

A Zeroth-Order Deep Learning Method for Fully Nonlinear Parabolic Partial Differential Equations with Unknown Coefficients

Yanwei Jia

YANWEIJIA@CUHK.EDU.HK

*Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong SAR*

Du Ouyang

DUOUYANG99@OUTLOOK.COM

*Department of Mathematical Sciences
Tsinghua University
Beijing, Beijing 100084, China*

Huyên Pham

HUYEN.PHAM@POLYTECHNIQUE.EDU

*CMAP
Ecole Polytechnique
Route de Saclay, Palaiseau 91128, France*

Xun Yu Zhou

XZ2574@COLUMBIA.EDU

*Department of Industrial Engineering and Operations Research &
The Data Science Institute
Columbia University
New York, NY 10027, USA*

Abstract

High-dimensional partial differential equations (PDEs) with unknown coefficients arise widely in scientific machine learning, including continuous-time reinforcement learning, yet solving them efficiently in a data-driven way remains challenging. Existing deep learning solvers often rely on repeated automatic differentiation to evaluate differential operators, which can cause instability and amplify derivative errors in high dimensions, while probabilistic methods based on stochastic representations require explicit knowledge of the data-generating dynamics and therefore do not apply to black-box environments. We introduce two types of simulators as data-generating mechanisms, and take a “representing-then-learning” approach that learns the solutions and their derivatives under settings where the underlying PDE operators are accessible only through simulations and pointwise evaluations. Our representation of derivatives relies on the zeroth-order derivative (ZOD) estimators derived from perturbed Monte Carlo trajectories. This fully model-free approach generates targets for the gradient and Hessian networks using only function evaluations. We provide a statistical learning analysis of the proposed approach, including a bias–variance tradeoff for ZODs. Assuming a standard contraction property of the underlying operator, we establish a non-asymptotic error bound that decomposes the total error into discretization error, approximation error, statistical error, and ZOD bias. Crucially, we derive the sample complexity of the learned representations in (weighted) Sobolev space, characterizing the error up to second-order derivatives. Numerical experiments illustrate the competitive performance of the method in moderate and high dimensions.

Keywords: Nonlinear PDEs, zeroth-order derivative estimators, model-free methods, deep learning, sample complexity in Sobolev space

1 Introduction

High-dimensional nonlinear partial differential equations (PDEs) are ubiquitous and fundamental in various fields of science and engineering, ranging from physics and fluid dynamics to mathematical finance and stochastic control. Recent progress in scientific machine learning employs deep neural networks (DNNs) for high-dimensional function approximation to solve PDEs, aiming to overcome the curse of dimensionality where the traditional numerical methods fall apart. While various deep learning methods have been proposed and have achieved promising numerical results, there are still limitations in their stability, scope of applications, and availability of theoretical foundations.

Broadly speaking, existing methods fall into two main categories. The first category consists of the residual-based methods, such as the deep Galerkin method (cf. Sirignano and Spiliopoulos 2018), and physics-informed neural networks (PINNs; cf. Raissi et al. 2019). These approaches approximate solutions by minimizing the PDE residuals over sampled points in space–time, and are applicable to a wide class of PDEs. However, they rely on repeated evaluation of differential operators through automatic differentiation and may suffer from optimization difficulties and stability issues, particularly in high dimensions or when second-order derivatives are involved. The second category consists of probabilistic methods that exploit stochastic representations of PDEs, including in particular methods based on backward stochastic differential equations (BSDEs) such as deep BSDEs (E et al., 2017) and deep backward dynamic programming (DBDP; cf. Huré et al. 2020). However, these methods are primarily tailored to the special type of nonlinear PDEs where BSDE representations are available based on Peng’s nonlinear Feynman–Kac formula (Peng, 1992); in particular, these PDEs are linear in the Hessians. For *fully* nonlinear PDEs, which are nonlinear in the unknown functions up to the second-order derivatives, such representations generally do not exist, limiting their applicability.¹

Notably, most of the existing literature on numerically solving PDEs, including the aforementioned papers, are “model-based”, namely, the studied PDEs are completely specified with their coefficient functions assumed to be known and given (hence so are the derivatives of those coefficients). However, PDEs that model *complex* systems can be impossible or impractical to specify, leading to equations without knowledge of the underlying operators and the derivatives of their coefficients. These PDEs operate like black-boxes for which we can only observe certain input–output data streams. Solving those black-box PDEs is motivated by ample real-life applications such as fluid turbulence, climate control, and the stabilization of plasma fusion reactors; see e.g. Li et al. (2020); Jiao et al. (2021); Lagergren et al. (2020) and the references therein for more motivating applications. It is also related to the recently developed continuous-time reinforcement learning (Wang et al., 2020; Jia and Zhou, 2022b,a, 2023), where the Hamilton–Jacobi–Bellman (HJB) equation becomes a black-box PDE when the parameters of the underlying control problem are unknown.

Another important aspect is that, besides obtaining the function approximation to the solution to a PDE, one often needs to find its derivatives for the downstream tasks, e.g., executing a control in a robotic system and hedging in finance, which concern the sensitivity of

1. We refer to Beck et al. (2023); Germain et al. (2023) for an overview, and to Zhang et al. (2026); Doumèche et al. (2025); Gazoulis et al. (2025); Bonito et al. (2026); Wang et al. (2022) for more recent progress on these two types of methods.

the solution with respect to the spatial variable. Existing methods generally treat derivatives implicitly. For example, PINNs and deep Galerkin follow a “learning-then-differentiating” approach: They train a neural network to minimize a value-based loss, which nonetheless does not theoretically guarantee that its derivatives will converge to the true derivatives. Consequently, when higher-order derivatives are computed via automatic differentiation on an imperfectly trained network, approximation errors compound, sometimes catastrophically. On the other hand, deep BSDE and DBDP require the equation coefficients to be known and given and embed gradients within model-specific stochastic formulations.

In this paper, we propose a new, data-driven approach to learning the solutions to a large class of *fully* nonlinear, black-box parabolic PDEs with strong theoretical guarantees in (weighted) Sobolev space up to second-order spatial derivatives. The PDEs consist of linear second-order parabolic operators and fully nonlinear source (nonhomogeneous) terms depending on the unknown solutions up to their second-order derivatives. Instead of recovering derivative information indirectly from a learned value approximation or from model-specific sensitivity representations, we adopt a representing-then-learning paradigm, where derivative quantities are first represented through stochastic estimators and then learned directly. This approach kills two birds with one stone: it deals with the unknown PDEs and learns the derivatives at the same time.

One of the main contributions of the paper is to define *precisely* what “data” and “data-driven” mean in our context of solving PDEs. Instead of the knowledge of the functional forms of PDE coefficients, we only require access to a “*simulator*” that, given an initial state, produces stochastic trajectories (i.e., “data”). These trajectories are diffusion processes, or equivalently solutions to the stochastic differential equations (SDEs) whose infinitesimal generator coincides with the aforementioned parabolic PDE operators. Moreover, the nonlinear source term and the function in the terminal condition are also black-box functions that can only output their function values when inputs are given. Our approach is applicable under two simulator settings: a “weak simulator” that takes an initial state as input and generates an independently and identically distributed (i.i.d.) random trajectory at each query, and a “strong simulator” that returns a *family* of trajectories indexed by a given set of initial states and driven by the *same* underlying randomness (Brownian motion in our setting), while remaining independent across queries. The notions of weak and strong simulators align with the classical definitions of weak and strong solutions to SDEs and capture the key distinction between them: a strong solution is defined with respect to a fixed Brownian motion, whereas in a weak solution the Brownian motion is itself a part of the solution (cf. Karatzas and Shreve 2014, Chapter 5, Definitions 2.1 and 3.1). Clearly, a strong simulator is automatically a weak one (but not *vice versa*) for which the “random seed” is also an input, so that querying the same initial state with the same seed produces the same trajectory. Importantly, the availability of the strong simulator covers model-based PDEs as a special case. It is also valid in applications such as finance where the “random seed” represents the realized asset price movements and inventory systems

where it represents exogenous demands. Note that in both cases, the simulator remains a black box, and we neither know nor attempt to learn or infer its internal structure.²

Our method combines: (i) approximate value iteration, to reduce a nonlinear problem to a sequence of linear problems, each effectively a Feynman–Kac-type equation (equivalent to the Bellman equation for policy evaluation in reinforcement learning); and (ii) zeroth-order derivative (ZOD) estimators, to approximate gradients and Hessians via function values at random perturbed points. At each iteration, we *jointly* learn the value, gradient, and Hessian through a composite loss function, ensuring consistency across all quantities. This loss function consists of two parts: the value loss and the derivative loss. The value loss is in the same spirit as the martingale loss function (Jia and Zhou, 2022b) for policy evaluation in continuous-time reinforcement learning, which captures the mean-squared error of the function value. The derivative loss captures the distance between the gradient/Hessian networks and their stochastic ZOD targets. Crucially, ZOD estimators require only function evaluations at perturbed trajectories and do not rely on model derivatives or automatic differentiation.

The main contributions of this paper are as follows:

- We introduce the notions of weak and strong simulators, and develop a zeroth-order deep learning method to solve a wide class of fully nonlinear black-box PDEs. At each iteration, we query the available simulator with randomly generated initial states based on a specially designed distribution and collect corresponding trajectories. In addition, we only rely on the point-wise evaluation of the nonlinear source term in a PDE, which is a general function of the state, function value, gradient, and Hessian. Thus, our method leads to a *model-free* algorithm applicable in black-box environments, in particular without attempting to estimate the PDE coefficients.
- To ensure convergence in (weighted) Sobolev space, we introduce a novel use of ZOD estimators as targets for approximating gradients and Hessians in the context of solving PDEs. In general, ZOD estimators are biased and noisy, and we characterize their unique bias–variance tradeoff through their bandwidth. In particular, the conventional variance reduction techniques for ZOD in optimization (e.g., the multi-point scheme) are not sufficient in our setting, and we show that the variance of ZOD can only be bounded with the strong simulator but not the weak one. Furthermore, using ZODs, we essentially first derive a (biased) stochastic representation for the gradient and Hessian and then fit two neural networks for them respectively. This representing-then-learning paradigm is therefore fundamentally different from approaches that recover derivatives indirectly from value approximations or through model-specific sensitivity representations. Notably, no automatic differentiation or the knowledge of the operator is required to derive ZOD.
- Most importantly, we establish convergence and sample complexity of the proposed learning method in the (weighted) Sobolev space up to second-order derivatives. To

2. There is a line of research called operator learning that aims to learn PDE solution operators by input–output data and DNNs (cf. Lu et al. 2021a), which is ultimately a model-based approach. By contrast, our approach is “model-free” (up to the form of the PDEs we study) in that we learn the solutions directly without learning the PDEs themselves.

our best knowledge, these are the first theoretical results on model-free learning for PDEs with an error analysis in second-order Sobolev space for fully nonlinear PDEs. Our theoretical analysis is built on several standard assumptions on PDEs and statistical learning theory for DNNs. On the one hand, we require a contraction property of the solution operator to an associated linear PDE. This property is typical to ensure the existence and uniqueness of the solution to the original fully nonlinear PDE, and is often necessary for most value iteration algorithms. On the other hand, we exclude the optimization error from our analysis and assume that the class of neural networks is sufficiently smooth and regular. Under these assumptions, we obtain an error bound characterizing the trade-offs among approximation error, sampling error, and ZOD bias. The overall error consists of an exponentially decaying initialization error (in terms of the number of approximate value iterations) and a geometrically weighted accumulation of one-step learning errors (Theorem 6). When the d -dimensional state process is observed on a uniform time grid with N grid points and the sample size is B , our results show that, besides the neural-network optimization and approximation errors, choosing the ZOD bandwidth optimally yields a one-step high-probability error bound of order $O((N^{-1} + \mathcal{P}(d)B^{-1/2})^{1/4})$ under the weak simulator with one-point ZOD estimators, and of order $O((N^{-1} + \mathcal{P}(d)B^{-1/2})^{1/2})$ under the strong simulator with multi-point ZOD estimators, up to logarithmic factors, where $\mathcal{P}(d)$ is a constant that depends polynomially on the dimension d (Theorem 9). We also experimentally demonstrate accurate approximation of both solutions and derivatives in moderate- and high-dimensional problems.

Related Literature

As a key ingredient of our method, ZOD estimators have been widely used in (black-box) stochastic optimization and reinforcement learning, where gradient information is unavailable; see, e.g., Nesterov and Spokoiny (2017); Salimans et al. (2017) among many others, but not yet in the context of PDE solvers to our best knowledge.³ Due to the unique context of solving parabolic PDEs, we rely on the celebrated Feynman–Kac formula to obtain a stochastic representation of the function value. Moreover, our ZOD estimator differs from the traditional one in optimization in two aspects. First, at every iteration, we query the simulator to generate a stochastic trajectory with the given initial state, which is then used to compute a functional value using the current function value, gradient and Hessian. Second, the ZOD estimator is not used for optimization directly; instead, it is used as a target to train other networks. Therefore, the bias–variance tradeoff of ZOD and its impact on the entire iterative procedure are both new problems in our context.

Another key ingredient of our method is the approximate value iteration, which is a classical approach adopted in reinforcement learning to solve the Bellman equations or HJB equations in continuous-time optimal controls, a well-known class of nonlinear equations; see, e.g., Bertsekas and Tsitsiklis (1996); Sutton and Barto (1998) for discrete-time Bellman equations and Yong and Zhou (1999); Fleming and Soner (2006); Pham (2009) for

3. ZOD has been used to replace auto-differentiation. For instance, He et al. (2023); Shi et al. (2024); Hu et al. (2025) use ZOD to approximate the differential operators of a neural network in PINN. The role of ZOD is fundamentally different in our approach.

continuous-time HJB equations. The core idea for the approximate value iteration is to reduce a fully nonlinear equation into successive linear equations. At every step, it is sufficient to solve the linear equation approximately, and the total error would be accumulated and controlled under suitable regularity conditions. We borrow this idea and solve a linear PDE at each iteration. Moreover, this linear PDE can be interpreted as a Feynman–Kac (Bellman) equation for policy evaluation in reinforcement learning and solvable in a model-free way (Jia and Zhou, 2022b). However, Jia and Zhou (2022b) only show the consistency of the value function, not its derivatives. In the current PDE setting, learning derivatives is equally crucial for the iteration to converge to the true solution because the nonlinear source term of the PDE depends explicitly on the gradient and Hessian.

