$V^u(t, x) = \mathbb{E} \left[\int_t^T \|u(s, X_s^u)\|^2 ds + \ln((1 + \|X_T^u\|^2)/2) \mid X_t^u = x \right], \quad (5.1)$$

where X^u is the controlled process governed by

$$dX_s^u = 2\sqrt{\lambda} u_s ds + \sqrt{2} dW_s, \quad t \leq s \leq T, \quad X_t^u = x. \quad (5.2)$$

The cost-to-go (5.1) is indeed non-negative for any control $u \in \mathcal{A}$. The corresponding HJB equation for the value function $V(t, x) = \inf_{u \in \mathcal{A}} V^u(t, x)$ is given by

$$\begin{aligned} 0 &= \partial_t V(t, x) + \text{Tr}[V_{xx}(t, x)] + \inf_{u \in \mathbb{U}} \left\{ \|u\|^2 + 2\sqrt{\lambda} [V_x(t, x)]^\top u \right\} \\ \Leftrightarrow 0 &= \partial_t V(t, x) + \sum_{i=1}^n V_{x_i x_i}(t, x) + \inf_{u \in \mathbb{U}} \left\{ \sum_{i=1}^n (u^{(i)})^2 + 2\sqrt{\lambda} V_{x_i}(t, x) u^{(i)} \right\}, \end{aligned} \quad (5.3)$$

with terminal condition $V(T, x) = \phi(x)$. Hence, the Hamiltonian is the operator

$$\mathcal{H}(t, x, V_x, V_{xx}, u) = \|u\|^2 + 2\sqrt{\lambda} [V_x(t, x)]^\top u + \text{Tr}[V_{xx}(t, x)].$$

From Han et al. (2017), it holds that $V : \bar{Q} \rightarrow \mathbb{R}$ is $C^{1,2}(Q) \cap C(\bar{Q})$ and for $(t, x) \in \bar{Q}$ that

$$V(t, x) = -\frac{1}{\lambda} \ln \left(\mathbb{E} \left[\exp \left(-\lambda \phi(x + \sqrt{2} W_{T-t}) \right) \right] \right). \quad (5.4)$$

This representation then enables the use of Monte-Carlo (MC) simulations to compare the approximate solution from Algorithm 3 with the exact solution (5.4). Since V is smooth, the optimal control is given by

$$u^*(t, x) = -\sqrt{\lambda} V_x(t, x), \quad (5.5)$$

for all $(t, x) \in Q$, where $V_x = (V_{x_1}, \dots, V_{x_n})^\top$ and with

$$V_{x_i}(t, x) = \frac{2}{\mathbb{E} [\exp(-\lambda \phi(x + \sqrt{2} W_{T-t}))]} \mathbb{E} \left[\frac{\exp(-\lambda \phi(x + \sqrt{2} W_{T-t})) (x_i + \sqrt{2} W_{T-t}^{(i)})}{1 + \|x + \sqrt{2} W_{T-t}\|^2} \right],$$

for $i = 1, \dots, n$. Therefore, the n -dimensional optimal control u^* can also be approximated using MC samples. Moreover, as highlighted in Bachouch et al. (2022), the frequencies with which various states

appear in the sample should be roughly proportional to the probabilities of their occurrence under an optimal policy. As one can guess from the representation of $V^u(t, x)$ in equation (5.1), it is probably optimal to drive the process X^u around 0. Hence, since we have $\mu_1(t, x) = \mu_1(x|t)\mu_1(t)$, we assume $\mu_1(t) \sim \mathcal{U}_{[0, T]}$ and $\mu_1(x|t) \sim \sqrt{t}\mathcal{N}_n(\mathbf{0}_n, I_n)$ to simulate the values of x and t , with $\mathcal{U}$ a uniform law and $\mathcal{N}_n$ a n -dimensional Gaussian law. Similarly, $\mu_2(x) \sim \sqrt{T}\mathcal{N}_n(\mathbf{0}_n, I_n)$. This leads to the following plots of the value function and optimal control, comparing the numerical results obtained via the representation (5.4)-(5.5) using 10^6 MC samples with those produced by Algorithm 3, for various values of (t, x) with $T = 1$ and $\lambda = 1$. We consider a 10-dimensional problem ($n = 10$). Note that only the first component $u_1^*(t, x)$ of this 10-dimensional optimal control is depicted in the right plots of Figures 5.1-5.2. Moreover, the orange shaded area in both figures represents the standard deviation of the numerical solution computed on 10 independent runs of Algorithm 3.

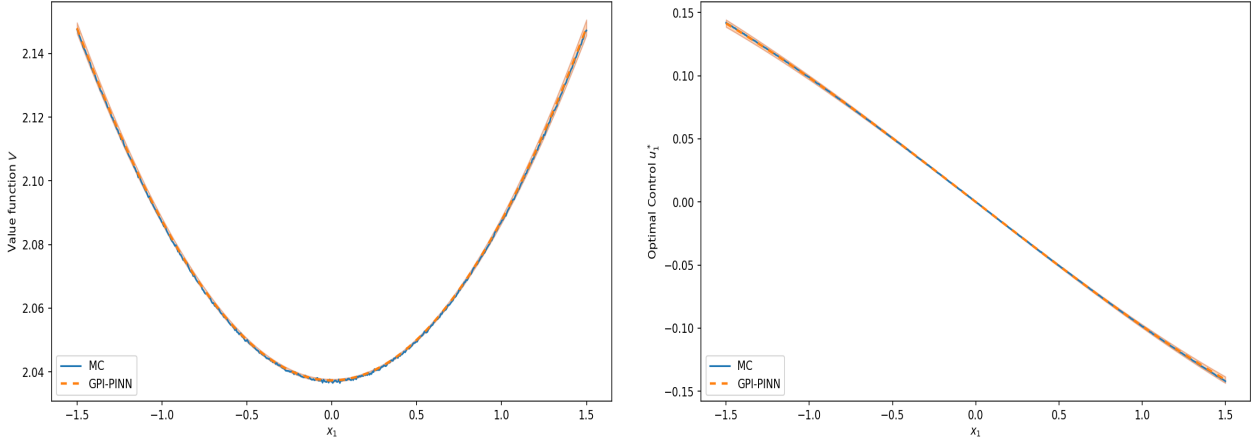


Figure 5.1: Value function $V(t, x)$ (left) and first component of the optimal control $u_1^*(t, x)$ (right) at $t = 0$ for $x = (x_1, \mathbf{0}_9)$ with $x_1 \in [-1.5, 1.5]$. In orange dotted line, numerical results from Algorithm 3 with $\pm$ one standard deviation in orange shaded area ($k^* = 500, M = 20,000$). In blue line, MC average of the solution (5.4)-(5.5).

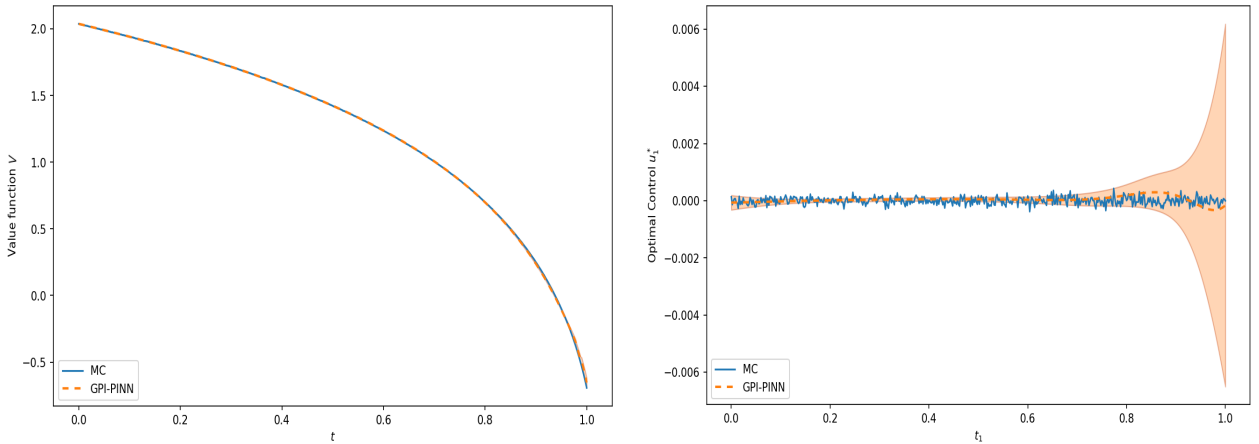


Figure 5.2: Optimal value function $V(t, x)$ (left) and first component of the optimal control $u_1^*(t, x)$ (right) at $x = \mathbf{0}_{10}$ with $t \in [0, 1]$. In orange dotted line, numerical results from Algorithm 3 with $\pm$ one standard deviation in orange shaded area ($k^* = 500, M = 20,000$). In blue line, MC average of the solution (5.4)-(5.5) with 10^6 samples.

We obtain highly accurate results using GPI-PINN 3 across all time and space, for both the value NN and the control NN. In the first Figure 5.1, we however observe a higher standard deviation for larger absolute values of x_1 (near ± 1.5) since we are sampling training observations using $\mu_1(t, x)$, from which the majority (around 85%) of observations from the training data falls within the interval $[-1, 1]$. Consequently, fewer observations are available for training the NNs at these larger absolute values of x_1 , which makes the NN outputs less accurate. Similarly, the standard deviation in the second Figure 5.2 is increasing with t since $\mu_1(x|t) \sim \sqrt{t}\mathcal{N}_n(\mathbf{0}_n, I_n)$ and hence, the higher the time t , the higher the variance of the conditional Gaussian distribution and the less the generated observations are close to the true value 0. Finally, we obtain the following numerical results in Tables 5.1 and 5.2 from the GPI-PINN approximation of $V(t, x)$ and $u_1^*(t, x)$ for $t = 0$ and $x = (0.5, \mathbf{0}_9)$. Again, $u_1^*(t, x)$ is the first component of the 10-dimensional optimal control $u^*(t, x)$. The standard deviations and means in both tables below are also obtained from 10 independent runs of Algorithm 3. The Mean Absolute Error (MAE) is computed between the mean of the value (resp. control) NN on these 10 independent runs and the mean of the 10^6 MC samples of representation (5.4) (resp. (5.5)). Finally, the loss $\mathcal{L}$ is defined as $\mathcal{L} := \frac{1}{2} \left(\widehat{\mathcal{L}}_1(\theta^{(k^*)}, \omega^{(k^*)}) + \widehat{\mathcal{L}}_2(\theta^{(k^*)}, \omega^{(k^*)}) \right)$, based on (3.6)-(3.7).