In supervised regression, recovering derivatives by automatic differentiation of a network trained only on function values is known to be statistically problematic, despite the availability of the well-known universal approximation theorem of neural networks that can be employed to approximate derivatives (Hornik et al., 1990; Siegel, 2023). Using the typical mean-squared error as the loss function, the theoretical guarantee is often on the function value (e.g., Barron 1994). To discipline the behavior of the learned derivatives, a commonly used trick is to regularize and penalize large derivatives of the neural network; see, e.g., Kim and Gu (2004); Fischer and Steinwart (2020) among many others. One can then establish statistical errors in the Sobolev space. By contrast, we go down a different path due to our access to a simulator and aim to solve a PDE rather than simply fitting a function to a given, exogenous data set. Indeed, we can simulate stochastic trajectories *strategically* to *create* training data. In other words, in our approach, data sets are also endogenously designed to learn a function in an “optimal” way in the Sobolev space.

Therefore, our approach, representing-then-learning, should be regarded as a (black-box/model-free) simulation-based learning method, and the first “representing” stage is essentially a sampling stage. From this perspective, our approach is comparable to the sensitivity analysis associated with mathematical expectations in the simulation literature, such as the likelihood ratio method (cf. Glasserman 2004) and the Malliavin calculus approach (cf. Ma and Zhang 2002). In a similar context of deep learning for solving PDEs, Han et al. (2026) propose the method of deep Picard iteration (DPI), where neural networks are trained using Monte Carlo estimates of the solution and its derivatives. Picard iteration coincides with the approximate value iteration in our context. The main difference between DPI (and its extensions) and our approach is that DPI employs the exact stochastic representations for derivatives based on Malliavin calculus (Bismut–Elworthy–Li formulas; cf. Baños 2018), which is unbiased but requires the knowledge of the coefficients of the differential operator and their derivatives, and thus operates in a model-based setting. Another line of work is the differential learning framework as in Lefebvre et al. (2023), which proposes to jointly learn the solution and its derivatives for fully nonlinear PDEs with convex Hamiltonians. Their approach leverages stochastic control representations and incorporates derivative information through Malliavin-type sensitivities. By contrast, our approach is applicable in a model-free context at the cost of inevitably incurring more bias. More importantly, we have theoretically justified the convergence of our resulting algorithm, showing that the bias introduced by ZOD can be controlled.

This paper also contributes broadly to the literature on solving PDEs by machine learning. In addition to the above-mentioned methods such as the deep Galerkin method (Sirig-

nano and Spiliopoulos, 2018), PINNs (Raissi et al., 2019), deep BSDEs (E et al., 2017), DBDP (Huré et al., 2020), and deep Picard iteration (Han et al., 2026), there are also specifically designed methods for special classes of PDEs, such as deep Ritz methods for variational problems (E and Yu, 2018; Lu et al., 2021b), actor–critic methods for HJB equations (Zhou et al., 2021; Jia and Zhou, 2022a; Zhu et al., 2025), and the high-order scheme for policy evaluation (Mou and Zhu, 2025). However, few of them analyze statistical errors and sample complexity in the presence of neural network-based function approximation. Mou (2025) analyzes errors in the Sobolev space of a least-squares temporal-difference algorithm arising from continuous-time policy evaluation. Lu et al. (2022) analyze two particular algorithms to solve linear PDEs. Little is known about fully nonlinear PDEs, especially in a black-box setting. The present paper not only develops a new deep learning method that is applicable to a general class of nonlinear unknown PDEs, but also theoretically establishes the sample complexity.

The remainder of the paper is organized as follows. In Section 2, we introduce the problem formulation, put forward the two simulators, and present the approximate value iteration framework for fully nonlinear PDEs. Section 3 develops the probabilistic representation of derivatives based on zeroth-order estimators and introduces the representing-then-learning paradigm. In Section 4, we develop the proposed deep learning algorithm, including the loss construction and implementation details. Section 5 provides the theoretical analysis, establishing convergence of the method and quantifying the impact of discretization, approximation, sampling, and perturbation errors. Section 6 presents numerical experiments that validate the performance of the proposed approach in both low- and high-dimensional settings. Section 7 concludes. Supplementary materials and proofs of statements are placed in the Appendix.

2 Problem Description and Preliminaries

In this paper, we study how to effectively solve a class of nonlinear parabolic PDEs:

$$\begin{cases} \partial_t v + \mathcal{L}v + f(x, v, \nabla v, \nabla^2 v) = 0, & (t, x) \in [0, T) \times \mathbb{R}^d, \\ v(T, x) = g(x), & x \in \mathbb{R}^d, \end{cases} \quad (2.1)$$

where $\nabla v \in \mathbb{R}^d$ and $\nabla^2 v \in \mathbb{R}^{d \times d}$ are respectively the gradient and Hessian of v with respect to the spatial variables, and the second order differential operator \mathcal{L} is

$$\mathcal{L}\varphi := \sum_{i,j=1}^d a_{ij}(x)\partial_{ij}\varphi + \sum_{i=1}^d b_i(x)\partial_i\varphi,$$

with ∂_t, ∂_i being respectively the partial derivative in the time variable t and the i -th entry of the spatial variable x_i , and ∂_{ij} the corresponding second-order derivative. Let $a(x) \equiv (a_{ij}(x)) \in \mathbb{S}_+^d$, assumed to be a positive semidefinite matrix.

The operator \mathcal{L} can therefore be viewed as the infinitesimal generator of a time-homogeneous diffusion process driven by a d -dimensional Brownian motion W , satisfying the following stochastic differential equation (SDE):

$$dX_s = b(X_s)ds + \sqrt{2}\sigma(X_s)dW_s, \quad (2.2)$$

where $\sigma(x)\sigma(x)^\top = a(x)$.⁴ We further denote by $X^{t,x}$ the solution to the SDE with initial condition $X_t = x$.

We first reiterate the following important consideration and goal in our study:

- (1) The dimension of the spatial variable d can be potentially high, and we want to avoid the curse of dimensionality arising from the conventional grid-based numerical methods (such as finite-difference and finite-element methods).
- (2) The PDE can be potentially described in a black-box manner and we aim to develop model-free, data-driven methods. A black-box environment means (i) the functions g, f are black-box functions that can only output their values if inputs are queried; and (ii) the operator \mathcal{L} is not known and only its associated processes/trajectories $\{X_s^{t,x}\}_{t \leq s \leq T}$ can be observed upon querying a *simulator*. Thus, “data” here means the trajectories that are returned upon a query.

We now formalize the “simulators” considered in this paper. Note that any simulator defined below remains unknown to us, being a device outputting data upon a query, and we do not attempt to learn its underlying structure. We consider two settings.

Condition 1 *A weak simulator returns a trajectory $\{X_s^{t,x}\}_{t \leq s \leq T}$ for each query with input (t, x) , such that*

- (i) $\{X_s^{t,x}\}_{t \leq s \leq T}$ is the weak solution to (2.2);
- (ii) the random sources (Brownian motions) in any two different queries are independent of each other.

Condition 2 *A strong simulator returns a collection of trajectories $\{\{X_s^{t,x_k}\}_{t \leq s \leq T}, k = 1, \dots, K\}$ for each query whose input is a finite set of initial states $\{(t, x_1), \dots, (t, x_K)\}$, such that*

- (i) upon each query, every trajectory of $\{X_s^{t,x_k}\}_{t \leq s \leq T}$ is the strong solution to (2.2) with respect to the same Brownian motion $\{W_s\}_{t \leq s \leq T}$ with the initial states (t, x_k) ;
- (ii) the random sources (Brownian motions) in any two different queries are independent of each other.

Evidently, a weak simulator is implied by a strong simulator, which is further implied by a PDE with known coefficients. Neither converse implication, however, is true. The notions of weak and strong simulators coincide with the classical definitions of weak and strong solutions to SDEs and capture the subtle distinction between them: a strong solution is defined with respect to a fixed Brownian motion, whereas in a weak solution the Brownian motion is a part of the solution. The weak simulator is the typical setting in which each query produces an i.i.d. sample path. By contrast, the strong simulator not only generates multiple trajectories per query, but also keeps the underlying randomness fixed within that query, much like fixing a “random seed”. Mathematically, querying a strong simulator

4. For ease of presentation, in this paper we consider only the case where \mathcal{L} and f do not depend on time, i.e. time-homogeneous problems. All the conclusions can be readily extended to the time inhomogeneous case.

with a set of initial states $\{(t, x_1), \dots, (t, x_K)\}$ means that K trajectories $\{X_s^{t, x_k}\}_{t \leq s \leq T}$ are generated satisfying

$$X_\tau^{t, x_k} = x_k + \int_t^\tau b(X_s^{t, x_k}) ds + \int_t^\tau \sqrt{2} \sigma(X_s^{t, x_k}) dW_s,$$

for all $\tau \in [t, T]$ and all $k = 1, \dots, K$, almost surely, where the same Brownian motion W drives all the K trajectories (but W is not the output of the simulator and hence is “hidden”). Upon another query, another hidden independent Brownian motion will be used to generate these trajectories.

These precise definitions of the simulators also give precise meanings to the terms “model-free” and “data-driven”, and it is possible to extend them to settings other than PDEs. Our approach and results in this paper involve both settings, but the theoretical guarantees differ, depending on which simulator is available. In short, access to a strong simulator leads to substantially sharper results than access only to a weak simulator because the former facilitates variance reduction; see Proposition 5 and Theorem 9.

Next, we discuss a few important building blocks of our approach and highlight the challenges of the existing approaches. As they are both well-known in the literature, we omit to specify needed regularity conditions to avoid unnecessary technicality in the discussion. Rigorous conditions and analysis will be presented in Section 5.

2.1 Value Iteration of the Nonlinear PDEs

We consider the value iteration for solving the nonlinear PDE (2.1): Define the solution mapping of a linear parabolic equation $\mathcal{S} : v \mapsto u$, such that u solves

$$\begin{cases} \partial_t u + \mathcal{L}u + f(x, v, \nabla v, \nabla^2 v) = 0, & (t, x) \in [0, T) \times \mathbb{R}^d, \\ u(T, x) = g(x), & x \in \mathbb{R}^d. \end{cases} \quad (2.3)$$

By assuming suitable regularity conditions on \mathcal{L} and f, g , \mathcal{S} is a well-defined mapping from a suitable space to itself. For any given initial value v_0 , we can therefore define the value iteration sequence by $v_1 = \mathcal{S}v_0, \dots, v_{n+1} = \mathcal{S}v_n, \dots$. Based on the Feynman–Kac formula (cf. Karatzas and Shreve 2014), the solution mapping \mathcal{S} admits a probabilistic representation:

$$\mathcal{S}v(t, x) = u(t, x) = \mathbb{E} \left[g(X_T^{t, x}) + \int_t^T f(X_s^{t, x}, v(s, X_s^{t, x}), \nabla v(s, X_s^{t, x}), \nabla^2 v(s, X_s^{t, x})) ds \right]. \quad (2.4)$$

Hence, correspondingly, the value iteration can be represented by

$$v_{n+1}(t, x) = \mathbb{E} \left[g(X_T^{t, x}) + \int_t^T f(X_s^{t, x}, v_n(s, X_s^{t, x}), \nabla v_n(s, X_s^{t, x}), \nabla^2 v_n(s, X_s^{t, x})) ds \right]. \quad (2.5)$$

We make three important observations. First, value iteration reduces the difficulty of solving a *nonlinear* PDE to solving *linear* PDEs successively and, thus, is commonly employed as the first attempt in many numerical procedures. It is also widely used to establish the well-posedness of nonlinear PDEs, and is referred to as Picard iteration in the

PDE literature (e.g., Han et al. 2026). In particular, if we assume that the solution mapping \mathcal{S} is a contraction under a suitable norm, then this iteration converges at an exponential rate. Such contraction properties are dictated by those of \mathcal{L} and f, g , and we will present a sufficient condition in Appendix A for the contraction property.

Second, besides value iteration, there are alternative ways to solve nonlinear PDEs by iteratively solving linearized PDEs, e.g., Newton’s method, which expands the nonlinear term f into its linear approximation. Our approach in this paper is a zeroth-order method: it uses only knowledge of f itself, rather than its first-order derivative (which is unknown in the black-box/model-free setting), but can be generalized to Newton’s method if such a form is known.

Third, the value iteration (2.5) provides a theoretical mechanism to solve the nonlinear PDE by solving a sequence of linear problems. However, numerical challenges remain, and numerically solving linear (black-box) PDEs in high dimensions is already difficult. The probabilistic representation of the solution mapping \mathcal{S} transforms the problem of solving PDEs into evaluating expectations over stochastic trajectories which can in turn be computed using Monte Carlo methods, thus avoiding using grid points in the conventional numerical PDE methods. Such a probabilistic representation seems to be the only way to remedy the curse of dimensionality in solving PDEs. Moreover, the probabilistic representation (2.4) only requires the function *values* of g, f along the trajectories of X_t and not the precise specification of \mathcal{L} nor the exact *functional forms* of g, f . Hence it is applicable to a black-box environment.⁵

2.2 Is Deep Neural Network a Savior For All?

To overcome the curse of dimensionality, deep neural networks (DNNs) have been widely adopted in recent algorithms based on probabilistic representations such as (2.4) to approximate solutions to PDEs. DNNs have been shown, both theoretically and empirically, to approximate high-dimensional functions efficiently without suffering from an exponential dependence on dimension; see e.g. Han et al. (2018), Beck et al. (2023), Germain et al. (2023). In our context, the value iteration requires successively solving linear PDEs and, hence, DNNs can be used to approximate each solution v_n . Such an approach has been explored by Jia and Zhou (2022b) and more broadly in the reinforcement learning literature for policy evaluation.

Does this, then, mean that DNNs provide a universal remedy for all these problems? The answer is negative. In value iteration, the solution obtained at the previous step is used as an input to define the next linear PDE. Therefore, it is crucial to ensure that the one-step error does not propagate and amplify over iterations. In particular, for a fully nonlinear PDE like (2.3), the next linear PDE in the value iteration (2.5) depends on the previous solution via its spatial *derivatives* up to the second order (i.e., f depends on ∇v_n and $\nabla^2 v_n$). Consequently, the one-step approximation error must be controlled not only in the current function iterate, but also in its first- and second-order derivatives, which imposes significantly more stringent requirements on the approximation scheme.

5. This problem also arises in the context of policy evaluation in reinforcement learning (see, e.g., Jia and Zhou 2022b).

To illustrate the issue, let V_n be the DNN approximation of v_n at step n . To compute the target for step $n + 1$, we must evaluate $f(\cdot, V_n, \nabla V_n, \nabla^2 V_n)$. A standard approach would be to train V_{n+1} to minimize the L^2 distance to the target value implied by (2.5). However, convergence in the L^2 norm of the value function does not guarantee convergence of its derivatives (i.e., $\|V_n - v^*\|_{L^2} \rightarrow 0$ does not imply $\|\nabla V_n - \nabla v^*\|_{L^2} \rightarrow 0$), particularly in high-dimensional spaces. This phenomenon is well-known in Sobolev training and non-parametric regression; see e.g. Czarnecki et al. (2017).

We emphasize that this error is not due to the numerical error in computing derivatives as in, say, the finite-difference method. When using DNNs, derivatives are computed via automatic differentiation (auto-diff) through *backpropagation*, which provides exact derivatives of the network representation.⁶ Therefore, if ∇V_n and $\nabla^2 V_n$ are computed directly via automatic differentiation of an imperfectly trained network V_n , the errors in derivatives may be significantly amplified, destabilizing the source term f for the next iteration. Consequently, learning the value function alone is insufficient, and one has to learn the gradient and Hessian, denoted by $G_n \approx \nabla v_n$ and $H_n \approx \nabla^2 v_n$, and update them *alongside* with V_n .

Existing approaches approximate derivatives using differential learning techniques in a model-based setting, often relying on the simulation of tangent processes associated with the underlying stochastic differential equation; see Glasserman (2004) and Huge and Savine (2020). These methods, however, typically require access to derivatives of the coefficients (e.g., the exact form of \mathcal{L}) and in addition may introduce additional variance and computational complexity.

The above discussions motivate us to develop alternative approaches that avoid explicit differentiation while maintaining stability over iterations.

3 Approximate Probabilistic Representation of Derivatives

In this section, we detail the proposed numerical scheme. Based on the discussions in Section 2.2, the core challenge lies in generating accurate and low-variance training targets for networks $(G, H) \approx (\nabla v, \nabla^2 v)$ that can be learned using function values of v . In other words, we will not rely on recovering derivatives from a learned value approximation, whether through automatic differentiation or model-dependent sensitivity formulas; rather, we need to derive a probabilistic model-free representation of the derivatives $(\nabla v, \nabla^2 v)$, i.e., forms similar to (2.4), and then train networks (G, H) to approximate them – a “representing-then-learning” paradigm.

To achieve this, we rely on the “*zeroth-order derivative*” (ZOD) estimators as the fundamental building block of our method. Since ZOD has not yet been utilized in the context of solving PDEs with DNNs, we first review its properties for the reader’s convenience.

3.1 Classical Zeroth-Order Derivative Estimators

The basic idea of ZOD is to use the function values at randomly perturbed points, $u(t, x + \epsilon Z)$, to approximate the derivative ∇u and even higher-order derivatives such as $\nabla^2 u$, where $\epsilon > 0$ is the bandwidth of the perturbation and Z follows a standard, sphere-symmetric dis-

6. The only exception occurs for non-differentiable functions, e.g. $v(x) = x^+$ at $x = 0$. In practice, this is rarely an issue since neural networks are smooth or almost everywhere differentiable.

tribution. In our case, without loss of generality, we take $Z \sim \mathcal{N}(0, I)$, a d -dimensional standard normal distribution. The following Lemmas 1 to 3 are standard results under slightly different regularity conditions; see, e.g., Nesterov and Spokoiny (2017); Balasubramanian and Ghadimi (2022). For completeness and for the reader's convenience, we provide their proofs in the appendix. These lemmas construct several well-known ZOD estimators and present their properties.

Lemma 1 *Given function $u(\cdot, \cdot)$, $(t, x) \in [0, T] \times \mathbb{R}^d$, and $Z \sim \mathcal{N}(0, I_d)$. If $u(t, \cdot)$ is three times continuously differentiable and its derivatives satisfy the polynomial growth condition, then $\frac{Z}{\epsilon}u(t, x + \epsilon Z)$ is a biased estimator for $\nabla u(t, x)$, and its bias and variance are upper bounded by*

$$\left| \mathbb{E} \left[\frac{Z}{\epsilon} u(t, x + \epsilon Z) \right] - \nabla u(t, x) \right| \leq C\epsilon^2, \quad \text{Var} \left[\frac{Z}{\epsilon} u(t, x + \epsilon Z) \right] \leq C \frac{1}{\epsilon^2}.$$

Furthermore, if $u(t, \cdot)$ is four times continuously differentiable and its derivatives satisfy the polynomial growth condition, then $\frac{ZZ^\top - I_d}{\epsilon^2}u(t, x + \epsilon Z)$ is a biased estimator for $\nabla^2 u(t, x)$, and its bias and variance are upper bounded by

$$\left| \mathbb{E} \left[\frac{ZZ^\top - I_d}{\epsilon^2} u(t, x + \epsilon Z) \right] - \nabla^2 u(t, x) \right| \leq C\epsilon^2, \quad \text{Var} \left[\frac{ZZ^\top - I_d}{\epsilon^2} u(t, x + \epsilon Z) \right] \leq C \frac{1}{\epsilon^4}$$

Variance Reduction of Multi-Point ZOD Estimators As shown in Lemma 1, the naïve one-point estimators suffer from a critical drawback: their variances diverge as the perturbation magnitude, ϵ , approaches zero. This means that to get an accurate derivative estimate, ϵ needs to be small to reduce the bias but the number of samples need to increase dramatically to reduce the variance. This issue can often be mitigated by the symmetric (or central) differencing technique, known as the multi-point zeroth-order estimators.

We begin by considering the estimation of the gradient, $\nabla u(t, x)$. A one-point estimator, as demonstrated in Lemma 1, has variances that scale as $O(\epsilon^{-2})$, whereas a two-point estimator exhibits *bounded* variances, as stipulated in Lemma 2:

Lemma 2 *Given function $u(\cdot, \cdot)$, $(t, x) \in [0, T] \times \mathbb{R}^d$, and $Z \sim \mathcal{N}(0, I_d)$. Assume $u(t, \cdot)$ is three times continuously differentiable and its derivatives satisfy the polynomial growth condition. Define*

$$\hat{g}(t, x; \epsilon) := \frac{u(t, x + \epsilon Z) - u(t, x - \epsilon Z)}{2\epsilon} Z. \tag{3.1}$$

Then

$$|\mathbb{E} [\hat{g}(t, x; \epsilon)] - \nabla u(t, x)| \leq C\epsilon^2, \quad \text{Var} [\hat{g}(t, x; \epsilon)] \leq C',$$

for some constants $C, C' > 0$ that are independent of ϵ .

The one-point estimator for Hessian has even larger variances that diverge at a rate of $O(\epsilon^{-4})$ and hence is much more inefficient. However, with symmetric differencing, we can construct a stable estimator with bounded variances.

Lemma 3 *Let the function $u : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}$ satisfy the conditions of Lemma 2, and further assume $u(t, \cdot)$ is four times continuously differentiable and its derivatives satisfy the polynomial growth condition. Define*

$$\hat{H}(t, x; \epsilon) := \frac{u(t, x + \epsilon Z) - 2u(t, x) + u(t, x - \epsilon Z)}{2\epsilon^2} (ZZ^\top - I_d). \quad (3.2)$$

Then

$$\left| \mathbb{E} \left[\hat{H}(t, x; \epsilon) \right] - \nabla^2 u(t, x) \right| \leq C\epsilon^2, \quad \text{Var} \left[\hat{H}(t, x; \epsilon) \right] \leq C',$$

for some constants $C, C' > 0$ that are independent of ϵ .

3.2 Illustrations of Representing-Then-Learning Paradigm

To illustrate our “representing-then-learning” paradigm, we first use a toy task of regression and contrast it with a conventional routine where the derivative is obtained from differentiating the learned value function. This simple regression task is indeed equivalent to solving a linear PDE, which will be demonstrated numerically in Subsection 6.1.

The regression task is described as follows: suppose $(X, Y) \in \mathbb{R}^2$ follows a joint distribution that can be written as $\pi(dx)\rho(x, dy)$, where $\pi(dx)$ is the marginal distribution of X and $\rho(x, dy)$ is the conditional distribution of $Y|X = x$. We can generate samples from both X and $Y|X = x$, but do not know the functions π and ρ . We are interested in numerically finding the function $f(x)$ and its derivatives $f'(x), f''(x)$ where $f(X) = \mathbb{E}[Y|X]$ is the conditional expectation.

Learning-then-Differentiating Conventionally, one can first generate samples $X_i \sim \pi(\cdot)$ and $Y_i \sim \rho(X_i, \cdot)$, and then fit a DNN regression by

$$V = \arg \min_{\varphi \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (Y_i - \varphi(X_i))^2,$$

where \mathcal{H} stands for the functional class of the chosen DNN family. The resulting function approximation V is well-known to satisfy performance guarantee (ignoring the optimization error). However, differentiating (equivalent to auto-diff) V to obtain V', V'' does not have any theoretical guarantee.⁷

Representing-then-Learning Our approach is based on obtaining a sample approximation for the derivatives directly via ZOD estimators. Let us illustrate the one-point ZOD for estimating derivatives first. Besides the samples (X_i, Y_i) , we further sample $Z_i \sim \mathcal{N}(0, 1)$ and sample $Y_i^{(+)} \sim \rho(X_i + \epsilon Z_i, \cdot)$. Now consider the newly constructed sample $\frac{Z_i}{\epsilon} Y_i^{(+)}$. It follows from Lemma 1 that

$$\mathbb{E} \left[\frac{Z_i}{\epsilon} Y_i^{(+)} \middle| X_i \right] = \mathbb{E} \left[\mathbb{E} \left[\frac{Z_i}{\epsilon} Y_i^{(+)} \middle| X_i, Z_i \right] \middle| X_i \right] = \mathbb{E} \left[\frac{Z_i}{\epsilon} f(X_i + \epsilon Z_i) \middle| X_i \right] \approx f'(X_i) + \text{bias}.$$

7. One possible mitigation to this is to consider the regularized regression by $\min_{\varphi \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (Y_i - \varphi(X_i))^2 + \lambda |\varphi''(X_i)|^2$, with a tuning parameter $\lambda > 0$ that induces smoother function approximation; see e.g. Wahba (1990).

Thus, $\frac{Z_1}{\epsilon} Y_1^{(+)}, \dots, \frac{Z_n}{\epsilon} Y_n^{(+)}$ are n independent, biased samples of $f'(X_1), \dots, f'(X_n)$ respectively with heterogeneous bias and standard errors. Then, we learn a function G by

$$G = \arg \min_{\varphi \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \left(\frac{Z_i}{\epsilon} Y_i^{(+)} - \varphi(X_i) \right)^2.$$

Similarly, we can construct samples, $\frac{Z_i^2-1}{\epsilon^2} Y_i^{(+)}, i = 1, \dots, n$, for the second-order derivative because

$$\begin{aligned} \mathbb{E} \left[\frac{Z_i^2-1}{\epsilon^2} Y_i^{(+)} \middle| X_i \right] &= \mathbb{E} \left[\mathbb{E} \left[\frac{Z_i^2-1}{\epsilon^2} Y_i^{(+)} \middle| X_i, Z_i \right] \middle| X_i \right] = \mathbb{E} \left[\frac{Z_i^2-1}{\epsilon^2} f(X_i + \epsilon Z_i) \middle| X_i \right] \\ &\approx f''(X_i) + \text{bias}. \end{aligned}$$

Hence, we can learn a function H by

$$H = \arg \min_{\varphi \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \left(\frac{Z_i^2-1}{\epsilon^2} Y_i^{(+)} - \varphi(X_i) \right)^2.$$

The learned DNNs G, H will then be reasonable approximations to f', f'' respectively, subject to the optimization error (controlled by the optimization solvers), approximation error (controlled by the functional class \mathcal{H}), sampling error (controlled by n and ϵ), and the bias (controlled by ϵ).

Furthermore, the multi-point ZOD estimators can be constructed as follows: We sample $Y_i^{(\pm)} \sim \rho(X_i \pm \epsilon Z_i, \cdot)$, and consider $\frac{Y_i^{(+)} - Y_i^{(-)}}{2\epsilon} Z_i$ and $\frac{Y_i^{(+)} - 2Y_i + Y_i^{(-)}}{2\epsilon^2} (Z_i^2 - 1)$ for the biased samples of $f'(X_i), f''(X_i)$ respectively. In the following, for ease of presentation we only describe the construction under the one-point ZOD, but multi-point ZOD can be similarly constructed.

3.3 Estimators for Solutions to Linear PDEs

We now apply the ZOD framework to our specific problem of solving nonlinear PDEs. Recall that in the value iteration, the target function at the $(n+1)$ -th step is given by the conditional expectation:

$$\hat{V}_{n+1}(t, x) = \mathbb{E} [R^{t,x}(\mathbf{U}_n)],$$

where $\mathbf{U}_n = (V_n, G_n, H_n)$ represents the networks from the previous iteration. The stochastic reward functional is defined as:

$$R^{t,x}(\mathbf{U}_n) := g(X_T^{t,x}) + \int_t^T f(X_s^{t,x}, V_n(s, X_s^{t,x}), G_n(s, X_s^{t,x}), H_n(s, X_s^{t,x})) ds.$$

Since \hat{V}_{n+1} is not available analytically, we cannot directly apply the ZOD estimators to the function \hat{V}_{n+1} itself. However, we have access to the random variable $R^{t,x}$, which is an *unbiased* estimator of $\hat{V}_{n+1}(t, x)$.

This motivates the *One-Point ZOD Estimator*. Instead of estimating \hat{V}_{n+1} and then employing ZOD, we directly use the perturbed realization of the reward $R^{t,x+\epsilon Z}$ to estimate the spatial derivatives. Taking the first-order derivative as an example, recall that

$\frac{Z}{\epsilon}\hat{V}_{n+1}(t, x + \epsilon Z)$ is a biased estimator for $\nabla\hat{V}_{n+1}(t, x)$, and $R^{t, x + \epsilon Z}(\mathbf{U}_n)$ is an unbiased estimator of $\hat{V}_{n+1}(t, x + \epsilon Z)$. Thus, we are motivated to take $\frac{Z}{\epsilon}R^{t, x + \epsilon Z}(\mathbf{U}_n)$ as a one-point estimator for $\nabla\hat{V}_{n+1}(t, x)$, without relying on an intermediary estimate of \hat{V}_{n+1} . In a similar fashion, a one-point estimator for the second-order derivative can also be constructed. The following proposition presents the error bounds of the one-point ZOD estimators in L_2 norm.

For integers $l, k \geq 0$, $C_b^{l, k}([0, T] \times \mathbb{R}^d)$ denotes the class of functions that are l times continuously differentiable in t and k times continuously differentiable in x , and whose derivatives are uniformly bounded, and $C_p^{l, k}([0, T] \times \mathbb{R}^d)$ denotes the corresponding class in which these derivatives have at most polynomial growth in x . Similarly, $C_b^k(\mathbb{R}^d)$ and $C_p^k(\mathbb{R}^d)$ denote the time-independent counterparts. These notations are understood componentwise for vector- or matrix-valued functions.