Table 5.1: Numerical results from the GPI-PINN Algorithm 3 for $V(0, (0.5, \mathbf{0}_9))$ and $u_1^*(0, (0.5, \mathbf{0}_9))$ with different sample size M and for a number of epochs $k^* = 500$.

Sample size	Mean V	Std. V	Mean u_1^*	Std. u_1^*	MAE $_V$	MAE $_{u_1^*}$	Mean $\mathcal{L}$	Std. $\mathcal{L}$	Runtime(s)
100	2.0621	0.01687	-0.07787	0.00739	0.06752	0.02738	-6.10	2.0761	769
1000	1.9956	0.0109	-0.04163	0.00675	0.05407	0.00887	-55.58	2.5810	1674
5000	2.0474	0.00325	-0.04842	0.00304	0.00304	0.00295	-80.13	0.6722	7168
10,000	2.0503	0.00162	-0.05111	0.00078	0.00131	0.00088	-81.40	0.2342	11,543
20,000	2.0499	0.00031	-0.05055	0.00059	0.00021	0.00005	-81.76	0.1946	29,454

Table 5.2: Numerical results from the GPI-PINN Algorithm 3 for $V(0, (0.5, \mathbf{0}_9))$ and $u_1^*(0, (0.5, \mathbf{0}_9))$ with various number of epochs k^* and for a fixed sample size $M = 10,000$.

Epochs	Mean V	Std. V	Mean u_1^*	Std. u_1^*	MAE $_V$	MAE $_{u_1^*}$	Mean $\mathcal{L}$	Std. $\mathcal{L}$	Runtime(s)
0	0.0908	0.18160	0.07019	0.11049	1.95890	0.14528	16 306 636	1 980 892	
50	1.9921	0.08227	-0.04185	0.00708	0.05754	0.00865	-49.6136	3.9522	2675
100	2.0213	0.01384	-0.04290	0.00623	0.02838	0.00759	-67.1391	3.2357	5501
500	2.0503	0.00162	-0.05111	0.00078	0.00131	0.00088	-81.3961	0.2342	11,543
1000	2.0503	0.00043	-0.05058	0.00045	0.00061	0.00038	-81.8925	0.2017	32,062

As expected, we clearly see that the higher the training size and the larger the number of epochs in the training procedure, the more accurate and precise are the outputs of the GPI-PINN Algorithm 3 for both the value function and the optimal control.

In Figure 5.3³, we now compare the accuracy of the GPI-PINN Algorithm 3 on the linear-quadratic Gaussian control (5.1) against two popular model-free RL algorithms with continuous action space, as well as the seminal time-discretized approach of Han and E (2016), based on a direct minimization of the empirical objective functional (5.1) using a single global network for the control. PPO and SAC perform the worst since they do not explicitly include the system dynamics (5.2) into the learning process and rely solely on sampling from the environment. The local method of Han and E gives great accuracy when applied at an initial pre-specified point $X_0^u = x_0 \in D$ (or along the optimal state trajectories $(X_t^{u*})_{t \in [0, T]}$, see Section 1 and 2.2), but does not generalize well when evaluated on a test set of $\mathcal{M}$ new points uniformly sampled over $[0, 1] \times [-2.5, 2.5]^n$, away from the initial state x_0 . Note that the mean absolute error for the value network is here defined as $MAE_V = \frac{1}{\mathcal{M}} \sum_{i=1}^{\mathcal{M}} |V_{\theta(k^*)}(t_i, x_i) - V(t_i, x_i)|$, with V given by the analytic formula (5.4). A second advantage of our approach is that it completely eliminates any time-discretization of the

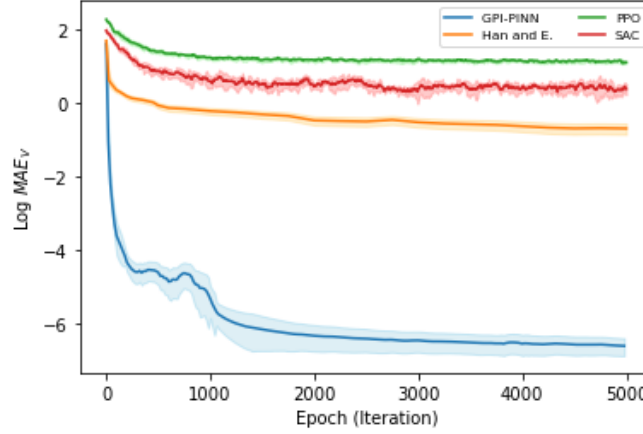


Figure 5.3: Comparison of $\log MAE_V$ in function of the epoch between the GPI-PINN Algorithm 3 and popular control methods.

state dynamics (2.1) (and the associated errors), thereby avoiding the sensitivity to the choice of time step Δt , which often has a significant impact on the performance of these local and semi-local algorithms. Finally, in these local methods, computing $V(t, x)$ requires each time to discretize and simulate the controlled state dynamics forward from t to T , whereas our framework enables fast, direct, and online evaluation of the value function across the entire domain. Altogether, these results validate the introduction of a second global network for approximating the value function.

Finally, we want to compare the accuracy and runtime between standard PINNs and our GPI-PINN Algorithm 3, when an explicit control is available. This will allow us to assess the price not to exploit this explicit control when available. From equation (5.5), we have that the optimal control in the linear-quadratic

³The four algorithms are configured in such a way to ensure roughly equal computing time per epoch/iteration in each method, so a similar plot would be obtained if we replaced the epochs by the actual running time in seconds.

regulator admits an explicit representation and we can then rewrite the HJB PDE (5.3) explicitly

$$0 = \partial_t V(t, x) + \sum_{i=1}^n (V_{x_i x_i}(t, x) - \lambda V_{x_i}^2(t, x)), \quad V(T, x) = \ln((1 + \|x\|^2)/2).$$

This explicit PDE enables the use of the (DGM) PINN algorithm of Sirignano and Spiliopoulos (2018), see Section 2.1. We therefore obtain in Figure 5.4 two plots comparing the accuracy in terms of MAE_V as well as the runtime between the DGM and the GPI-PINN algorithm in function of the training size and the number of epochs.

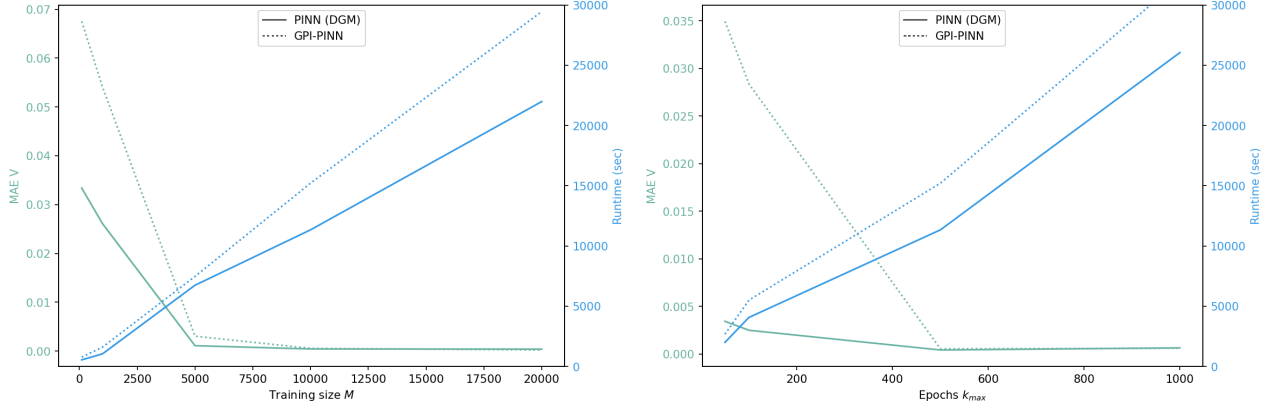


Figure 5.4: Comparison of the MAE_V (green) and the runtime in seconds (blue) between the DGM algorithm (solid line) and the GPI-PINN algorithm (dashed line) in function of the training size (left) and the number of epochs (right).

Clearly, the MAE_V for the value function V between the two methods converges when the number of epochs or the training size increases. In contrast, the higher the number of epochs or the training size, the more exacerbated is the difference in runtime between the two algorithms.

6. A practical example in mathematical finance

A classical problem in finance is how an agent can sell a large amount of shares and yet minimize adverse price movements generated by her own trades. The agent must then take into account the balance between price impact (trade quicker) and price risk (take longer to complete all trades). Indeed, if an agent trades quickly, then her orders will walk through the buy side of the limit order book (LOB) and she will obtain lower prices for her orders, leading to a *temporary* price impact. Even if she splits each order into smaller trades so that each one does not walk the book, then other traders will notice an excess of sell orders and adjust their quotes inducing again a negative price impact, which is in this case *permanent*. On the other hand, if she trades slowly so as to avoid this price impact, then she will face the uncertainty of future prices, which could have dropped to a less desirable level by the time the agent executes the whole order. The aim of optimal liquidation is hence to propose an optimal trading strategy such as to balance these two factors (price impact vs. market risk), based on a coherent price impact model reproducing the underlying asset price dynamics and the impact of the agent's transactions on this price. Following the setting of Cartea et al.

(2014) and Muhle-Karbe et al. (2024), we denote by T the finite trading horizon required for liquidating the shares. The trading rate $u = (u_t)_{0 \leq t \leq T}$ is the (positive) speed at which the agent is liquidating shares using market orders and which can be seen as the volume sold on each infinitesimal step of time. Q^u is the agent's inventory of remaining shares to liquidate, affected by how fast she trades and given by $dQ_t^u = -u_t dt$ with $Q_0^u = Q$. S^u is the midprice of the asset, affected by the speed of trading u and given by

$$dS_t^u = -g(u_t) dt + \sigma dW_t, \quad (6.1)$$

where $g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ denotes the permanent price impact of the agent's trading and $W = (W_t)_{0 \leq t \leq T}$ is a standard $\mathbb{F}$ -Brownian motion. The execution price $\hat{S}^u$ is then the price at which the agent can effectively sell the asset (also affected by the trading rate) :

$$\hat{S}_t^u = S_t^u - f(u_t), \quad (6.2)$$

where $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ denotes the temporary price impact that the agent's trading has on the execution price. Finally, X^u is the agent's cash process resulting from her execution strategy and defined by $X_t^u = \int_0^t \hat{S}_s^u u_s ds = \int_0^t (S_s^u - f(u_s)) u_s ds$. As classically proposed in the financial literature (Cartea et al., 2014), we assume that the agent is willing to maximize the following performance criterion

$$V^u(t, x, S, q) = \mathbb{E}_{t,x,S,q} \left[\underbrace{X_T^u}_{\text{Terminal Wealth}} + \underbrace{Q_T^u (S_T^u - \pi Q_T^u)}_{\text{Terminal Execution}} - \underbrace{\phi \int_t^T (Q_s^u)^2 ds}_{\text{Inventory Penalty}} \right], \quad (6.3)$$

with respect to the trading rate u , and where $\mathbb{E}_{t,x,S,q}[\cdot]$ denotes the expectation conditional to $X_t^u = x$, $S_t^u = S$, $Q_t^u = q$. The agent thus wants to optimize her expected wealth at maturity minus some penalties. If the agent's strategy reaches the terminal date T with inventory left, then she must execute an order to liquidate the remaining shares for a total revenue of $Q_T^u (S_T^u - \pi Q_T^u)$, where $\pi \geq 0$ is the terminal penalty parameter. We also introduce a running inventory penalty of the form $\phi \int_t^T (Q_s^u)^2 ds$ with $\phi \geq 0$. This running inventory penalty is not a financial cost to the agent's strategy but instead allows to incorporate the agent's urgency for executing the trade (risk-aversion penalty). The value function is then

$$V(t, x, S, q) = \sup_{u \in \mathcal{A}} V^u(t, x, S, q), \quad (6.4)$$

where $\mathcal{A}$ is the set of admissible strategies which consist in $\mathbb{F}$ -predictable, non-negative and bounded from above trading rates u . Standard results in dynamic programming (Cartea et al., 2015) imply that the value function V is a classical solution of the following HJB PDE for all $(t, S, q) \in [0, T] \times \mathbb{R} \times (0, Q]$:

$$0 = \left(\partial_t + \frac{1}{2} \sigma^2 \partial_{SS} \right) V - \phi q^2 + \sup_{u \in \mathcal{A}} \{ (u(S - f(u)) \partial_x - g(u) \partial_S - u \partial_q) V \}, \quad (6.5)$$

with terminal condition $V(T, x, S, q) = x + q(S - \pi q)$ and boundary condition $V(t, x, S, 0) = 0$ since there are no more shares to liquidate and since the agent cannot buy back shares (non-negative trading rate).

Using integration by part, it is easy to show that

$$\begin{aligned}
V(t, x, S, q) &= x + Sq + \sup_{u \in \mathcal{A}} \mathbb{E}_{t,q} \left[- \int_t^T f(u_s) u_s ds - \int_t^T g(u_s) Q_s^u ds - \pi (Q_T^u)^2 - \phi \int_t^T (Q_s^u)^2 ds \right] \\
&= x + Sq - \inf_{u \in \mathcal{A}} \mathbb{E}_{t,q} \left[\int_t^T f(u_s) u_s ds + \int_t^T g(u_s) Q_s^u ds + \pi (Q_T^u)^2 + \phi \int_t^T (Q_s^u)^2 ds \right] \\
&= x + Sq - \mathcal{V}(t, q),
\end{aligned} \tag{6.6}$$

where $\mathcal{V}(t, q)$ is again a value function but which now states for the penalty stemming from optimally liquidating the remaining of the shares. It is clearly non-negative and it is again a classical solution of the following HJB PDE for all $(t, q) \in [0, T) \times (0, Q]$:

$$0 = \partial_t \mathcal{V}(t, q) + \phi q^2 + \inf_{u \in \mathcal{A}} \{ u f(u) + g(u) q - u \partial_q \mathcal{V}(t, q) \}, \tag{6.7}$$

with $\mathcal{V}(T, q) = \pi q^2$ and $\mathcal{V}(t, 0) = 0$. Let us assume that $u f(u)$ and $g(u)$ are convex functions in u . Almgren et al. (2005), Gatheral (2010) and Cartea et al. (2014) have shown that a power-law function with exponent less than one is the most suitable choice for the temporary price impact $f(u) = \kappa u^\gamma$ with $\kappa \geq 0, \gamma \in [0, 1]$. The permanent impact function is also here assumed to be a power-law function of the form $g(u) = b u^\eta$ with $b \geq 0, \eta \geq 1$, which is consistent with Figure 1 of Cartea and Jaimungal (2016). This allows us to generalize the classical models of Almgren and Chriss (2001) and Almgren (2003) by considering both nonlinear temporary and permanent impact functions. We finally obtain that the value function $\mathcal{V}$ must satisfy for all $(t, q) \in [0, T) \times (0, Q]$:

$$0 = \partial_t \mathcal{V}(t, q) + \phi q^2 + \inf_{u \in \mathcal{A}} \{ \kappa u^{\gamma+1} + b u^\eta q - u \partial_q \mathcal{V}(t, q) \}, \tag{6.8}$$

with $\mathcal{V}(T, q) = \pi q^2$ and $\mathcal{V}(t, 0) = 0$. For general exponents $\gamma \in [0, 1]$ and $\eta \geq 1$, we directly see from (6.8) that the first order condition, $\kappa(\gamma + 1)u^\gamma + b\eta u^{\eta-1}q - \partial_q \mathcal{V}(t, q) = 0$, does not lead to an explicit solution for the optimal control u^* , requiring the use of the GPI-PINN Algorithm 3. In the classical Almgren-Chriss model $\gamma = 1$ and $\eta = 1$ studied in Almgren and Chriss (2001) and Cartea et al. (2014), it is however shown that

$$\mathcal{V}(t, q) = q^2 \left(-\frac{1}{2}b + \chi(t) \right), \tag{6.9}$$

with $\chi(t) = \sqrt{\kappa\phi} \frac{1+\zeta e^{2\rho(T-t)}}{1-\zeta e^{2\rho(T-t)}}$, $\rho = \sqrt{\phi/\kappa}$ and $\zeta = \frac{(\pi-0.5b+\sqrt{\kappa\phi})}{(\pi-0.5b-\sqrt{\kappa\phi})}$. Moreover, this leads to

$$u^*(t, q) = \gamma \frac{\zeta e^{\rho(T-t)} + e^{-\rho(T-t)}}{\zeta e^{\rho(T-t)} - e^{-\rho(T-t)}} q. \tag{6.10}$$

We can therefore compare these analytical solutions of the value function (6.9) and optimal control (6.10) with $\gamma = 1$ and $\eta = 1$ to the numerical results output by our GPI-PINN Algorithm 3, as depicted in Figures 6.1 and 6.2.

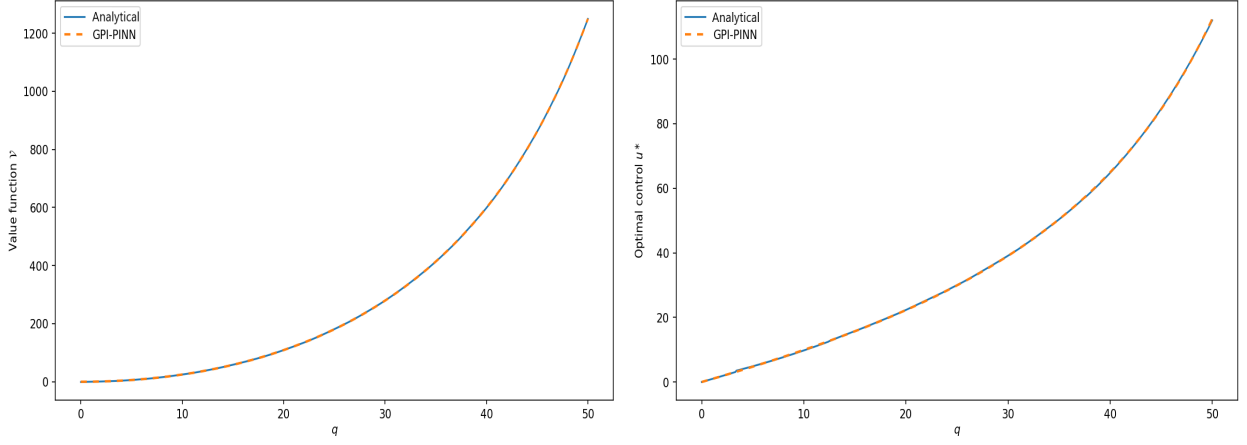


Figure 6.1: Value function $\mathcal{V}(t, q)$ and optimal control $u^*(t, q)$ at $t = 0$ for $q \in [0, 50]$ with $\gamma = 1, \eta = 1, T = 1, \kappa = 0.2, b = 0.1, \pi = 0.5, \phi = 0.1$. In orange dotted line, numerical solution from Algorithm 3 ($k^* = 500, M = 20,000$). In blue line, analytical solutions (6.9)-(6.10).