We need the following standing assumption.

Assumption 1 *The coefficients b and σ are sufficiently regular so that the SDE (2.2) admits a unique strong solution. Moreover, they satisfy the linear growth and bounded condition, i.e., there are constants L_b and σ_0 such that*

$$|b(t, x)| \leq L_b(1 + |x|), \quad |\sigma(t, x)| \leq \sigma_0, \quad \text{for all } (t, x) \in [0, T] \times \mathbb{R}^d.$$

Assumption 1 is a standard condition for SDEs, and we require the existence and uniqueness of a strong solution since we consider both the weak and strong simulators. Furthermore, we impose regularity conditions on the drift and volatility coefficient so that the usual moment estimates for the solution of SDE hold, which is needed in the subsequent analysis. In particular, we require stronger boundedness condition on the volatility coefficient for us to establish the desired statistical error guarantee later; otherwise, the underlying SDEs may be too volatile, losing the property required for the empirical loss functions used in the training to concentrate around its population counterpart.

Under Assumption 1, we present the property of ZOD estimators under our context.

Proposition 4 *Suppose Assumption 1 holds. Assume also that the coefficients $b, \sigma, g, f, V_n, G_n, H_n$ are smooth enough so that $\hat{V}_{n+1} \in C_p^{0, 3}([0, T] \times \mathbb{R}^d)$ and \mathbf{U}_n, f, g all have polynomial growth in x , uniformly in $t \in [0, T]$. Then $\frac{Z}{\epsilon}R^{t, X_t + \epsilon Z}(\mathbf{U}_n)$ is a biased estimator for $\nabla\hat{V}_{n+1}(t, X_t)$ conditioning on X_t , and there is a constant $C > 0$ independent of ϵ such that its bias and variance are upper bounded by*

$$\mathbb{E} \left[\int_0^T \left| \mathbb{E} \left[\frac{Z}{\epsilon} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \middle| X_t \right] - \nabla\hat{V}_{n+1}(t, X_t) \right|^2 dt \right] \leq C\epsilon^4, \quad \left| \text{Var} \left[\frac{Z}{\epsilon} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \right] \right| \leq C \frac{1}{\epsilon^2},$$

Moreover, if $\hat{V}_{n+1} \in C_p^{0, 4}([0, T] \times \mathbb{R}^d)$, then $\frac{ZZ^\top - I_d}{\epsilon^2} R^{t, X_t + \epsilon Z}(\mathbf{U}_n)$ is a biased estimator for $\nabla^2\hat{V}_{n+1}(t, X_t)$ conditioning on X_t , and there is a constant $C > 0$ independent of ϵ such that its bias and variance are upper bounded by

$$\mathbb{E} \left[\int_0^T \left| \mathbb{E} \left[\frac{ZZ^\top - I_d}{\epsilon^2} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \middle| X_t \right] - \nabla^2\hat{V}_{n+1}(t, X_t) \right|^2 dt \right] \leq C\epsilon^4, \\ \left| \text{Var} \left[\frac{ZZ^\top - I_d}{\epsilon^2} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \right] \right| \leq C \frac{1}{\epsilon^4}.$$

Proposition 4 describes the one-point ZOD estimators for the derivatives of the solution to a linear PDE and confirms that their biases are small and controlled by ϵ^2 . A significant limitation is that the variances of the estimators are generally unbounded, and in fact have the orders of ϵ^{-2} and ϵ^{-4} for the gradient and Hessian respectively, similarly to the classical ZOD in Lemma 1. We refer to them as ‘‘ZOD-1’’ in our subsequent numerical experiment, and will demonstrate that they usually produce worse results in learning derivatives.

Variance Reduction with Strong Simulators Theoretically, it is natural to investigate the symmetric versions (analogous to Lemma 2 and Lemma 3) of ZOD estimators for the gradient and Hessian of the solution to this linear PDE and inquire whether this variance reduction technique is helpful in the current PDE setting as well. It turns out that the answer depends on which type of simulators is accessible to us, i.e., when simulating multiple trajectories starting from *different* initial states x , $x + \epsilon Z$, and $x - \epsilon Z$, whether we can control the noises in the environment in a way that the corresponding rewards $R^{t,x}, R^{t,x+\epsilon Z}, R^{t,x-\epsilon Z}$ are highly correlated and therefore cancel the noises off to great extent.⁸ With a weak simulator, we need to query the simulator *separately*, one at a time, to generate the trajectories $X^{t,x}, X^{t,x+\epsilon Z}, X^{t,x-\epsilon Z}$. Because the Brownian motions in the underlying stochastic processes are independent, these trajectories, as well as the associated rewards $R^{t,x}, R^{t,x+\epsilon Z}, R^{t,x-\epsilon Z}$, are conditionally independent (conditioned on the realized Z), and hence we cannot expect the benefit of variance reduction. However, with a strong simulator, we are presented with trajectories of $X^{t,x}, X^{t,x+\epsilon Z}, X^{t,x-\epsilon Z}$ driven by the *same* Brownian path upon one single query, and $R^{t,x}, R^{t,x+\epsilon Z}, R^{t,x-\epsilon Z}$ are expected to be highly correlated, leading to additional noise cancellation in $R^{t,x+\epsilon Z} - R^{t,x-\epsilon Z}$ and $R^{t,x+\epsilon Z} + R^{t,x-\epsilon Z} - 2R^{t,x}$. The next Proposition 5 formalizes and indeed validates this idea.

Proposition 5 *Suppose Assumption 1 holds, and \mathbf{U}_n, f, g all have polynomial growth in x , uniformly in $t \in [0, T]$. Construct*

$$\begin{aligned} \hat{g} &:= \frac{Z}{2\epsilon} (R^{t, X_t + \epsilon Z}(\mathbf{U}_n) - R^{t, X_t - \epsilon Z}(\mathbf{U}_n)), \\ \hat{h} &:= \frac{ZZ^\top - I_d}{2\epsilon^2} (R^{t, X_t + \epsilon Z}(\mathbf{U}_n) + R^{t, X_t - \epsilon Z}(\mathbf{U}_n) - 2R^{t, X_t}(\mathbf{U}_n)). \end{aligned}$$

(a) *With a weak simulator as specified in Condition 1, if the coefficients $b, \sigma, g, f, V_n, G_n, H_n$ are smooth enough so that $\hat{V}_{n+1} \in C_p^{0,3}([0, T] \times \mathbb{R}^d)$, then \hat{g} is a biased estimator for $\nabla \hat{V}_{n+1}(t, X_t)$ conditioning on X_t , and its bias and variance are upper bounded by*

$$\mathbb{E} \left[\int_0^T \left| \mathbb{E}[\hat{g} \mid X_t] - \nabla \hat{V}_{n+1}(t, X_t) \right|^2 dt \right] \leq C\epsilon^4, \quad |\text{Var}[\hat{g}]| \leq C \frac{1}{\epsilon^2}.$$

Furthermore, if $\hat{V}_{n+1} \in C_p^{0,4}([0, T] \times \mathbb{R}^d)$, then \hat{h} is a biased estimator for $\nabla^2 \hat{V}_{n+1}(t, X_t)$ conditioning on X_t , and its bias and variance are upper bounded by

$$\mathbb{E} \left[\int_0^T \left| \mathbb{E}[\hat{h} \mid X_t] - \nabla^2 \hat{V}_{n+1}(t, X_t) \right|^2 dt \right] \leq C\epsilon^4, \quad |\text{Var}[\hat{h}]| \leq C \frac{1}{\epsilon^4}.$$

8. This idea resembles the classical simulation techniques of *control variates* and *antithetic variates* (cf. Glasserman 2004, Chapter 4) to use correlated samples for variance reduction.

(b) With a strong simulator as specified in Condition 2, if $b, \sigma \in C_b^{0,4}([0, T] \times \mathbb{R}^d)$, $\tilde{f} \in C_p^{0,3}([0, T] \times \mathbb{R}^d)$ where $\tilde{f}(t, x) := f(t, x, \mathbf{U}_n(t, x))$ and $g \in C_p^3(\mathbb{R}^d)$, then \hat{g} has the same order of biases as the one-point estimator in Proposition 4 but bounded variances as $\epsilon \rightarrow 0$. Furthermore, if $b, \sigma \in C_b^{0,5}([0, T] \times \mathbb{R}^d)$, $\tilde{f} \in C_p^{0,4}([0, T] \times \mathbb{R}^d)$ and $g \in C_p^4(\mathbb{R}^d)$, then \hat{h} has the same order of biases as the one-point estimator in Proposition 4 but bounded variances as $\epsilon \rightarrow 0$.

Proposition 5 reveals a unique bias–variance tradeoff associated with the multi-point ZOD estimators in our particular context and the critical difference between the weak and strong simulators. It turns out that the conventional bounded variance in Lemmas 2 and 3 no longer holds for the weak simulator and only holds for the strong simulator. In particular, a multi-point ZOD estimator does not reduce variance with a weak simulator, and its variance still has the same order as the one-point ZOD in Proposition 4. Hence, in general, there is no benefit to apply multi-point ZOD when only a weak simulator is accessible. We refer to the multi-point ZOD estimator with a strong simulator as “ZOD-m” in our subsequent numerical experiments, and will demonstrate that it produces much more stable results in learning derivatives while requiring fewer samples.

4 The Zeroth-Order Deep Learning Algorithm

Let $\mathbf{U} = (V, G, H)$ denote the triplet of functions. We define the approximating neural networks as:

$$\varphi^\theta := (\varphi_v^\theta, \varphi_g^\theta, \varphi_h^\theta),$$

where $\varphi_v^\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}$, $\varphi_g^\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$, and $\varphi_h^\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$ approximate the value, gradient, and Hessian, respectively. The parameters are denoted by $\theta \in \Theta \subset \mathbb{R}^p$.

We define the reward function starting from state (t, x) given a function triplet \mathbf{U} as:

$$R^{t,x}(\mathbf{U}) := g(X_T^{t,x}) + \int_t^T f(X_s^{t,x}, V(s, X_s^{t,x}), G(s, X_s^{t,x}), H(s, X_s^{t,x})) ds.$$

Crucially, the source term f is evaluated using the explicit outputs of G and H , rather than derivatives computed via auto-diff.

We implement the value iteration (2.5) as follows: At iteration n , let $\mathbf{U}_n = (V_n, G_n, H_n)$ be the trained networks from the previous step. We aim to learn \mathbf{U}_{n+1} such that:

1. $V_{n+1}(t, x) \approx \hat{V}_{n+1}(t, x) := \mathbb{E}[R^{t,x}(\mathbf{U}_n)]$.
2. $G_{n+1}(t, x) \approx \nabla \hat{V}_{n+1}(t, x)$.
3. $H_{n+1}(t, x) \approx \nabla^2 \hat{V}_{n+1}(t, x)$.

4.1 Loss Function Design

To train the candidate networks $\varphi = (\varphi_v, \varphi_g, \varphi_h)$ at iteration n , we minimize a composite loss function that enforces consistency between the networks and the stochastic targets derived from \mathbf{U}_n :

$$L(\varphi, \mathbf{U}_n) := L_v(\varphi_v, \mathbf{U}_n) + L_g(\varphi_g, \mathbf{U}_n) + L_h(\varphi_h, \mathbf{U}_n),$$

where each individual loss will be defined below, with $\beta > 0$ some fixed constant.

4.1.1 VALUE LOSS

The value network φ_v^θ is trained to match the conditional expectation of the reward. The loss is defined as the mean squared error (MSE) against the stochastic realization:

$$L_v(\varphi_v, \mathbf{U}_n) := \beta \mathbb{E} \left[\int_0^T e^{\beta s} |\varphi_v(s, X_s) - R^{s, X_s}(\mathbf{U}_n)|^2 ds \right].$$

Minimizing this loss implies $\varphi_v(t, x) \approx \mathbb{E}[R^{t, x}(\mathbf{U}_n)]$.

4.1.2 ZERO-ORDER DERIVATIVES LOSSES

Since \hat{V}_{n+1} is defined as an expectation, its derivatives are not directly available. We employ the one-point ZOD estimators (Proposition 4) to construct biased targets using perturbed trajectories.⁹ Let $Z \sim \mathcal{N}(0, I_d)$ be a perturbation vector independent of the trajectory X .

Gradient Loss. We rely on the relationship $\nabla \mathbb{E}[R^{t, x}] \approx \mathbb{E}[\frac{Z}{\epsilon} R^{t, x + \epsilon Z}]$. The gradient network φ_g^θ minimizes:

$$L_g(\varphi_g^\theta, \mathbf{U}_n) := \beta \mathbb{E} \left[\int_0^T e^{\beta t} \left| \varphi_g^\theta(t, X_t) - \frac{Z}{\epsilon} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \right|^2 dt \right].$$

Hessian Loss. Similarly, the Hessian network φ_h^θ minimizes the error against the second-order ZOD estimator, where the norm stands for the Frobenius norm for matrices:

$$L_h(\varphi_h^\theta, \mathbf{U}_n) := \mathbb{E} \left[\int_0^T e^{\beta t} \left| \varphi_h^\theta(t, X_t) - \frac{ZZ^\top - I_d}{\epsilon^2} R^{t, X_t + \epsilon Z}(\mathbf{U}_n) \right|^2 dt \right].$$

4.2 Discretization and Implementation

In the numerical implementation, expectations are estimated via Monte Carlo sampling. We define a time grid $\mathcal{G} := \{0 = t_0 < \dots < t_m = T\}$ with steps Δt_j . Let $\{X^{(b)}\}_{b=1}^B$ denote a batch of base trajectories starting from $(0, x_0)$.

For the value loss, the empirical reward along a base trajectory is computed using the frozen networks \mathbf{U}_n :

$$\tilde{R}^{t_j, X_{t_j}^{(b)}}(\mathbf{U}_n) := g(X_T^{(b)}) + \sum_{i=j}^{m-1} f \left(X_{t_i}^{(b)}, V_n(t_i, X_{t_i}^{(b)}), G_n(t_i, X_{t_i}^{(b)}), H_n(t_i, X_{t_i}^{(b)}) \right) \Delta t_i. \quad (4.1)$$

To compute the derivative losses, for each sample b and time step j , we generate a random perturbation $Z^{(b)} \sim \mathcal{N}(0, I_d)$ and simulate (observe) one perturbed trajectory $\{X^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}\}_{s \geq t_j}$ starting from $X_{t_j}^{(b)} + \epsilon Z^{(b)}$. The reward along this perturbed path is:

$$\begin{aligned} \tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n) &:= g(X_T^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}) \\ &+ \sum_{i=j}^{m-1} f \left(X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}, V_n(t_i, X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}), G_n(t_i, X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}), H_n(t_i, X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}) \right) \Delta t_i. \end{aligned} \quad (4.2)$$

9. Here we present the one-point ZOD estimators for illustration. Multi-point ZOD à la Proposition 5 can be discussed similarly.