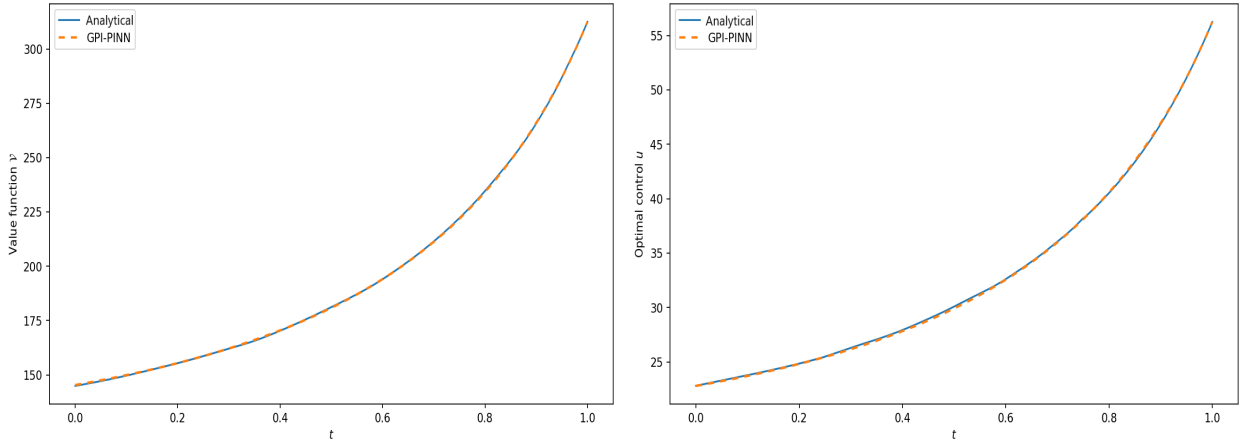


Figure 6.2: Value function $\mathcal{V}(t, q)$ and optimal control $u^*(t, q)$ at $q = 25$ for $t \in [0, 1]$, with $\gamma = 1, \eta = 1, T = 1, \kappa = 0.2, b = 0.1, \pi = 0.5, \phi = 0.1$. In orange dotted line, numerical solution from Algorithm 3 ($k^* = 500, M = 20,000$). In blue line, analytical solutions from equations (6.9)-(6.10).

In this one-dimensional optimal liquidation setting, we clearly see that the GPI-PINN algorithm manages to perfectly approximate both the value function and the optimal control across all time and space. The following Figure 6.3 compares the analytical solution in the case $\gamma = \eta = 1$ to the implicit case $\gamma = 0.7, \eta = 1.5$ solved using the GPI-PINN 3, allowing us to assess the impact of the parameters γ and η on the optimal strategies.

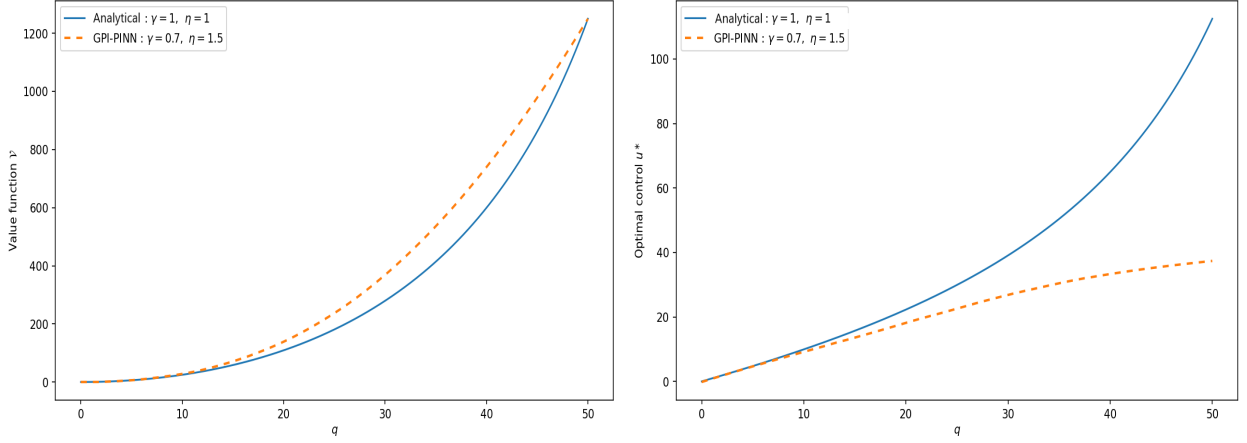


Figure 6.3: Value function $\mathcal{V}(t, q)$ and optimal control $u^*(t, q)$ at $t = 0$ for $q \in [0, 50]$ with $T = 1, \kappa = 0.2, b = 0.1, \pi = 0.5, \phi = 0.1$. In orange dotted line, numerical solution from Algorithm 3 with $\gamma = 0.7, \eta = 1.5$ ($k^* = 500, M = 20,000$). In blue line, analytical solution from (6.9)-(6.10) with $\gamma = 1, \eta = 1$.

6.1. Multi-asset optimal liquidation

We now consider a multi-asset optimal liquidation setting since the (G)PI-PINNs have been especially devised for solving high-dimensional (implicit) optimal control problems. The studied financial portfolio now consists in n assets which are co-integrated in the sense that each asset's trading permanently affects the price at which all the other assets are executed. Indeed, in this setting, market sell orders in one stock also exert a permanent downward pressure on the price of co-integrated assets. As explained in Cartea et al. (2019) and Li et al. (2022), this cross-effect is partly caused by trading algorithms that take positions based on the co-movements of assets. We also include correlations between each pair of assets in the portfolio. Therefore, the above one-dimensional optimal liquidation model is enlarged in the following way. The trading rate in the i^{th} stock is denoted $u^{(i)} = (u_t^{(i)})_{0 \leq t \leq T}$ and the related remaining inventory satisfies $dQ_t^{u,i} = -u_t^{(i)} dt$, for all $i = 1, \dots, n$. We also denote $\mathbf{u}_t = (u_t^{(1)}, \dots, u_t^{(n)})^\top$ and $\mathbf{Q}_t^u = (Q_t^{u,1}, \dots, Q_t^{u,n})^\top$. The midprice of the i^{th} stock is the i^{th} component of $\mathbf{S}_t^u = (S_t^{u,1}, \dots, S_t^{u,n})^\top$, which follows $d\mathbf{S}_t^u = -\mathbf{g}(\mathbf{u}_t)dt + \boldsymbol{\sigma}d\mathbf{W}_t$, so that the dynamic of the i^{th} stock is given by

$$dS_t^{u,i} = -g^{(i)}(\mathbf{u}_t)dt + \sum_{k=1}^i \sigma_{ik} dW_t^{(k)}, \quad (6.11)$$

where the function $g^{(i)} : \mathbb{R}_+^n \rightarrow \mathbb{R}_+$ describes the permanent impact of each asset's trading on the midprice of the i^{th} asset. We also have that $\boldsymbol{\sigma} \in \mathbb{R}^{n \times n}$ is the (upper) Cholesky decomposition of the asset prices' correlation matrix $\tilde{\boldsymbol{\Sigma}}$ (i.e., $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\sigma}\boldsymbol{\sigma}^\top$) and $\mathbf{W} = (\mathbf{W}_t)_{0 \leq t \leq T}$ is a n -dimensional $\mathbb{F}$ -Brownian motion. The execution price of the i^{th} asset then follows

$$\hat{S}_t^{u,i} = S_t^{u,i} - f^{(i)}(u_t^{(i)}),$$

where the function $f^{(i)} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ describes the temporary impact of trading in stock i , which only affects in this case the execution price of the corresponding asset i . Indeed, as the temporary price impact for asset

i comes from the market orders walking the LOB of this specific asset, it does not induce cross-effects from other assets. The wealth of the agent from liquidating the portfolio is given by $X_t^u = \sum_{i=1}^n \int_0^t \widehat{S}_s^{u,i} u_s^{(i)} ds$. As in the one-dimensional setting (6.3), we want to optimize the following performance criterion with respect to the vector of trading rates $\mathbf{u}$:

$$V^u(t, x, \mathbf{S}, \mathbf{q}) = \mathbb{E}_{t,x,\mathbf{S},\mathbf{q}} \left[X_T^u + (\mathbf{Q}_T^u)^\top \mathbf{S}_T^u - \pi(\mathbf{Q}_T^u)^\top \mathbf{Q}_T^u - \phi \int_t^T (\mathbf{Q}_s^u)^\top \widetilde{\Sigma} \mathbf{Q}_s^u ds \right] \quad (6.12)$$

conditionally on $X_t^u = x$, $\mathbf{S}_t^u = \mathbf{S} := (S_1, \dots, S_n)^\top$ and $\mathbf{Q}_t^u = \mathbf{q} := (q_1, \dots, q_n)^\top$. The value function is then $V(t, x, \mathbf{S}, \mathbf{q}) = \sup_{\mathbf{u} \in \mathcal{A}} V^u(t, x, \mathbf{S}, \mathbf{q})$ with $\mathcal{A}$ the set of admissible strategies consisting in non-negative, $\mathbb{F}$ -predictable processes such that $\int_0^T |u_s^{(i)}| ds < +\infty$, $\mathbb{P}$ -a.s. for each asset $i = 1, \dots, n$. Using integration by parts as in (6.6), we can show that

$$\begin{aligned} V(t, x, \mathbf{S}, \mathbf{q}) &= x + \mathbf{q}^\top \mathbf{S} + \sup_{\mathbf{u} \in \mathcal{A}} \mathbb{E}_{t,\mathbf{q}} \left[-\pi(\mathbf{Q}_T^u)^\top \mathbf{Q}_T^u - \phi \int_t^T (\mathbf{Q}_s^u)^\top \widetilde{\Sigma} \mathbf{Q}_s^u ds \right. \\ &\quad \left. - \sum_{i=1}^n \left(\int_t^T f^{(i)}(u_s^{(i)}) u_s^{(i)} ds + \int_t^T g^{(i)}(\mathbf{u}_s) Q_s^{u,i} ds \right) \right] \\ &= x + \mathbf{q}^\top \mathbf{S} - \inf_{\mathbf{u} \in \mathcal{A}} \mathbb{E}_{t,\mathbf{q}} \left[\pi(\mathbf{Q}_T^u)^\top \mathbf{Q}_T^u + \phi \int_t^T (\mathbf{Q}_s^u)^\top \widetilde{\Sigma} \mathbf{Q}_s^u ds \right. \\ &\quad \left. + \sum_{i=1}^n \left(\int_t^T f^{(i)}(u_s^{(i)}) u_s^{(i)} ds + \int_t^T g^{(i)}(\mathbf{u}_s) Q_s^{u,i} ds \right) \right] \\ &= x + \mathbf{q}^\top \mathbf{S} - \mathcal{V}(t, \mathbf{q}). \end{aligned}$$

The dynamic programming principle suggests that the value function $\mathcal{V}$ is a classical solution to the HJB equation for all $(t, \mathbf{q}) \in [0, T) \times ([0, Q]^n \setminus \mathbf{0}_n)$:

$$0 = \partial_t \mathcal{V}(t, \mathbf{q}) + \phi \mathbf{q}^\top \widetilde{\Sigma} \mathbf{q} + \inf_{\mathbf{u} \in \mathcal{A}} \left\{ \sum_{i=1}^n \left(f^{(i)}(u^{(i)}) u^{(i)} + g^{(i)}(\mathbf{u}) q_i - u^{(i)} \partial_{q_i} \mathcal{V}(t, \mathbf{q}) \right) \right\}, \quad (6.13)$$

with terminal and boundary conditions $\mathcal{V}(T, \mathbf{q}) = \pi \mathbf{q}^\top \mathbf{q}$ and $\mathcal{V}(t, \mathbf{0}_n) = 0$. As motivated in the one-dimensional setting, we choose the following power-law functions for the permanent and temporary impact functions $g^{(i)}$ and $f^{(i)}$ with $\gamma_i \in [0, 1]$, $\kappa_i \geq 0$, $\eta_{i,j} \geq 1$, $b_{i,j} \geq 0$ for all $i, j = 1, \dots, n$:

$$g^{(i)}(\mathbf{u}) = \sum_{j=1}^n b_{i,j} \left(u^{(j)} \right)^{\eta_{i,j}} \quad \text{and} \quad f^{(i)}(u^{(i)}) = \kappa_i \left(u^{(i)} \right)^{\gamma_i}.$$

With this parametrization, the permanent impact on the transaction price of asset i indeed depends on the trading rate $u^{(j)}$ of all assets $j = 1, \dots, n$, whereas the temporary impact depends solely on the trading rate $u^{(i)}$ of the corresponding asset. Equation (6.13) hence becomes

$$0 = \partial_t \mathcal{V}(t, \mathbf{q}) + \phi \mathbf{q}^\top \widetilde{\Sigma} \mathbf{q} + \inf_{\mathbf{u} \in \mathcal{A}} \left\{ \sum_{i=1}^n \left(\kappa_i \left(u^{(i)} \right)^{\gamma_i+1} + \sum_{j=1}^n b_{i,j} \left(u^{(j)} \right)^{\eta_{i,j}} q_i - u^{(i)} \partial_{q_i} \mathcal{V}(t, \mathbf{q}) \right) \right\}. \quad (6.14)$$

First order condition on $u^{(i)}$ with $\eta_{i,j} \geq 1$, $\gamma_i \in [0, 1]$ leads for all $i = 1, \dots, n$ to

$$\kappa_i(\gamma_i + 1) \left(u^{(i)}\right)^{\gamma_i} + \sum_{j=1}^n b_{j,i} \eta_{j,i} \left(u^{(i)}\right)^{\eta_{j,i}-1} q_j - \partial_{q_i} \mathcal{V}(t, \mathbf{q}) = 0. \quad (6.15)$$

We see that if $\eta_{j,i} = 1$ for all j (*i.e.* linear permanent impacts), then the optimal trading rate $u^{(i)*}$ can be obtained explicitly for all $i = 1, \dots, n$ by

$$u^{(i)*}(t, \mathbf{q}) = \left(\frac{-\sum_{j=1}^n b_{j,i} q_j + \partial_{q_i} \mathcal{V}(t, \mathbf{q})}{\kappa_i(\gamma_i + 1)} \right)^{1/\gamma_i}. \quad (6.16)$$

In this case, we can plug back this explicit optimal control (6.16) inside equation (6.14) and solve the associated HJB PDE for $\mathcal{V}$ using finite differences when the dimensionality n is not too large. Setting $n = 3$ with the following parameters allows us to compare the accuracy of our GPI-PINN algorithm 3 to the standard finite-difference method and confirm its numerical convergence: $T = 1$, $\pi = 0.5$, $\phi = 0.1$, $\gamma = (1, 1, 1)$, $\kappa = (0.2, 0.1, 0.15)$, and

$$\boldsymbol{\eta} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0.1 & 0.03 & 0.02 \\ 0.01 & 0.24 & 0 \\ 0.01 & 0.02 & 0.05 \end{pmatrix}, \quad \tilde{\boldsymbol{\Sigma}} = \begin{pmatrix} 1 & 0.2 & 0.13 \\ 0.2 & 1 & 0.05 \\ 0.13 & 0.05 & 1 \end{pmatrix}.$$

The following Figure 6.4 then confirms the numerical convergence of the GPI-PINN algorithm and gives further confidence in our numerical results, as both methods produce nearly identical solutions for the value function $\mathcal{V}$.

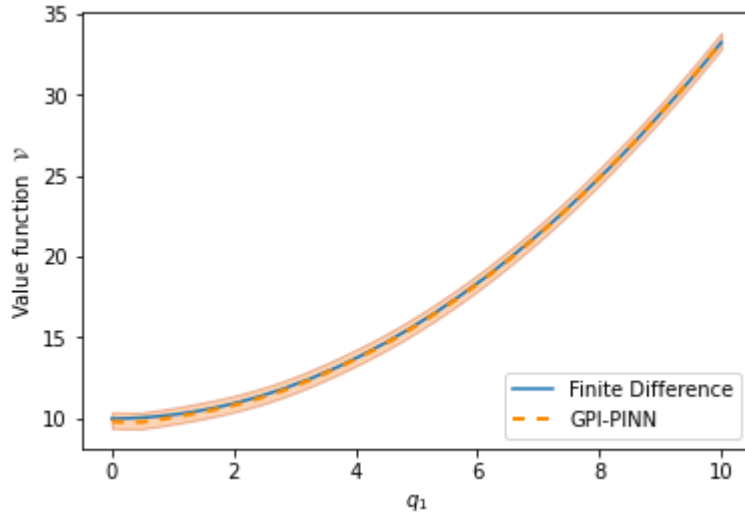


Figure 6.4: Value function $\mathcal{V}(t, \mathbf{q})$ at $t = 0$ and $\mathbf{q} = (q_1, 5, 5)$ for $q_1 \in [0, 10]$. In orange dotted line, numerical results from Algorithm 3 with $\pm$ one standard deviation in orange shaded area ($k^* = 500$, $M = 20,000$). In blue line, numerical solution from finite (central) differences.

However, for general exponents $\eta_{j,i} \geq 1$, we have from the first-order condition (6.15) that the optimal trading rate $u^{(i)*}$ for each asset $i = 1, \dots, n$ does not admit a closed-form solution. We then again face an

implicit high-dimensional HJB equation (when n is large), requiring the use of the GPI-PINN Algorithm

3. We then obtain the following results for a portfolio of $n = 10$ stocks.

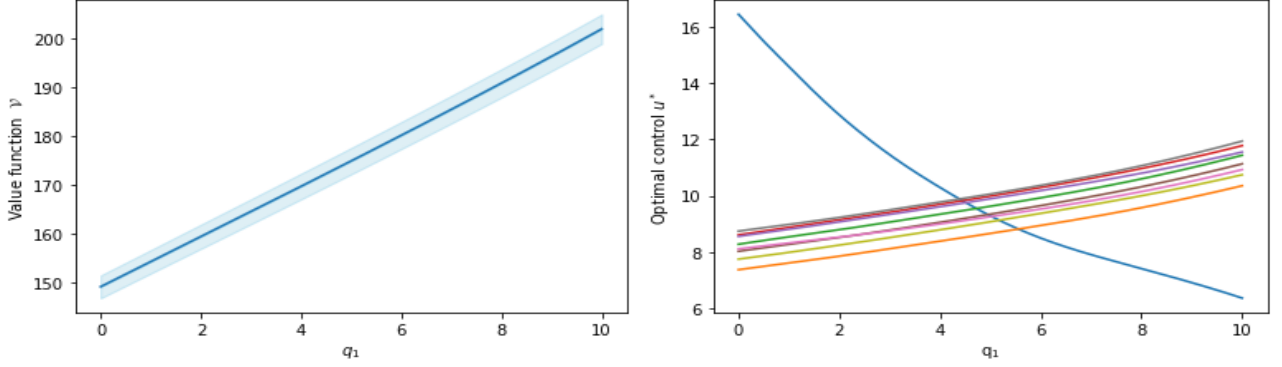


Figure 6.5: Value function $\mathcal{V}(t, \mathbf{q})$ (left plot) and optimal trading rate $\mathbf{u}^*(t, \mathbf{q})$ (right plot) at $t = 0$ and $\mathbf{q} = (q_1, 5 \cdot \mathbb{1}_9)$ for $q_1 \in [0, 10]$ with $n = 10$ stocks from Algorithm 3. In the left plot, $\pm$ one standard deviation in blue shaded area from 10 independent runs of Algorithm 3 ($k^* = 2000, M = 20,000$).