The tractable empirical loss functions are:

$$\begin{aligned}\tilde{L}_v^{(B)}(\varphi_v, \mathbf{U}_n) &:= \beta \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left(\varphi_v(t_j, X_{t_j}^{(b)}) - \tilde{R}^{t_j, X_{t_j}^{(b)}}(\mathbf{U}_n) \right)^2 \Delta t_j, \\ \tilde{L}_g^{(B)}(\varphi_g, \mathbf{U}_n) &:= \beta \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left| \varphi_g(t_j, X_{t_j}^{(b)}) - \frac{Z^{(b)}}{\epsilon} \tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n) \right|^2 \Delta t_j, \\ \tilde{L}_h^{(B)}(\varphi_h, \mathbf{U}_n) &:= \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left| \varphi_h(t_j, X_{t_j}^{(b)}) - \frac{Z^{(b)}(Z^{(b)})^\top - Id}{\epsilon^2} \tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n) \right|^2 \Delta t_j.\end{aligned}$$

The empirical total loss is defined by

$$\tilde{L}^{(B)}(\varphi, \mathbf{U}_n) := \tilde{L}_v^{(B)}(\varphi_v, \mathbf{U}_n) + \tilde{L}_g^{(B)}(\varphi_g, \mathbf{U}_n) + \tilde{L}_h^{(B)}(\varphi_h, \mathbf{U}_n). \quad (4.3)$$

The complete procedure is summarized in Algorithm 1.

5 Convergence Analysis

In this section, we establish the convergence properties of the proposed zeroth-order based learning algorithm. Let $p(t, \cdot)$ denote the probability density function of X_t . We define the following weighted norms:

$$\|v\|_{L_2(p)}^2 := \int_{\mathbb{R}^d} |v(t, x)|^2 p(t, x) dx, \quad \|v\|_\beta^2 := \int_0^T e^{\beta t} \int_{\mathbb{R}^d} |v(t, x)|^2 p(t, x) dx dt,$$

as well as the mixed derivative norm:

$$\|v\|_{\beta, \text{mix}}^2 := \beta \|v\|_\beta^2 + \beta \|\nabla v\|_\beta^2 + \|\nabla^2 v\|_\beta^2.$$

For a function triplet $\mathbf{U} := (V, G, H)$, we abuse the notation slightly and define its corresponding norm as:

$$\|\mathbf{U}\|_{\beta, \text{mix}}^2 := \beta \|V\|_\beta^2 + \beta \|G\|_\beta^2 + \|H\|_\beta^2.$$

We formally define the solution mapping \mathcal{S} that maps a coefficient triplet to a solution triplet. Let $\mathbf{U} = (V, G, H)$ be a given triplet in an appropriate Hilbert space. Let u be the unique classical solution to the linear parabolic PDE:

$$\begin{cases} \partial_t u + \mathcal{L}u + f(x, V(t, x), G(t, x), H(t, x)) = 0, & (t, x) \in [0, T) \times \mathbb{R}^d, \\ u(T, x) = g(x), & x \in \mathbb{R}^d. \end{cases}$$

The solution mapping \mathcal{S} is then defined as the operator that returns the solution and its spatial derivatives:

$$\mathcal{S}(\mathbf{U}) := (u, \nabla u, \nabla^2 u).$$

Under this definition, the fixed-point problem $\mathbf{u}^* = \mathcal{S}(\mathbf{u}^*)$ is equivalent to the original nonlinear PDE (2.1) with v^* being the solution of PDE and $\mathbf{u}^* = (v^*, \nabla v^*, \nabla^2 v^*)$.

To proceed with the analysis, we introduce the following contraction assumption on \mathcal{S} .

Algorithm 1 Zeroth-Order PDE Solver via Independent Networks

- 1: **Input:** Function f , terminal condition g , time horizon T , initial state x_0 , time grid $\mathcal{G} = \{0 = t_0 < \dots < t_m = T\}$ with $\Delta t_j = t_{j+1} - t_j$, batch size B , outer iterations N_{out} , gradient steps per iteration K_{gd} , learning rate η , perturbation size ϵ .
 - 2: **Required Program:** A simulator generating sample trajectories whose infinitesimal generator is \mathcal{L} .
 - 3: **Initialize:** Independent neural networks $\mathbf{U}_0 = (V_0, G_0, H_0)$, parameter θ_0 .
 - 4: **for** $n = 0$ to $N_{\text{out}} - 1$ **do**
 - 5: Initialize candidate networks $\varphi^\theta = (\varphi_v^\theta, \varphi_g^\theta, \varphi_h^\theta)$ with parameters $\theta = \theta_n$
 - 6: **for** $k = 1$ to K_{gd} **do**
 - 7: Observe B trajectories $\{X_s^{(b)}\}_{s \in [0, T], b=1, \dots, B}$ starting from $(0, x_0)$.
 - 8: For $0 \leq j \leq m$, $1 \leq b \leq B$, evaluate $G_n(t_j, X_{t_j}^{(b)})$ and $H_n(t_j, X_{t_j}^{(b)})$ using frozen networks.
 - 9: Compute empirical reward random variable $\tilde{R}^{t_j, X_{t_j}^{(b)}}(\mathbf{U}_n)$ by (4.1).
 - 10: Compute $\tilde{L}_v^{(B)} := \beta \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left(\tilde{R}^{t_j, X_{t_j}^{(b)}}(\mathbf{U}_n) - \varphi_v^\theta(t_j, X_{t_j}^{(b)}) \right)^2 \Delta t_j$.
 - 11: Sample $\{Z^{(b)}\}_{b=1}^B \sim \mathcal{N}(0, I_d)$, for each $0 \leq j \leq m$, $1 \leq b \leq B$, observe one trajectory $\left\{ X_t^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}} \right\}_{t_j \leq t \leq T}$ starting at $(t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)})$.
 - 12: For $0 \leq j \leq m$, $1 \leq b \leq B$, $j \leq i \leq m$, evaluate $G_n(t_i, X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}})$ and $H_n(t_i, X_{t_i}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}})$ on the perturbed paths.
 - 13: Compute perturbed reward random variable $\tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n)$ by (4.2).
 - 14: Compute the derivatives loss by zeroth-order estimators:

$$\tilde{L}_g^{(B)} := \beta \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left(\frac{Z^{(b)}}{\epsilon} \tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n) - \varphi_g^\theta(t_j, X_{t_j}^{(b)}) \right)^2 \Delta t_j,$$

$$\tilde{L}_h^{(B)} := \frac{1}{B} \sum_{b=1}^B \sum_{j=0}^{m-1} e^{\beta t_j} \left(\frac{Z^{(b)}(Z^{(b)})^\top - I_d}{\epsilon^2} \tilde{R}^{t_j, X_{t_j}^{(b)} + \epsilon Z^{(b)}}(\mathbf{U}_n) - \varphi_h^\theta(t_j, X_{t_j}^{(b)}) \right)^2 \Delta t_j.$$
 - 15: Total Loss: $\tilde{L}^{(B)} := \tilde{L}_v^{(B)} + \tilde{L}_g^{(B)} + \tilde{L}_h^{(B)}$
 - 16: Update $\theta \leftarrow \theta - \eta D_\theta \tilde{L}^{(B)}$.
 - 17: **end for**
 - 18: Set $\theta_{n+1} \leftarrow \theta$ and $\mathbf{U}_{n+1} \leftarrow \varphi^{\theta_{n+1}}$
 - 19: **end for**
 - 20: **Output:** Trained networks $\mathbf{U}_{N_{\text{out}}} = (V_{N_{\text{out}}}, G_{N_{\text{out}}}, H_{N_{\text{out}}})$
-

Assumption 2 (Contraction) *There exist constants $\beta > 0$ and $\gamma < 1$, such that for any two triplets $\mathbf{U}_1 = (V_1, G_1, H_1)$ and $\mathbf{U}_2 = (V_2, G_2, H_2)$, the following inequality holds*

$$\|\mathcal{S}(\mathbf{U}_1) - \mathcal{S}(\mathbf{U}_2)\|_{\beta, \text{mix}} \leq \gamma \|\mathbf{U}_1 - \mathbf{U}_2\|_{\beta, \text{mix}}.$$

In our context, Assumption 2 represents the fundamental condition required for the well-posedness and convergence of the underlying value iteration scheme itself, independent of the numerical approximation errors. While establishing the necessary and sufficient conditions for general nonlinear PDEs to satisfy this contraction property is beyond the scope of this paper, we will provide a rigorous justification for its validity in an important and relevant setting. Specifically, in Appendix A we will study the case of Langevin dynamics with a stationary distribution and present a sufficient condition for Assumption 2.

5.1 Approximate Value Iteration

In this subsection, we analyze the propagation of learning errors through the iterative process and establish a global error bound for the proposed algorithm.

Let \mathcal{H} denote the space of neural network triplets. We consider the empirical risk minimization (ERM) problem (cf. Shalev-Shwartz and Ben-David 2014) at iteration n . Let \mathbf{U}_n be the empirical minimizer:

$$\mathbf{U}_n \in \arg \min_{\varphi \in \mathcal{H}} \tilde{L}^{(B)}(\varphi, \mathbf{U}_{n-1}), \quad (5.1)$$

where $\tilde{L}^{(B)}$ is the empirical loss defined in (4.3). Under the ERM framework, omitting the optimization error, at each iteration $n \geq 1$, the algorithm produces a neural network triplet $\mathbf{U}_n = (V_n, G_n, H_n)$, which is the ERM minimizer, intended to approximate the theoretical target $\mathcal{S}(\mathbf{U}_{n-1})$.

We define the *one-step approximation error* at iteration n as the deviation between the trained network and the exact image of the previous iterate under the solution map \mathcal{S} :

$$E_n := \|\mathbf{U}_n - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}. \quad (5.2)$$

The quantity E_n aggregates two sources of error: the statistical error arising from the Monte Carlo estimation of the reward and its derivatives (via ZOD estimators), and the approximation error associated with the neural network hypothesis space (optimization and statistical error) to be defined momentarily.

The following theorem characterizes the convergence of the sequence $\{\mathbf{U}_n\}_{n \geq 0}$ to the ground truth \mathbf{u}^* .

Theorem 6 *Suppose that Assumption 2 holds with a contraction constant $\gamma \in [0, 1)$. Let $\{\mathbf{U}_n = (V_n, G_n, H_n)\}_{n \geq 0}$ be the sequence of function triplets, which are ERM minimizers defined by (5.1). Then, almost surely, for any $n \geq 1$, the total error is bounded by:*

$$\|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} \leq \gamma^n \|\mathbf{U}_0 - \mathbf{u}^*\|_{\beta, \text{mix}} + \sum_{k=1}^n \gamma^{n-k} E_k. \quad (5.3)$$

Theorem 6 presents the convergence result of the proposed method as an *approximate value iteration*. The error bound consists of two distinct components:

1. The *contraction term* $\gamma^n \|\mathbf{U}_0 - \mathbf{u}^*\|_{\beta, \text{mix}}$, which decays exponentially. This term captures the inherent stability of the value iteration for the underlying PDE.
2. The *accumulation term* $\sum_{k=1}^n \gamma^{n-k} E_k$, which represents the accumulated error. The magnitude of E_k is determined by the expressivity of the neural networks (approximation error) and the variance of the ZOD estimators (statistical error).

This result highlights a critical trade-off: while the contraction constant γ is dictated and fixed by the PDE physics, the approximation accuracy E_k can be controlled by increasing the network capacity and the batch size B , and properly tuning the ZOD perturbation ϵ . In Section 5.2, we will provide a detailed analysis of E_k , explicitly characterizing its dependence on the sample size B and perturbation parameter ϵ .

5.2 One-Step Sample Complexity Analysis

In this subsection, we provide an analysis of the one-step approximation error E_n . Recall from (5.2) that $E_n = \|\mathbf{U}_n - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}$, where \mathbf{U}_n is the trained network triplet (the ERM minimizer) and $\mathcal{S}(\mathbf{U}_{n-1})$ is the ground truth target defined by the solution mapping.

To characterize the learning target, we first introduce the *ZOD solution mapping* \mathcal{S}^ϵ , which maps the previous iterate to the expected values of the zeroth-order estimators. Namely, $\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) := (V^\epsilon, G^\epsilon, H^\epsilon)$ is the target triplet defined by the conditional expectations:

$$\begin{aligned} V^\epsilon(t, x) &:= \mathbb{E} [R^{t,x}(\mathbf{U}_{n-1})] \equiv \hat{V}_n(t, x), \\ G^\epsilon(t, x) &:= \mathbb{E} \left[\frac{Z}{\epsilon} R^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) \right], \\ H^\epsilon(t, x) &:= \mathbb{E} \left[\frac{ZZ^\top - I_d}{\epsilon^2} R^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) \right]. \end{aligned}$$

Note that $\mathcal{S}^\epsilon(\mathbf{U}_{n-1})$ represents the “smoothed” solution induced by the Gaussian perturbation, which serves as the proxy for the true solution $\mathcal{S}(\mathbf{U}_{n-1})$.

Under the strong simulator in Condition 2, the derivative components of the empirical loss can alternatively be evaluated by the multi-point ZOD estimators \hat{g} and \hat{h} in Proposition 5. In this multi-point construction, the rewards entering each estimator are generated with the same Brownian path upon a single query. By the symmetry of Z and the linearity of expectation, these multi-point ZOD estimators have the same conditional means as G^ϵ and H^ϵ . Thus the population target remains $\mathcal{S}^\epsilon(\mathbf{U}_{n-1})$; in other words, the multi-point construction changes the variance and the empirical-process analysis, but not the target being learned.

The following lemma establishes the relationship between the population loss function and this ZOD target.

Lemma 7 *For any triplet φ , the population loss with respect to the previous iterate \mathbf{U}_{n-1} decomposes as:*

$$L(\varphi, \mathbf{U}_{n-1}) = \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 + C,$$

where C is a constant independent of φ .

As a direct consequence of Lemma 7, the triplet $\mathcal{S}^\epsilon(\mathbf{U}_{n-1})$ is the unique minimizer of the population loss over the space of all measurable functions:

$$\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) \in \arg \min_{\varphi} L(\varphi, \mathbf{U}_{n-1}).$$

Hence, the learning process can be viewed as approximating the ZOD target $\mathcal{S}^\epsilon(\mathbf{U}_{n-1})$ via empirical risk minimization.