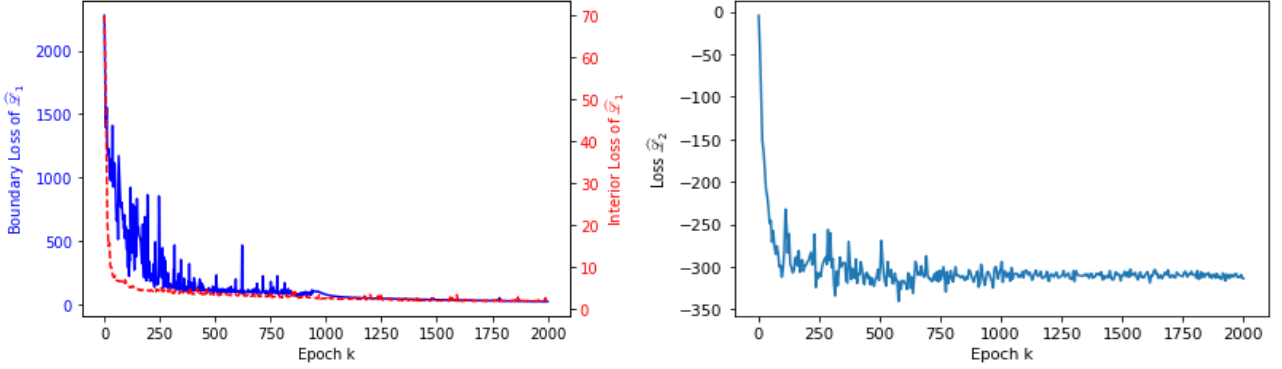


Figure 6.6: Losses $\widehat{\mathcal{L}}_1$ (left plot) and $\widehat{\mathcal{L}}_2$ (right plot) of Algorithm 3 in function of the epoch k . The red curve in the left plot represents the interior loss of $\widehat{\mathcal{L}}_1$ and the blue curve its boundary part, see equation (3.6). Note here that the loss $\widehat{\mathcal{L}}_2$ depicted in the right plot does not include the terms $\partial_t \mathcal{V}(t, \mathbf{q}) + \phi \mathbf{q}^\top \widetilde{\Sigma} \mathbf{q}$ as they do not depend on the control u .

Despite the lack of reference for comparison, Figures 6.5 and 6.6 lead to consistent numerical results, giving us confidence in the GPI-PINN 3 for multidimensional optimal execution problems. In particular, when varying the first dimension q_1 of the remaining quantity $\mathbf{q}$, the main impact is on the corresponding optimal trading rate u_1^* . We then also confirm with Figure 6.6 that the loss $\widehat{\mathcal{L}}_1$ (3.6) converges to 0 and that we are minimizing the average Hamiltonian through the loss $\widehat{\mathcal{L}}_2$ (3.7), as the number of epoch k increases. Note that the red curve in the left plot represents the interior loss (first right-hand term) of $\widehat{\mathcal{L}}_1$ and the blue curve the boundary loss (second right-hand term) of $\widehat{\mathcal{L}}_1$, see equation (3.6).

7. Conclusion

This paper proposes a new iterative deep-learning algorithm able to solve CT stochastic optimal control problems in high dimensions, even when the control does not admit an explicit representation. This numerical scheme is built from the combination of PINNs on the HJB residuals with an Actor-Critic structure,

relying on the GPI procedure. By approximating the value function and the optimal control using two separate NNs, the GPI-PINN algorithm provides a global solution for stochastic optimal control problems across all time and space, which can be evaluated online fast. Thanks to an early termination scheme in the training process, we prove the uniform convergence of our algorithm toward the true value function and optimal control. Besides these theoretical guarantees, we also show empirically the efficacy and accuracy of the GPI-PINN scheme thanks to a comparative example for which the solution is known up to MC simulations. Finally, we provide a second practical example from mathematical finance by generalizing the well-known Almgren-Chriss model arising from optimal liquidation. More precisely, we introduce optimal liquidation strategies based on fully nonlinear temporary and permanent impact functions in a multidimensional setting with co-integrated assets, leading to an implicit optimal control problem in high dimensions. Once again, this example highlights the benefits of using the GPI-PINN algorithm to solve implicit high-dimensional CT stochastic optimal control problems and paves the way for various other applications in stochastic optimal control.

8. Appendix

8.1. Proof of Theorem 3.4

First part. We first introduce some notations and show that the modified Hamiltonian (2.10) satisfies a non-linear Lyapunov equation. Recall that the cost-to-go associated with the non-optimal admissible Markov control u is defined as

$$V^u(t, x) = \mathbb{E}^{\mathbb{P}} \left[\phi(X_T^u) + \int_t^T \ell(\tau, X_\tau^u, u(\tau, X_\tau^u)) d\tau \mid X_t^u = x \right]. \quad (8.1)$$

As in Oksendal (2013), we propose the change of variable $y = (t, x)$ and $Y_\tau^u = (\tau, X_\tau^u)$ so as to fall back to the time-invariant optimal control problem

$$V^u(y) = \mathbb{E}_y^{\mathbb{P}} \left[\phi(Y_T^u) + \int_t^T \ell(Y_\tau^u, u(Y_\tau^u)) d\tau \right],$$

with $\mathbb{E}_y^{\mathbb{P}}$ the expectation conditional to $Y_t^u = y$ under the probability measure $\mathbb{P}$ and with the dynamic

$$dY_\tau^u = \left(d\tau, f(Y_\tau^u, u(Y_\tau^u))d\tau + \sigma(Y_\tau^u, u(Y_\tau^u))dW_\tau^{\mathbb{P}} \right). \quad (8.2)$$

Since $V^u \in C^2(Q)$, using the dynamic programming principle and Dynkin's formula

$$\mathbb{E}_y^{\mathbb{P}} [V^u(Y_{t+h}^u)] = V^u(y) + \mathbb{E}_y^{\mathbb{P}} \left[\int_t^{t+h} (\mathcal{L}^u V^u)(Y_\tau^u) d\tau \right],$$

where $\mathcal{L}$ is the infinitesimal generator associated with the diffusion (8.2),

$$(\mathcal{L}^u \phi)(y) := \frac{\partial \phi}{\partial t}(y) + \sum_{i=1}^n f_i(y, u(y)) \frac{\partial \phi}{\partial x_i}(y) + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\sigma \sigma^\top)_{ij}(y, u(y)) \frac{\partial^2 \phi}{\partial x_i \partial x_j}(y), \quad (8.3)$$

we have, see Touzi (2012), Oksendal (2013), that V^u satisfies the following nonlinear Lyapunov equation for all $y \in Q$,

$$0 = \ell(y, u(y)) + (\mathcal{L}^u V^u)(y). \quad (8.4)$$

The notation (2.4) for the Hamiltonian $\mathcal{H}(t, x, V_x^u(t, x), V_{xx}^u(t, x), u(t, x))$ is simplified as $\mathcal{H}(y, V^u(y), u(y))$ and hence from (8.4), equation (2.10) becomes $\forall y \in Q$,

$$\tilde{\mathcal{H}}(y, V^u(y), u(y)) := \mathcal{H}(y, V^u(y), u(y)) + \frac{\partial V^u}{\partial t}(y) = \ell(y, u(y)) + (\mathcal{L}^u V^u)(y) = 0.$$

When considering⁴ the NNs $V_{\theta^{(k)}}(y) := V(y; \theta^{(k)})$ and $u_{\omega^{(k)}}(y) := u(y; \omega^{(k)})$, we again use the notation

$$\tilde{\mathcal{H}}(y, \theta^{(k)}, \omega^{(k)}) := \tilde{\mathcal{H}}(y, V_{\theta^{(k)}}(y), u_{\omega^{(k)}}(y)),$$

which is in general not equal to 0 since $V_{\theta^{(k)}}(y)$ is not governed by the control $u_{\omega^{(k)}}(y)$, *i.e.* two separate NNs. Notice that we have $\left(u(t, X_t^u; \omega)\right)_{0 \leq t \leq T} \in \mathcal{A}$ for all $\omega \in \mathbb{R}^p$ from the assumptions of Theorem 3.4. Then, since $\sigma_{V_{out}}(\cdot) \geq 0$, we have $V_{\theta}(y) \geq 0$ for all $y \in \bar{Q}$ and all $\theta \in \mathbb{R}^p$. Moreover, since $\sigma_V(\cdot)$ and $\sigma_{V_{out}}(\cdot)$ are twice differentiable, we have $V_{\theta}(y) \in C^2(\bar{Q})$. Finally, Assumption 3.3 allows us to ignore the NN approximation error ε when approximating $\phi(x)$ by $V_{\theta}(T, x)$. The loss function (3.2) then becomes

$$\begin{aligned} \mathcal{L}_1(\theta, \omega) &= \mathbb{E}_{y \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(y, \theta, \omega)^2 \right] + \mathbb{E}_{x \sim \mu_2(\mathbb{X})} \left[(V_{\theta}(T, x) - \phi(x))^2 \right] \\ &= \mathbb{E}_{y \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(y, \theta, \omega)^2 \right], \end{aligned} \quad (8.5)$$

and the loss (3.4) becomes

$$\mathcal{L}_2(\theta, \omega) = \mathbb{E}_{y \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(y, \theta, \omega) \right].$$

Second part. We then show that the early termination conditions in Step 1 and 2 can effectively be satisfied based on the assumptions of Theorem 3.4 when using Algorithm 3. According to (8.4), Lemma 3.2⁵ and Assumption 3.3, there exists $(\theta^\dagger, \omega^\dagger)$ such that $\tilde{\mathcal{H}}(y, \theta^\dagger, \omega^\dagger) = 0$ for all $y \in Q$. If $V_{\theta^\dagger}$ is the value function V , then from Theorem 2.2, $u_{\omega^\dagger}(y) = u^*(y) = \arg \min_u \mathcal{H}(y, V_{\theta^\dagger}(y), u)$ for all $y \in Q$ (and we are done). If $V_{\theta^\dagger}$ is not the value function, then $u_{\omega^\dagger}(y) > \arg \min_u \mathcal{H}(y, V_{\theta^\dagger}(y), u)$ and $\tilde{\mathcal{H}}(y, \theta^\dagger, \omega^\dagger) > \tilde{\mathcal{H}}(y, \theta^\dagger, u^*(y))$ for all $y \in Q$. Therefore, we can find $\omega^\ddagger$ such that

$$0 = \tilde{\mathcal{H}}(y, \theta^\dagger, \omega^\dagger) > \tilde{\mathcal{H}}(y, \theta^\dagger, \omega^\ddagger) \geq \tilde{\mathcal{H}}(y, \theta^\dagger, u^*(y)), \quad \forall y \in Q.$$

From both alternatives above, there exist $(\theta^\dagger, \omega^\ddagger)$ with $\tilde{\mathcal{H}}(y, \theta^\dagger, \omega^\ddagger) \leq 0$ for all $y \in Q$, implying $\mathcal{L}_2(\theta^\dagger, \omega^\ddagger) \leq 0$ and that the global minimum $\min_{\theta, \omega} \mathcal{L}_2(\theta, \omega) \leq 0$ is such that $\tilde{\mathcal{H}}(y, \theta, \omega) \leq 0 \forall y \in Q$. Since the global minimum of $\mathcal{L}_2(\theta, \omega)$ with respect to (θ, ω) can be attained in polynomial time from Assumption 3.1, we can find $(\theta^{(1)}, \omega^{(1)})$ by continuously applying (3.8) such that

$$\tilde{\mathcal{H}}(y, \theta^{(1)}, \omega^{(1)}) \leq 0, \quad \forall y \in Q.$$

⁴Note that V^u is the cost-to-go associated with the control u while V_{θ} is the NN parameterized by the weights θ .