We now define the network architecture and hypothesis space used in this paper. For a smooth activation function $\rho : \mathbb{R} \rightarrow \mathbb{R}$, we use the same notation $\rho(z)$ for its element-wise application to a vector z .

Definition 8 *Given input dimension d_{in} , output dimension d_{out} , depth L , and width vector $\mathbf{m} = (m_0, m_1, \dots, m_L)$ with $m_0 = d_{in}$ and $m_L = d_{out}$, we define the multilayer perceptron (MLP) $\Phi(\cdot; \vartheta) : \mathbb{R}^{d_{in}} \rightarrow \mathbb{R}^{d_{out}}$ by*

$$\Phi(x; \vartheta) := \mathcal{A}_L \circ \rho \circ \mathcal{A}_{L-1} \circ \dots \circ \rho \circ \mathcal{A}_1(x),$$

where each $\mathcal{A}_l : \mathbb{R}^{m_{l-1}} \rightarrow \mathbb{R}^{m_l}$ is the affine map $\mathcal{A}_l(z) = W_l z + b_l$. The parameter vector ϑ collects all trainable parameters of this MLP, namely $\vartheta = (W_l, b_l)_{l=1}^L$. The hypothesis space is defined by

$$\mathcal{H} := \{\varphi_\theta = (V_\theta, G_\theta, H_\theta) \mid \theta \in \Theta\},$$

where $V_\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}$, $G_\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$, and $H_\theta : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$ are three possibly different MLPs. The output of H_θ has dimension d^2 and is reshaped into a $d \times d$ matrix.

Note that we use the same symbol θ for all the three networks: θ denotes the collection of their trainable parameters, and when appearing in one of V_θ , G_θ , or H_θ it refers to the corresponding subcollection of parameters.

Assumption 3 *The activation function satisfies $\rho \in C_b^5(\mathbb{R})$. The parameter set $\Theta \subset \mathbb{R}^p$ is compact and satisfies $|\theta|_\infty \leq R$ for all $\theta \in \Theta$, where p is the total number of trainable parameters in the three networks.*

Under the above construction and assumption, due to the smoothness of ρ and the compactness of Θ , all functions in \mathcal{H} and their partial derivatives up to the fourth order with respect to x are uniformly bounded.

Next we introduce the regularity assumption on the expected reward function induced by any candidate in the hypothesis space. This is crucial for establishing the one-step sample complexity result.

For any $\varphi \in \mathcal{H}$, denote

$$J_1(t, x; \varphi) := \mathbb{E} [R^{t,x}(\varphi)], \quad J_2(t, x; \varphi) := \mathbb{E} [(R^{t,x}(\varphi))^2], \quad \tilde{f}(t, x; \varphi) := f(t, x, \varphi(t, x)).$$

Assumption 4 *The functions $J_1(\cdot, \cdot; \varphi) \in C_p^{1,4}([0, T] \times \mathbb{R}^d)$, $J_2(\cdot, \cdot; \varphi) \in C_p^{1,2}([0, T] \times \mathbb{R}^d)$, $\tilde{f} \in C_p^{1,2}([0, T] \times \mathbb{R}^d)$ uniformly over the hypothesis space $\varphi \in \mathcal{H}$ in the sense that their derivatives have polynomial growth with constants independent of φ . The terminal function g has polynomial growth.*

Assumption 4 is a regularity condition on the solutions to the linear PDE in the value iterate. Typical sufficient conditions for it to hold include stronger regularity conditions on the SDE coefficients and/or uniform ellipticity.

Theorem 9 *Suppose Assumptions 1, 3, and 4 hold. Let $0 < \epsilon < 1$ and $0 < \delta < 1$, and define $L_{B,\delta} := \log B + \log(1/\delta)$. Let \mathcal{P} be a polynomial determined by the architecture of \mathcal{H} , C be a constant depending only on the PDE coefficients and the constants in the above assumptions. Let q be a constant depending only on the polynomial growth orders of the functions in Assumption 4.*

(a) *With the weak simulator in Condition 1, if the derivative terms in the empirical loss are evaluated using the one-point ZOD estimators, then, with probability at least $1 - \delta$,*

$$E_n^2 \leq C \left(\underbrace{\frac{1}{N\epsilon^4}}_{\text{Discretization Error}} + \underbrace{\epsilon^4}_{\text{ZOD Bias}} + \underbrace{\left(1 + \sqrt{\log(1/\delta)}\right) \frac{\mathcal{P}(d)\sqrt{\log(1/\epsilon)}L_{B,\delta}^q}{B^{1/2}\epsilon^4}}_{\text{Statistical Error}} \right) + 2 \underbrace{\inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2}_{\text{Approximation Error}}.$$

(b) *Suppose further that $b, \sigma \in C_b^{0,5}([0, T] \times \mathbb{R}^d)$, $g \in C_p^4(\mathbb{R}^d)$ and $f(t, x, \varphi(t, x)) \in C_p^{1,5}([0, T] \times \mathbb{R}^d)$ uniformly over the hypothesis space $\varphi \in \mathcal{H}$ in the sense that their derivatives have polynomial growth with constants independent of φ . With the strong simulator in Condition 2, if the derivative terms in the empirical loss are evaluated using the multi-point estimators stipulated in Proposition 5, then, with probability at least $1 - \delta$,*

$$E_n^2 \leq C \left(\underbrace{\frac{1}{N}}_{\text{Discretization Error}} + \underbrace{\epsilon^4}_{\text{ZOD Bias}} + \underbrace{\left(1 + \sqrt{\log(1/\delta)}\right) \frac{\mathcal{P}(d)\Gamma_{\epsilon, B, \delta}}{B^{1/2}}}_{\text{Statistical Error}} \right) + 2 \underbrace{\inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2}_{\text{Approximation Error}},$$

where $\Gamma_{\epsilon, B, \delta} := \min \left\{ \epsilon^{-4} \sqrt{\log(1/\epsilon)} L_{B, \delta}^q, \exp(C \sqrt{L_{B, \delta}}) \right\}$.

Theorem 9 provides high-probability upper bounds on the one-step error, under both the weak and strong simulators, in terms of four components: the discretization error from approximating the time integral, the ZOD bias from using ZOD-based derivative representations instead of exact stochastic representations, the statistical error from the empirical loss due to a finite sample size, and the approximation error of the neural-network family. Several points are worth noting. First, the discretization error is controlled by the number of time-grid points used to compute the integrals. Since we use the simplest forward Euler scheme, this term is of order N^{-1} , which can be improved by adopting a higher-order numerical integration scheme. Importantly, increasing the number of time-grid points does

not introduce instability into the algorithm or the loss function. Second, the ZOD bias is controlled by the bandwidth and is of order ϵ^4 in both simulator settings. This part of the error follows from Propositions 4 and 5. Third, the statistical error is jointly controlled by the sample size and the complexity of the neural-network class. Our statistical bound is conservative: the squared one-step error contains a sample-size term of order $B^{-1/2}$ and depends polynomially on the dimension of the state variable; this rate arises from our use of a global Rademacher-complexity bound for the empirical-loss term.¹⁰ Finally, the approximation error is in terms of the weighted Sobolev norm of the neural network classes, which reflects the quality of the neural network classes chosen.

Moreover, under the weak simulator, the ZOD estimator has exploding variances. As a consequence, both the discretization error and the statistical error contain the factor ϵ^{-4} , which substantially worsens the bound and creates a bias–variance tradeoff against the ZOD bias. Ignoring logarithmic factors, the dominant terms are of the order $(N^{-1} + \mathcal{P}(d)B^{-1/2})\epsilon^{-4} + \epsilon^4$. This suggests choosing $\epsilon = O((N^{-1} + \mathcal{P}(d)B^{-1/2})^{1/8})$ to yield the squared one-step error $E_n^2 = O((N^{-1} + \mathcal{P}(d)B^{-1/2})^{1/2})$, up to logarithmic factors and the approximation error.

By contrast, under the strong simulator, the variance of the multi-point ZOD estimator is bounded uniformly in ϵ . Hence neither the discretization error nor the statistical error suffers from the ϵ^{-4} blow-up. We note, however, that to cancel this ϵ^{-4} term, one relies on the tangent process. Since the global Rademacher complexity argument requires a bounded loss class, a truncation argument is needed. The tangent process typically has a heavier tail than the original state process, which leads to the factor $\exp(C\sqrt{\log B + \log(1/\delta)})$ in the statistical error bound. Although this factor is worse than the logarithmic factor appearing in the weak one-point estimate, it is still sub-polynomial in both B and $1/\delta$, because $\exp(C\sqrt{L_{B,\delta}}) = (B/\delta)^{o(1)}$. Consequently, in the asymptotic sense, choosing $\epsilon = O((N^{-1} + \mathcal{P}(d)B^{-1/2})^{1/4})$ balances the ZOD bias with the discretization and statistical errors, and gives the squared one-step error $E_n^2 = O(N^{-1} + \mathcal{P}(d)B^{-1/2})$, up to the approximation error. To sum, compared with the weak simulator, the strong simulator gives a substantially better asymptotic order by removing the unfavorable ϵ^{-4} dependence.

Combining our analysis for the approximate value iteration and one-step sample complexity results, Theorem 10 presents the total error of the algorithm up to the n -th value iterate with any fixed bandwidth ϵ , the number of time discretization grids N , and the sample size B for training.

Theorem 10 *Suppose Assumptions 1–4 hold. Let $0 < \epsilon < 1$ and $0 < \delta < 1$, and write $L_{B,\delta,n} := \log B + \log(n/\delta)$. Let \mathcal{P} be a polynomial determined by the architecture of \mathcal{H} , C be a constant depending only on the PDE coefficients and the constants in the assumptions. Let q be a constant depending only on the polynomial growth orders of the functions in Assumption 4.*

10. Sharper localized critical radius analyses are possible in principle, but they require additional technical assumptions and substantially more involved arguments, which are not the focus of this paper; see, e.g., Bartlett et al. (2005); Wainwright (2019).

(a) Under the conditions of Theorem 9-(a), with probability at least $1 - \delta$, we have

$$\begin{aligned} \|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} &\leq \gamma^n \|\mathbf{U}_0 - \mathbf{u}^*\|_{\beta, \text{mix}} + \sqrt{2} \sum_{k=1}^n \gamma^{n-k} \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{k-1})\|_{\beta, \text{mix}} \\ &+ C \frac{1 - \gamma^n}{1 - \gamma} \left(N^{-1/2} \epsilon^{-2} + \epsilon^2 + \left(1 + (\log(n/\delta))^{1/4}\right) \frac{\sqrt{\mathcal{P}(d)} (\log(1/\epsilon))^{1/4} L_{B, \delta, n}^{q/2}}{B^{1/4} \epsilon^2} \right). \end{aligned}$$

(b) Under the conditions of Theorem 9-(b), with probability at least $1 - \delta$, we have

$$\begin{aligned} \|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} &\leq \gamma^n \|\mathbf{U}_0 - \mathbf{u}^*\|_{\beta, \text{mix}} + \sqrt{2} \sum_{k=1}^n \gamma^{n-k} \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{k-1})\|_{\beta, \text{mix}} \\ &+ C \frac{1 - \gamma^n}{1 - \gamma} \left(N^{-1/2} + \epsilon^2 + \left(1 + (\log(n/\delta))^{1/4}\right) \frac{\sqrt{\mathcal{P}(d)} \Gamma_{\epsilon, B, \delta, n}}{B^{1/4}} \right), \end{aligned}$$

where $\Gamma_{\epsilon, B, \delta, n} := \min \left\{ \epsilon^{-4} \sqrt{\log(1/\epsilon)} L_{B, \delta, n}^q, \exp(C \sqrt{L_{B, \delta, n}}) \right\}$.

6 Numerical Experiments

In this section, we investigate the numerical performance of the proposed zeroth-order solver in learning the full solution triplet $(v^*, \nabla v^*, \nabla^2 v^*)$.

Let $\mathbf{U}_n = (V_n, G_n, H_n)$ denote the learned value, gradient, and Hessian networks respectively after the n -th value iteration. Throughout the numerical experiments, we report empirical relative root mean squared errors (rRMSE) between the learned networks and the ground truth solution by evaluating the errors on a set of randomly generated test samples. Specifically, given test samples $z_i = (t_i, x_i)$ and a target quantity Q^* evaluated at these samples, define

$$\text{rRMSE}(\hat{Q}, Q^*) = \left(\frac{\sum_{i=1}^{N_{\text{test}}} |\hat{Q}(z_i) - Q^*(z_i)|^2}{\sum_{i=1}^{N_{\text{test}}} |Q^*(z_i)|^2} \right)^{1/2},$$

with the absolute value for scalar quantities, the Euclidean norm for vectors, and the Frobenius norm for matrices.¹¹ For the solution triplet, \hat{Q} is taken directly from V_n , G_n , or H_n and compared with v^* , ∇v^* , or $\nabla^2 v^*$, respectively. Note once again that our approach learns the three networks separately, and the derivatives are not obtained by auto-differentiating the learned value network V_n .

For the PDE-solving experiments reported below, we use a short initialization stage as a pre-training stage. With the initial scalar network V_0 fixed, the derivative networks G_0 and H_0 are pretrained to fit $\nabla_x V_0$ and $\nabla_x^2 V_0$, respectively, where these derivatives are computed by automatic differentiation of V_0 on samples from the corresponding training

11. In our theoretical analysis, we rely on the weighted Sobolev norm that depends on a constant β . This constant exists in theory but is difficult to determine in experiments. Therefore, in the numerical study, we only calculate the unweighted Sobolev norm in training and present the results for function value, gradient, and Hessian in terms of the relative mean squared errors respectively.

law. This pretraining is used only to initialize the derivative networks. In the subsequent value iterations, the networks are warm-started: the parameters of $V_{n+1}, G_{n+1}, H_{n+1}$ are initialized from the trained parameters of V_n, G_n, H_n rather than from a fresh random initialization. All numerical experiments were run on a single NVIDIA RTX 3090 GPU with 24 GB memory. Code is available at <https://github.com/du-ouyang/Z04PDE>.

In the following subsections, we first present a simple toy example of a linear PDE in Subsection 6.1, which is effectively a regression problem explained in Subsection 3.2, to contrast our approach to the existing ones. In Subsections 6.2 and 6.3, we compare our method to the deep Picard iteration (DPI) in Han et al. (2026) on benchmark nonlinear PDEs where the accuracy can be measured against exact solutions. For a fairer comparison in these DPI benchmarks, we use the multi-point ZOD estimators (ZOD-m) for gradients and Hessians (see Proposition 5). This is because the DPI method is model-based, which assumes knowing the operator \mathcal{L} , a much stronger assumption including the availability of a strong simulator as a special case. Moreover, the SDEs in these benchmarks can be exactly simulated without numerical error. Hence, the time integrals appearing both in the training loss and in the reward random variable R can be estimated by sampling t uniformly on $[0, T]$ and then sampling X_t exactly from its marginal law.¹² In this way, the reported errors isolate the learning and ZOD-estimation errors, without having to consider the time discretization errors.