⁵Since $V_{\theta}(y) \in C^2(\bar{Q})$, $\tilde{\mathcal{H}}(y, \theta, \omega)$ is well a continuous function defined on $\bar{Q}$. From the compactness of $\bar{Q}$, we can always use Lemma 3.2 to approximate $\tilde{\mathcal{H}}(y, \theta, \omega)$ for all $y \in \bar{Q}$ and hence for all $y \in Q$.

Then, Algorithm 3 enters Phase 2 and again from (8.4), Lemma 3.2 and Assumption 3.3, there exists $\theta^\dagger$ such that $\tilde{\mathcal{H}}(y, \theta^\dagger, \omega^{(1)}) = 0$ for all $y \in Q$, which implies

$$\mathcal{L}_1(\theta^\dagger, \omega^{(1)}) = \mathbb{E}_{y \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(y, \theta^\dagger, \omega^{(1)})^2 \right] = 0 = \min_{\theta} \mathbb{E}_{y \sim \mu_1(Q)} \left[\tilde{\mathcal{H}}(y, \theta, \omega^{(1)})^2 \right] = \min_{\theta} \mathcal{L}_1(\theta, \omega^{(1)}).$$

We can then always minimize $\mathcal{L}_1(\theta, \omega^{(1)})$ over θ by continuously applying (3.9) and from Assumption 3.1, the global minimum $\mathcal{L}_1(\theta^\dagger, \omega^{(1)})$ of $\mathcal{L}_1(\theta, \omega^{(1)})$ can be attained in polynomial time. This means that we can always find $\theta^{(2)}$ by continuously applying (3.9) such that

$$\tilde{\mathcal{H}}(y, \theta^{(1)}, \omega^{(1)}) \leq \tilde{\mathcal{H}}(y, \theta^{(2)}, \omega^{(1)}) \leq 0, \quad \forall y \in Q.$$

Similarly, from Lemma 3.2 and Assumption 3.3, there always exists $\omega^\dagger$ such that

$$\tilde{\mathcal{H}}(y, \theta^{(2)}, \omega^\dagger) = \min_{\omega} \tilde{\mathcal{H}}(y, \theta^{(2)}, \omega), \quad \forall y \in Q.$$

Therefore, since

$$\min_{\omega} \mathcal{L}_2(\theta^{(2)}, \omega) = \min_{\omega} \mathbb{E}_{y \sim \mu_1(Q)} [\tilde{\mathcal{H}}(y, \theta^{(2)}, \omega)] \geq \mathbb{E}_{y \sim \mu_1(Q)} [\tilde{\mathcal{H}}(y, \theta^{(2)}, \omega^\dagger)] = \mathcal{L}_2(\theta^{(2)}, \omega^\dagger),$$

and since by definition, $\min_{\omega} \mathcal{L}_2(\theta^{(2)}, \omega) \leq \mathcal{L}_2(\theta^{(2)}, \omega)$, $\forall \omega \in \mathbb{R}^p$, we have

$$\mathcal{L}_2(\theta^{(2)}, \omega^\dagger) = \min_{\omega} \mathcal{L}_2(\theta^{(2)}, \omega).$$

Then, since the global minimum of $\mathcal{L}_2(\theta^{(2)}, \omega)$ can be obtained in polynomial time, the modified Hamiltonian $\tilde{\mathcal{H}}(y, \theta^{(2)}, \omega)$ can be taken to a global minimum for all $y \in Q$ by minimizing over ω . Therefore, we can also find $\omega^{(2)}$ by applying successively (3.10) such that

$$\tilde{\mathcal{H}}(y, \theta^{(2)}, \omega^{(2)}) \leq \tilde{\mathcal{H}}(y, \theta^{(2)}, \omega^{(1)}), \quad \forall y \in Q. \quad (8.6)$$

Applying this repeatedly for subsequent steps, we have for all $k \geq 1$:

$$\tilde{\mathcal{H}}(y, \theta^{(k)}, \omega^{(k)}) \leq \tilde{\mathcal{H}}(y, \theta^{(k+1)}, \omega^{(k)}) \leq 0, \quad \forall y \in Q, \quad (8.7)$$

and

$$\tilde{\mathcal{H}}(y, \theta^{(k+1)}, \omega^{(k+1)}) \leq \tilde{\mathcal{H}}(y, \theta^{(k+1)}, \omega^{(k)}), \quad \forall y \in Q. \quad (8.8)$$

Third part. We finally show that the value NN $V_{\theta^{(k)}}$ and the optimal control NN $u_{\omega^{(k)}}$ both converge uniformly to the value function V and optimal control u^* as $k \rightarrow \infty$ when using Algorithm 3 with the above early termination conditions. Since $V_{\theta}(y)$ is chosen $C^2(\bar{Q})$ for all $\theta \in \mathbb{R}^p$ (twice differentiable activation functions), we get by Dynkin's formula for all Markov controls $u := u(y)$,

$$\mathbb{E}_y^{\mathbb{P}} [V_{\theta^{(k)}}(Y_T^u)] = V_{\theta^{(k)}}(y) + \mathbb{E}_y^{\mathbb{P}} \left[\int_t^T (\mathcal{L}^u V_{\theta^{(k)}})(Y_r^u) dr \right],$$

which then leads to

$$V_{\theta^{(k)}}(y) = \mathbb{E}_y^{\mathbb{P}} \left[\phi(Y_T^u) - \int_t^T (\mathcal{L}^u V_{\theta^{(k)}})(Y_r^u) dr \right],$$

from Assumption 3.3, and similarly to

$$V_{\theta^{(k+1)}}(y) = \mathbb{E}_y^{\mathbb{P}} \left[\phi(Y_T^u) - \int_t^T (\mathcal{L}^u V_{\theta^{(k+1)}})(Y_r^u) dr \right].$$

Both of these equations are of course valid in particular for the Markov control $u(y) = u_{\omega^{(k)}}(y)$ since it is admissible and therefore,

$$\begin{aligned} V_{\theta^{(k+1)}}(y) - V_{\theta^{(k)}}(y) &= \mathbb{E}_y^{\mathbb{P}} \left[\int_t^T \left((\mathcal{L}^{u_{\omega^{(k)}}} V_{\theta^{(k)}})(Y_r^{u_{\omega^{(k)}}}) - (\mathcal{L}^{u_{\omega^{(k)}}} V_{\theta^{(k+1)}})(Y_r^{u_{\omega^{(k)}}}) \right) dr \right] \\ &= \mathbb{E}_y^{\mathbb{P}} \left[\int_t^T \left(\tilde{\mathcal{H}}(Y_r^{u_{\omega^{(k)}}}, \theta^{(k)}, \omega^{(k)}) - \ell(Y_r^{u_{\omega^{(k)}}}, u_{\omega^{(k)}}(Y_r^{u_{\omega^{(k)}}})) \right. \right. \\ &\quad \left. \left. - \tilde{\mathcal{H}}(Y_r^{u_{\omega^{(k)}}}, \theta^{(k+1)}, \omega^{(k)}) + \ell(Y_r^{u_{\omega^{(k)}}}, u_{\omega^{(k)}}(Y_r^{u_{\omega^{(k)}}})) \right) dr \right] \\ &= \mathbb{E}_y^{\mathbb{P}} \left[\int_t^T \left(\tilde{\mathcal{H}}(Y_r^{u_{\omega^{(k)}}}, \theta^{(k)}, \omega^{(k)}) - \tilde{\mathcal{H}}(Y_r^{u_{\omega^{(k)}}}, \theta^{(k+1)}, \omega^{(k)}) \right) dr \right]. \end{aligned} \quad (8.9)$$

However, from (8.7) and since we have $V_{\theta}(y) \geq 0$, we find for $k \geq 1$,

$$0 \leq V_{\theta^{(k+1)}}(y) \leq V_{\theta^{(k)}}(y), \quad \forall y \in \bar{Q}, \forall k \geq k^\dagger. \quad (8.10)$$

Therefore, $V_{\theta^{(k)}}(y)$ is pointwise convergent as k goes to ∞ and we can write $\lim_{k \rightarrow \infty} V_{\theta^{(k)}}(y) = V_{\theta^{(\infty)}}(y)$.