6.1 A Linear PDE: Regression

We consider the following (one-dimensional) linear PDE:

$$\partial_t v(t, x) + \frac{1}{2} \sigma^2 \partial_{xx} v(t, x) = 0, \quad (t, x) \in [0, T] \times \mathbb{R}, \quad (6.1)$$

with terminal condition $v(T, x) = g(x)$.

We are interested only in the time-0 value, gradient, and Hessian, namely $u(x) = v(0, x)$, $u'(x) = \partial_x v(0, x)$, and $u''(x) = \partial_{xx} v(0, x)$. By the Feynman–Kac formula, this is equivalent to computing the conditional expectation

$$u(x) = \mathbb{E}[g(X_T) \mid X_0 = x],$$

where $dX_t = \sigma dW_t$. It is also equivalent to a regression task introduced in Subsection 3.2, with $Y = g(X + \sigma\xi)$.

Set the terminal function g as

$$g(y) = A_1 \sin(\alpha y) + A_2 \cos(\beta y) + \sum_{j=1}^J a_j \exp\left(-\frac{(y - c_j)^2}{2\ell_j^2}\right),$$

12. More precisely, the time integral in the loss is evaluated by first sampling $t \sim \text{Unif}[0, T]$ and then sampling X_t exactly from its marginal law, using the identity $\int_0^T e^{\beta t} \mathbb{E}[\ell(t, X_t)] dt = T \mathbb{E}[e^{\beta t} \ell(t, X_t)]$. Similarly, for the reward $R^{t,x} = g(X_T^{t,x}) + \int_t^T f(s, X_s^{t,x}, \mathbf{U}(s, X_s^{t,x})) ds$, the integral term is estimated by first sampling $s \sim \text{Unif}[t, T]$ and then sampling $X_s^{t,x}$ exactly from the transition law, so that $\int_t^T \mathbb{E}[f(s, X_s^{t,x}, \mathbf{U}(s, X_s^{t,x}))] ds = (T - t) \mathbb{E}[f(s, X_s^{t,x}, \mathbf{U}(s, X_s^{t,x}))]$.

and the ground truth solution is available by Gaussian convolution:

$$u^*(x) = A_1 e^{-\alpha^2 \sigma^2 / 2} \sin(\alpha x) + A_2 e^{-\beta^2 \sigma^2 / 2} \cos(\beta x) + \sum_{j=1}^J a_j \frac{\ell_j}{s_j} \exp\left(-\frac{(x - c_j)^2}{2s_j^2}\right), \quad s_j^2 = \ell_j^2 + \sigma^2.$$

The ground truth derivatives can then be computed analytically from the above expression, denoted by $u^{*'}, u^{*''}$.

For the learning-then-differentiating paradigm, we first train a scalar network V_ϑ with the least-squares loss function:

$$\mathcal{L}_{\text{NN}}(\vartheta) = \frac{1}{B} \sum_{i=1}^B |V_\vartheta(x_i) - g(x_i + \sigma \xi_i)|^2,$$

where $x_i \sim \text{Unif}[-2, 2]$ and $\xi_i \sim \mathcal{N}(0, T)$. The derivative estimates are then obtained by automatic differentiation:

$$\widehat{\Delta}_{\text{NN}}(x) = \partial_x V_\vartheta(x), \quad \widehat{\Gamma}_{\text{NN}}(x) = \partial_{xx} V_\vartheta(x).$$

For our ZOD approach, we train three networks $(V_{\theta_v}, G_{\theta_g}, H_{\theta_h})$, parameterized collectively by θ . At each step, we sample $\{x_i\}_{i=1}^B \sim \text{Unif}[-2, 2]$, $\{z_i\}_{i=1}^B \sim \mathcal{N}(0, 1)$, and $\{\xi_i\}_{i=1}^B \sim \mathcal{N}(0, T)$. With the multi-point ZOD estimator, define

$$\begin{aligned} \widehat{V}_i &= g(x_i + \sigma \xi_i), \\ \widehat{G}_i &= \frac{z_i}{2\epsilon} [g(x_i + \epsilon z_i + \sigma \xi_i) - g(x_i - \epsilon z_i + \sigma \xi_i)], \\ \widehat{H}_i &= \frac{z_i^2 - 1}{2\epsilon^2} [g(x_i + \epsilon z_i + \sigma \xi_i) + g(x_i - \epsilon z_i + \sigma \xi_i) - 2g(x_i + \sigma \xi_i)]. \end{aligned}$$

The three networks are trained jointly by

$$\mathcal{L}_{\text{ZOD}}(\theta) = \frac{1}{B} \sum_{i=1}^B \left(|V_{\theta_v}(x_i) - \widehat{V}_i|^2 + |G_{\theta_g}(x_i) - \widehat{G}_i|^2 + |H_{\theta_h}(x_i) - \widehat{H}_i|^2 \right).$$

Note that the same Brownian noise ξ_i is reused in definition of the multi-point ZOD estimator \widehat{G}_i and \widehat{H}_i , which is available only with the strong simulator.

In the experiment, we use

$$\begin{aligned} \sigma &= 0.02, \quad T = 1, \quad A_1 = 0.22, \quad A_2 = 0.06, \quad \alpha = 1.3, \quad \beta = 4.7, \quad J = 7, \\ (c_j) &= (-1.55, -1.05, -0.62, -0.18, 0.24, 0.71, 1.28), \\ (\ell_j) &= (0.065, 0.055, 0.070, 0.060, 0.055, 0.065, 0.060), \\ (a_j) &= (0.035, -0.030, 0.032, 0.028, -0.034, 0.030, -0.027). \end{aligned}$$

Both auto-differentiation and ZOD methods use MLPs with four hidden layers of width 256 and tanh activation. The rRMSEs are evaluated on 1000 uniformly generated initial spatial points in $[-2, 2]$. NN-autodiff is trained with 10000 gradient steps and batch size

Table 1: **rRMSEs for the linear PDE (6.1).**

Method	Value Error	Gradient Error	Hessian Error	Time (s)
NN-autodiff	5.425×10^{-2}	4.532×10^{-1}	9.368×10^{-1}	47.2
ZOD-m	7.184×10^{-2}	6.751×10^{-2}	3.633×10^{-1}	44.1

$B = 32768$, learning rate 3×10^{-4} . Our ZOD estimator is trained with 5000 gradient steps, batch size $B = 16384$, learning rate 5×10^{-4} , and ZOD bandwidth $\epsilon = 0.01$.

Table 1 summarizes the rRMSEs of both approaches under comparable running times. Observe that although the value errors are of the same order, our ZOD approach yields substantially smaller derivative errors in both the first and second order and outperforms by at least 3 times. Judging from the magnitude of these errors, one might conclude that NN-autodiff, while lagging behind ZOD, could still achieve reasonable accuracy in the derivatives. However, rRMSE is an *averaged* measure (over the initial states) that may not fully capture the point-by-point errors of a learned function from its oracle. To visualize the latter, we draw diagnostic plots in Figure 1, where we put the ground truth solution and the learned functions by the two approaches together as functions of x . In terms of the learned value function, both methods produce very close fit to the true one, whereas the gap becomes significant for the gradient and Hessian. In particular, the ZOD-based derivatives follow the oracle solutions closely all the time, capturing almost every peak and trough as well as the shapes of the functions, but the ones by NN-autodiff smoothly vary over the whole interval and miss most of the peaks and troughs. Thereby, despite the small average errors in the gradient and Hessian, the learned derivative functions are not close at all with NN-autodiff. This example highlights the limitations of auto-differentiating a learned value function directly when accurate derivative approximation is required, and justifies our method that aims for theoretical guarantee in not only function values, but also in derivatives values.

6.2 A Semilinear PDE

We next consider a semilinear PDE studied in Han et al. (2026).

$$\partial_t u(t, x) + \frac{1}{2} \Delta u(t, x) + x \cdot \nabla u(t, x) - \frac{1}{2} |\nabla u(t, x)|^2 - d = 0, \quad (t, x) \in [0, T] \times \mathbb{R}^d, \quad (6.2)$$

with terminal condition $u(T, x) = g(x) := -\log p_0(x)$. Here p_0 is the density of a five-component Gaussian mixture

$$p_0(x) = \sum_{k=1}^5 \pi_k \mathcal{N}(x; \mu_0^{(k)}, \Sigma_0^{(k)}), \quad (6.3)$$

where the means $\mu_0^{(k)}$ are sampled uniformly from $[-1, 1]^d$, the covariance matrices are diagonal with $\Sigma_0^{(k)} = 2I_d$, and the weights π_k are positive and normalized with $\sum_{k=1}^5 \pi_k = 1$.

This equation has differential operator $\mathcal{L}u = \frac{1}{2} \Delta u$, which corresponds to the SDE

$$dX_t = dW_t, \quad X_0 = \xi, \quad (6.4)$$

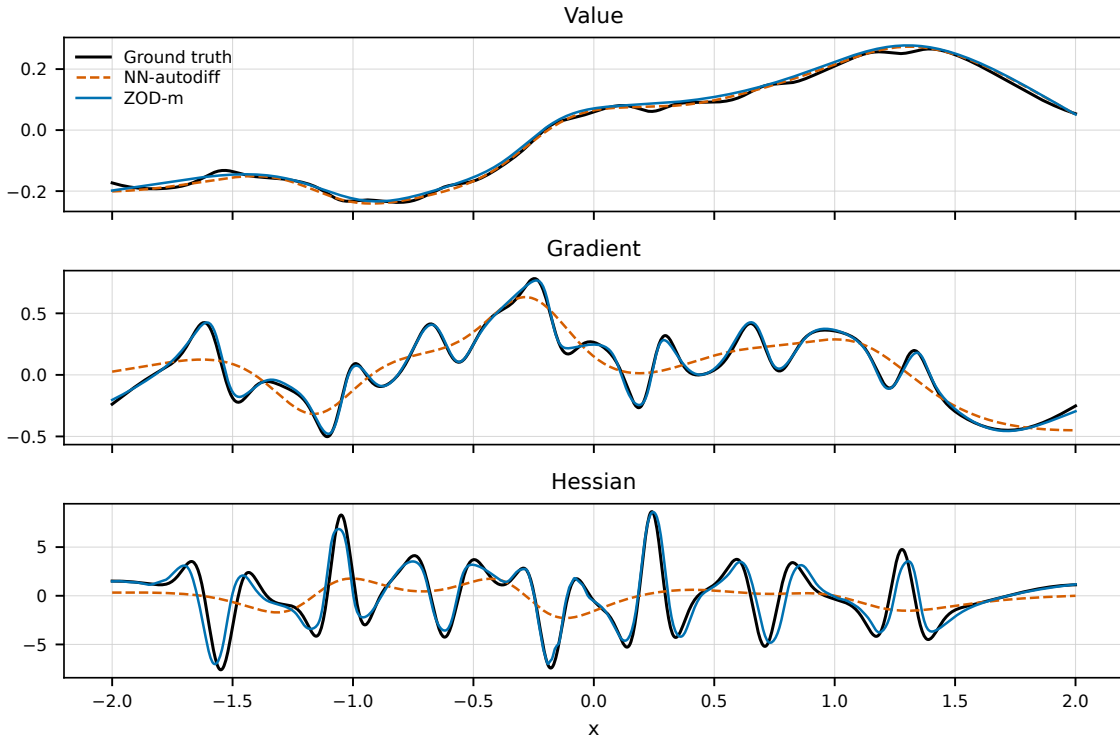


Figure 1: **The learned value function, gradient, and Hessian for example (6.1).** NN-autodiff learns the value function by the least-squares method and obtains derivatives by automatic differentiation. ZOD-m trains value, gradient, and Hessian networks directly using multi-point ZOD estimators.

and the nonlinear source term is $f(t, x, \nabla u) = x \cdot \nabla u(t, x) - \frac{1}{2} |\nabla u(t, x)|^2 - d$.

Then we can find the exact solution u^* of the PDE (6.2) as¹³

$$u^*(t, x) = -\log p(T - t, x),$$

where

$$p(t, x) = \sum_{k=1}^5 \pi_k \mathcal{N}(x; \mu_t^{(k)}, \Sigma_t^{(k)}),$$

with

$$\mu_t^{(k)} = e^{-t} \mu_0^{(k)}, \quad \Sigma_t^{(k)} = 2e^{-2t} I_d + \frac{1 - e^{-2t}}{2} I_d.$$

In our experiment, we set $d = 20, T = 1$. The above configuration is identical to that in Han et al. (2026) except that we lower the dimension to $d = 20$. This is because we also need to evaluation and compare the accuracy of the learned Hessian, which has an output

13. This is because we know the density $p(t, x)$ of \hat{X}_t , where $d\hat{X}_t = -\hat{X}_t dt + dW_t$, satisfies the Fokker–Planck equation $\partial_t p = \nabla \cdot (xp) + \frac{1}{2} \Delta p$. And we can verify that $u(t, x) = -\log p(T - t, x)$ satisfies PDE (6.2). Hence, if \hat{X}_t has the initial distribution $p_0(x)$, we can deduce its distribution time- t distribution $p(t, x)$.

dimension d^2 , whereas Han et al. (2026) only evaluate the gradient whose output dimension is d . For the SDE (6.4) used to generate training and test datasets, we set $\xi \sim \mathcal{N}(0, 4I_d)$ as in Han et al. (2026).

For a fair comparison, both ZOD and the DPI baseline use the same neural network framework from Han et al. (2026), which explicitly enforces the terminal condition. Denote

$$u_\theta(t, x) = (r_\eta(T-t) - r_0(0)) \langle N_\gamma(T-t, x), x \rangle + (1 - r_\eta(T-t) + r_\eta(0)) g\left(e^{-(T-t)/2} x\right), \quad (6.5)$$

where $N_\gamma(t, x) \in \mathbb{R}^d$ and $r_\eta(t) \in \mathbb{R}^+$ are neural networks. Consequently, $u_\theta(T, x) = g(x)$.

Note that we compare our ZOD approach with DPI in Han et al. (2026) on the 20-dimensional PDE (6.2), using the same Gaussian-mixture terminal density (6.3) and the same network (6.5) for the value function. This isolates the main algorithmic difference: how derivative information is represented and controlled over iterations.