From the compactness of $\bar{Q}$, we obtain uniform convergence in y by Dini's Theorem. This implies that

$$\lim_{k \rightarrow \infty} \sup_{y \in \bar{Q}} |V_{\theta^{(k)}}(y) - V_{\theta^{(k+1)}}(y)| = 0.$$

Hence, from (8.9) and (8.7), this implies

$$\lim_{k \rightarrow \infty} \sup_{y \in \bar{Q}} \left| \tilde{\mathcal{H}}(y, \theta^{(k+1)}, \omega^{(k)}) - \tilde{\mathcal{H}}(y, \theta^{(k)}, \omega^{(k)}) \right| = 0. \quad (8.11)$$

Recall that we have $\min_{\theta} \mathcal{L}_1(\theta, \omega^{(k)}) = \min_{\theta} \mathbb{E}_{y \sim \mu_1(Q)} [\tilde{\mathcal{H}}(y, \theta, \omega^{(k)})^2] = 0$, for all $k \geq 1$. Therefore, according to (8.11) and the limit of $V_{\theta^{(\infty)}}(y)$, we must have

$$\lim_{k \rightarrow \infty} \tilde{\mathcal{H}}(y, \theta^{(k)}, \omega^{(k)}) = \lim_{k \rightarrow \infty} \tilde{\mathcal{H}}(y, \theta^{(k+1)}, \omega^{(k)}) = 0, \quad \forall y \in Q. \quad (8.12)$$

Therefore, $V_{\theta^{(\infty)}}(y)$ is solution of the Lyapunov equation (8.4) for a given policy $u_{\omega^{(\infty)}}(y)$ and hence,

$$V_{\theta^{(\infty)}}(y) = V^{u_{\omega^{(\infty)}}}(y), \quad \forall y \in Q. \quad (8.13)$$

where $V^{u_{\omega^{(\infty)}}}(y)$ is the cost-to-go function (8.1) associated with the Markov control $u_{\omega^{(\infty)}}(y)$. As shown in Abu-Khalaf and Lewis (2005), since $V_{\theta^{(k)}}(y)$ converges uniformly to $V_{\theta^{(\infty)}}(y) = V^{u_{\omega^{(\infty)}}}(y)$ on $\bar{Q}$ and from the uniqueness of the state trajectories (see the Lipschitz continuity conditions on the dynamic (2.2)) and the continuity of $u_{\omega^{(k)}}$, the sequence of controls $u_{\omega^{(k)}}$ converges uniformly to $u_{\omega^{(\infty)}}(y)$ on Q

when $k \rightarrow \infty$. Since $\omega^{(k+1)}$ is obtained by minimizing $\mathcal{L}_2(\theta^{(k+1)}, \omega)$ using (3.10), it is obvious from the dominated convergence theorem with (8.7) and from (8.12) that

$$\lim_{k \rightarrow \infty} \mathcal{L}_2(\theta^{(k+1)}, \omega^{(k+1)}) = \lim_{k \rightarrow \infty} \mathcal{L}_2(\theta^{(k+1)}, \omega^{(k)}) = 0, \quad (8.14)$$

which implies that

$$\lim_{k \rightarrow \infty} \min_{\omega} \tilde{\mathcal{H}}(y, \theta^{(k)}, \omega) = 0, \quad \forall y \in Q. \quad (8.15)$$

From (8.12), (8.15) and Theorem 2.2, one has $\lim_{k \rightarrow \infty} V_{\theta^{(k)}}(y) = V(y)$ for all $y \in \bar{Q}$ and $\lim_{k \rightarrow \infty} u_{\omega^{(k)}}(y) = u^*(y)$ for all $y \in Q$. Therefore, we conclude that $V_{\theta^{(k)}}$ goes to the value function V uniformly on $\bar{Q}$ and $u_{\omega^{(k)}}$ to the optimal control u^* uniformly on Q as $k \rightarrow \infty$, which means the global optimal solution is achieved. $\square$

Data availability statement The datasets and code generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Disclosure statement The authors declare that they have no conflict of interest.

References

- Abu-Khalaf, M. and Lewis, F. L. (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica*, 41(5):779–791.
- Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR.
- Almgren, R. (2003). Optimal execution with nonlinear impact functions and trading-enhanced risk. *Applied Mathematical Finance*, 10(1):1–18.
- Almgren, R. and Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, 3(2):5–40.
- Almgren, R., Thum, C., Hauptmann, E., and Li, H. (2005). Direct estimation of equity market impact. *Risk*, 18(7):58–62.
- Bachouch, A., Huré, C., Langrené, N., and Pham, H. (2022). Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodology and Computing in Applied Probability*, 24(1):143–178.
- Bai, Y., Chaolu, T., and Bilige, S. (2022). The application of improved physics-informed neural network (IPINN) method in finance. *Nonlinear Dynamics*, 107(4):3655–3667.
- Beck, C., Becker, S., Cheridito, P., Jentzen, A., and Neufeld, A. (2021). Deep splitting method for parabolic PDEs. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154.
- Beck, C., Becker, S., Grohs, P., Jaafari, N., and Jentzen, A. (2018). Solving stochastic differential equations and Kolmogorov equations by means of deep learning. *arXiv preprint arXiv:1806.00421*, 1(1).
- Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR.
- Berg, J. and Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41.
- Berner, J., Dablander, M., and Grohs, P. (2020). Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33:16615–16627.
- Bertsekas, D. (2023). *A Course in Reinforcement Learning*. Athena Scientific.
- Blechschmidt, J. and Ernst, O. G. (2021). Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006.
- Cartea, Á., Gan, L., and Jaimungal, S. (2019). Trading co-integrated assets with price impact. *Mathematical Finance*, 29(2):542–567.

- Cartea, Á. and Jaimungal, S. (2016). Incorporating order-flow into optimal execution. *Mathematics and Financial Economics*, 10:339–364.
- Cartea, Á., Jaimungal, S., and Penalva, J. (2015). *Algorithmic and High-Frequency Trading*. Cambridge University Press.
- Cartea, Á., Jaimungal, S., and Ricci, J. (2014). Buy low, sell high: A high frequency trading perspective. *SIAM Journal on Financial Mathematics*, 5(1):415–444.
- Domingo-Enrich, C., Drozdal, M., Karrer, B., and Chen, R. T. (2024a). Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*.
- Domingo-Enrich, C., Han, J., Amos, B., Bruna, J., and Chen, R. T. (2024b). Stochastic optimal control matching. *Advances in Neural Information Processing Systems*, 37:112459–112504.
- Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.
- Duan, J., Li, J., Ge, Q., Li, S. E., Bujarbaruah, M., Ma, F., and Zhang, D. (2023). Relaxed Actor-Critic with convergence guarantees for continuous-time optimal control of nonlinear systems. *IEEE Transactions on Intelligent Vehicles*, 8(5):3299–3311.
- Duan, Y., Chen, X., Houthoofd, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR.
- Fecamp, S., Mikael, J., and Warin, X. (2020). Deep learning for discrete-time hedging in incomplete markets. *Journal of computational Finance*, 25(2).
- Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR.
- Gatheral, J. (2010). No-dynamic-arbitrage and market impact. *Quantitative Finance*, 10(7):749–759.
- Greif, C. (2017). Numerical methods for Hamilton-Jacobi-Bellman equations.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.
- Hainaut, D. (2023). Valuation of guaranteed minimum accumulation benefits (GMAB) with physics inspired neural networks. *LIDAM Discussion Paper ISBA*, 2023(29).
- Han, J. and E, W. (2016). Deep learning approximation for stochastic control problems. *arXiv preprint arXiv:1611.07422*.
- Han, J. and Hu, R. (2021). Recurrent neural networks for stochastic control problems with delay. *Mathematics of Control, Signals, and Systems*, 33(4):775–795.
- Han, J., Jentzen, A., and E, W. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380.
- Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- Howard, R. A. (1960). *Dynamic programming and Markov processes*. John Wiley.
- Hu, R. and Lauriere, M. (2023). Recent developments in machine learning methods for stochastic control and games. *arXiv preprint arXiv:2303.10257*.
- Huré, C., Pham, H., Bachouch, A., and Langrené, N. (2021). Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557.
- Hutzenthaler, M., Jentzen, A., Kruse, T., and Nguyen, T. A. (2020). A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *SN partial differential equations and applications*, 1:1–34.
- Jacka, S. D. and Mijatović, A. (2017). On the policy improvement algorithm in continuous time. *Stochastics*, 89(1):348–359.
- Jentzen, A., Kuckuck, B., Neufeld, A., and von Wurstemberger, P. (2021). Strong error analysis for stochastic gradient descent optimization algorithms. *IMA Journal of Numerical Analysis*, 41(1):455–492.
- Ji, S., Peng, S., Peng, Y., and Zhang, X. (2022). Solving stochastic optimal control problem via stochastic maximum principle with deep learning method. *Journal of Scientific Computing*, 93(1):30.

- Li, X., Verma, D., and Ruthotto, L. (2024). A neural network approach for stochastic optimal control. *SIAM Journal on Scientific Computing*, 46(5):C535–C556.
- Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.
- Li, Y. and Forsyth, P. A. (2019). A data-driven neural network approach to optimal asset allocation for target based defined contribution pension plans. *Insurance: Mathematics and Economics*, 86:189–204.
- Li, Y., Guo, J., Lai, K. K., and Shi, J. (2022). Optimal portfolio liquidation with cross-price impacts on trading. *Operational Research*, 22(2):1083–1102.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Alex, G., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Mnih, V., Puigdomènech Badia, A., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*.
- Mowlavi, S. and Nabi, S. (2023). Optimal control of PDEs using physics-informed neural networks. *Journal of Computational Physics*, 473:111731.
- Muhle-Karbe, J., Wang, Z., and Webster, K. (2024). Stochastic liquidity as a proxy for nonlinear price impact. *Operations Research*, 72(2):444–458.
- Nakamura-Zimmerer, T., Gong, Q., and Kang, W. (2021). Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247.
- Nüsken, N. and Richter, L. (2021). Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2(4):48.
- Oksendal, B. (2013). *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media.
- Peyrl, H., Herzog, F., and Geering, H. P. (2005). Numerical solution of the Hamilton–Jacobi–Bellman equation for stochastic optimal control problems. In *Proc. 2005 WSEAS International Conference on Dynamical Systems and Control*, pages 489–497.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). 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.
- Schmidhuber, J. and Hochreiter, S. (1997). Long short-term memory. *Neural Computing*, 9(8):1735–1780.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Sirignano, J. and Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364.
- Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Touzi, N. (2012). *Optimal stochastic control, stochastic target problems, and backward SDE*, volume 29. Springer Science & Business Media.
- van Staden, P. M., Forsyth, P. A., and Li, Y. (2023). A parsimonious neural network approach to solve portfolio optimization problems without using dynamic programming. *arXiv preprint arXiv:2303.08968*.
- Wang, S., Yu, X., and Perdikaris, P. (2022). When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768.
- Warin, X. (2023). Reservoir optimization and machine learning methods. *EURO Journal on Computational Optimization*, 11:100068.
- Werbos, P. (1992). Approximate dynamic programming for real-time control and neural modeling. *Handbook of intelligent control*.
- Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671.
- Zhou, M., Han, J., and Lu, J. (2021). Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066.