For ZOD, we use the three-network model, where the value network uses the parameterization (6.5) and the gradient and Hessian networks are 4-layer MLPs with width 512 and ELU activations. We use the Adam optimizer with a learning rate 10^{-3} , batch size 16384, 10 value iterations, and 512 gradient updating steps per iteration. The derivative networks are pretrained for 5000 steps. The bandwidth of ZOD is $\epsilon = 0.02$.

For DPI, we use the configuration in Han et al. (2026). Specifically, the value network is again (6.5) with the same 4 hidden layers of width 512 and ELU activations. DPI uses 10 outer Picard iterations, batch size 512, data size 4096, and 16 epochs per outer iteration, with the Adam optimizer and a learning rate 10^{-3} . Gradient supervision is enabled with the same weight as in Han et al. (2026). Moreover, second-order information is obtained by differentiating the learned scalar value approximation.

Figure 2 reports the rRMSE of the learned value, gradient, and Hessian over iterations for the problem (6.2). The curves show the means over three random seeds, and the shaded regions indicate the empirical 2.5%–97.5% quantile range across seeds. We see that our ZOD approach converges faster than DPI in all the three quantities, and eventually achieves slightly smaller errors in terms of the value and gradient after sufficient numbers of iterations. However, the performance differs dramatically in terms of the Hessian accuracy. Over iterations, DPI barely reduces the error while ZOD effectively decreases it although the terminal error tends to be larger than its gradient and value counterparts.

Table 2 summarizes the corresponding mean final errors at the 10 value (Picard) iterations. The two methods produce comparable approximations of the value function and its gradient. ZOD achieves a smaller rRMSE, while the gradient errors of the two are similar. However, the final Hessian rRMSE of ZOD is 1.252×10^{-1} , whereas DPI gives 1.434, differing by more than 10 times. This discrepancy is a result of how the Hessian is obtained. In DPI, the Hessian is not learned directly; rather it is obtained by applying automatic differentiation twice on the learned value network. Since the training loss does not explicitly control the second-order derivative, the resulting Hessian need not be accurate even when the value function and its gradient are reasonably well approximated. By contrast, the proposed ZOD method augments the learning procedure with a dedicated Hessian network and a second-order zeroth-order objective. This second-order training signal directly targets the Hessian, leading to a substantially more accurate approximation. Finally, it is notable that ZOD spends about 30% less time obtaining these results.

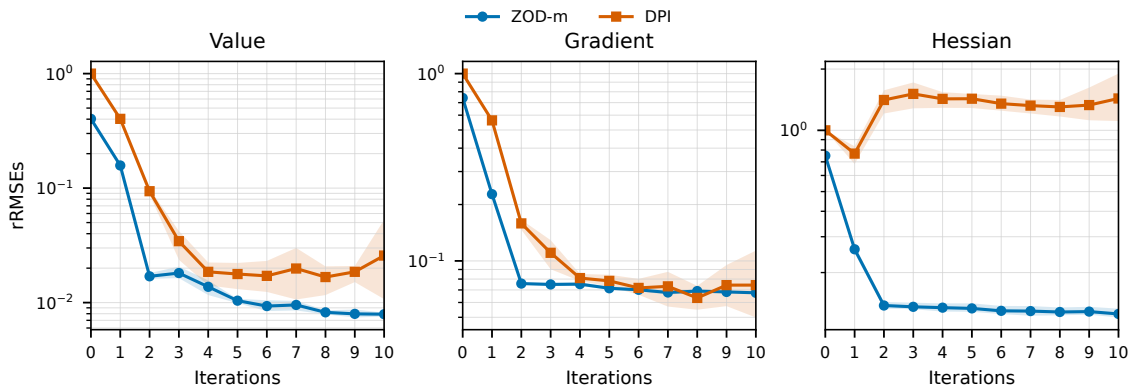


Figure 2: **rRMSE of the learned value, gradient, and Hessian over iterations** for the PDE (6.2) with dimension $d = 20$. ZOD-m and DPI use the same network (6.5) for value function.

Table 2: **rRMSEs for the semilinear PDE (6.2)**. The results are based on 10 value (Picard) iterations.

Method	Value Error	Gradient Error	Hessian Error	Time (s)
DPI	2.576×10^{-2}	7.429×10^{-2}	1.434×10^0	1301
ZOD-m	7.921×10^{-3}	6.757×10^{-2}	1.252×10^{-1}	1025

6.3 A Fully Nonlinear PDE

We next consider a fully nonlinear PDE studied in Han et al. (2026):

$$\partial_t u(t, x) + \frac{1}{2} \Delta u(t, x) + \frac{1}{4} \sum_{i=1}^d \left| \frac{\partial^2 u}{\partial x_i^2}(t, x) \right| - h(t, x) = 0, \quad (t, x) \in [0, T] \times \mathbb{R}^d. \quad (6.6)$$

The form of this PDE is reverse-engineered from the exact solution

$$u^*(t, x) = \sum_{j=1}^J v_j \sin \left(t + \sum_{i=1}^d w_i^j x_i \right), \quad (6.7)$$

with

$$h(t, x) = \partial_t u^*(t, x) + \frac{1}{2} \Delta u^*(t, x) + \frac{1}{4} \sum_{i=1}^d \left| \frac{\partial^2 u^*}{\partial x_i^2}(t, x) \right|.$$

The PDE (6.6) has differential operator $\mathcal{L}u = \frac{1}{2} \Delta u$, and the source term $f(t, x, \nabla^2 u) = \frac{1}{4} \sum_{i=1}^d \left| \frac{\partial^2 u}{\partial x_i^2}(t, x) \right| - h(t, x)$. Hence, it corresponds to the SDE $dX_t = dW_t$.

We use $d = 20$, $T = 1.0$, $J = 2$ and set $X_0 = 0$. The parameters in (6.7) are sampled once and then fixed throughout the experiment:

$$w_i^j \sim d^{-1/2} \mathcal{N}(0, 1), \quad v_j \sim \mathcal{N}(0, 1).$$

For the DPI baseline, we use a configuration with data size 4096, integral sample size 1024, three hidden layers of width 128, 16 epochs per iteration, and in total 40 iterations.

For ZOD, we use the three-network model with 3 hidden layers of width 64, ELU activations, ZOD bandwidth $\epsilon = 0.05$. The derivative networks are pretrained for 5000 steps. To ensure a fair comparison, we control the computational time of ZOD to be comparable to that of DPI. To this end, in our implementation of ZOD, we use batch size 32768 and 4096 gradient steps per value iteration.

Figure 3 compares the time-matched ZOD run with the 40-iteration DPI run. The rapid decay of the ZOD errors in the first few value (Picard) iterations is consistent with the contraction estimate in Theorem 6. After a small number of Picard iterations, the curves approach stable accuracy levels. This behavior agrees with the structure of (5.3): once the one-step learning error has been reduced to a fixed floor, the geometrically weighted accumulation term limits the additional gain from further iterations. Meanwhile, DPI decreases more gradually under the reported configuration. In particular, directly learning the Hessian through a second-order ZOD target gives a substantially lower second-order error than differentiating a learned scalar value approximation. Table 3 further reports the final errors and runtimes. ZOD is more accurate than DPI for all the three quantities, and the gap is the largest for the Hessian.

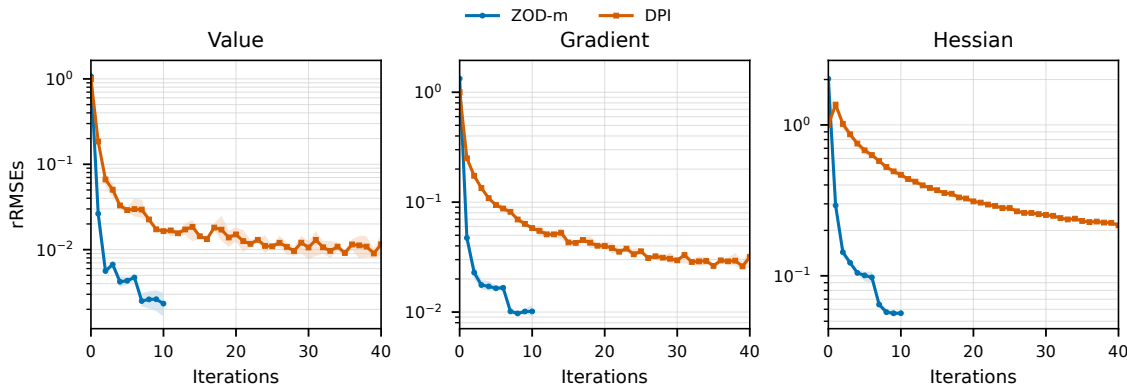


Figure 3: **rRMSE of the learned value, gradient, and Hessian over iterations** for the fully nonlinear PDE (6.6) with $d = 20$. The panels show value, gradient, and Hessian rRMSEs over iterations for the time-matched ZOD configuration and DPI. Curves are averaged over three random seeds, and shaded regions indicate empirical 2.5%–97.5% quantile bands.

Table 3: **rRMSE for the fully nonlinear PDE (6.6)**. The results of ZOD are based on 10 value iterations, while DPI is based on 40 iterations.

Method	Value Error	Gradient Error	Hessian Error	Time (s)
DPI	1.155×10^{-2}	3.167×10^{-2}	2.160×10^{-1}	1553
ZOD-m	2.333×10^{-3}	1.015×10^{-2}	5.649×10^{-2}	1886

Finally, we examine two controlled ablations for our ZOD approach. Table 4 summarizes the impact of training budget, batch size, and availability of the strong simulator. In the training-budget panel, we vary the number of gradient steps while fixing the batch size at 32768 and the bandwidth at $\epsilon = 0.05$. In the batch-size panel, we vary the batch size while fixing the training budget at 4096 steps and the bandwidth at $\epsilon = 0.05$. These two panels demonstrate the expected monotone improvement as the training budget and batch size increase, consistent with the theoretical decomposition in Theorem 9. Specifically, increasing the number of gradient steps reduces the optimization error in training the empirical loss (although we did not include the optimization error in Theorem 9). On the other hand, the batch-size ablation aligns directly with the theorem: a larger batch size decreases the statistical error term in the one-step bound, and therefore is expected to reduce the overall error. These results are also illustrated in Figure 4.

The last panel of Table 4 shows the effect of the strong simulator availability when everything else is the same. As predicted by the theoretical results, access to the strong simulator boosts performance. In Appendix B, we further examine the performance of ZOD when only the weak simulator is available. Overall, ZOD-1 is slightly worse than ZOD-m, and a small bandwidth is not always preferred with the former.

Table 4: **ZOD ablations for the fully nonlinear PDE** (6.6). Each row reports the mean final rRMSE over three random seeds.

Setting	Gradient Steps	Batch Size	Bandwidth ϵ	Strong Simulator	Value Error	Gradient Error	Hessian Error	Time (s)
Varying training steps	2048	32768	0.05	✓	3.601×10^{-3}	1.336×10^{-2}	7.736×10^{-2}	1006
	4096	32768	0.05	✓	2.333×10^{-3}	1.015×10^{-2}	5.649×10^{-2}	1886
	8192	32768	0.05	✓	2.032×10^{-3}	7.518×10^{-3}	4.083×10^{-2}	3665
Varying batch size	4096	16384	0.05	✓	3.016×10^{-3}	1.140×10^{-2}	6.713×10^{-2}	1061
	4096	32768	0.05	✓	2.333×10^{-3}	1.015×10^{-2}	5.649×10^{-2}	1886
	4096	65536	0.05	✓	1.997×10^{-3}	7.607×10^{-3}	4.575×10^{-2}	3636
Strong simulator availability	2048	65536	0.25	✗	6.398×10^{-3}	2.781×10^{-2}	6.192×10^{-1}	1330
	2048	65536	0.25	✓	3.687×10^{-3}	1.205×10^{-2}	5.178×10^{-2}	1819

7 Concluding Remarks

In this paper we develop a data-driven learning method for solving a broad class of potentially high-dimensional, fully nonlinear black-box PDEs. The thrust of our analysis is to represent the derivatives of the solutions via ZODs and then learn them in the same way as learning the solutions, rather than outright differentiating the learned solutions. Thus, the learning ultimately corresponds to policy evaluation (Jia and Zhou, 2022b) in continuous-time reinforcement learning (RL). Indeed, we solve the nonlinear PDEs through iterations each of which is a linear PDE and hence a policy evaluation problem. This idea

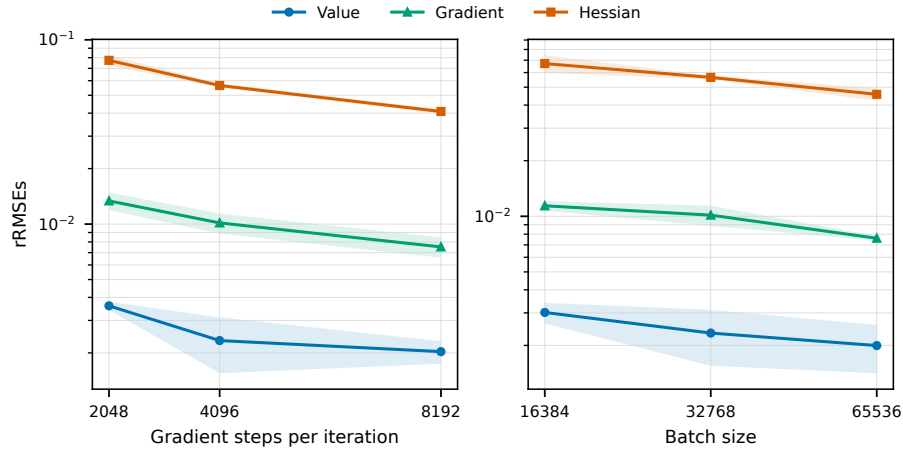


Figure 4: **Ablations for the fully nonlinear PDE (6.6)**. All curves report the mean rRMSE over 3 seeds, and shaded regions indicate one empirical standard deviation.

further inspires us to formulate learning the gradient and Hessian also as policy evaluation problems.¹⁴

The approach developed in this paper hints on an important implication in the study of data-driven methodologies in machine learning: the design of algorithms should be tailored to the availability of the data-generating mechanisms or the properties of simulators. With strong simulators similar to the one introduced in this paper, various variance reduction tricks in simulations may be employed to improve the efficiency of the algorithms. This opens a gate to further studying its potential in other tasks.

Acknowledgments and Disclosure of Funding

Huÿen Pham is supported by the Société Générale Chair “Risques Financiers”, FiME (Laboratory of Finance and Energy Markets), and the EDF–CACIB Chair “Finance and Sustainable Development”. Yanwei Jia is supported by the University Start-up Fund at The Chinese University of Hong Kong. Xun Yu Zhou is supported by the Nie Center for Intelligent Wealth Management at Columbia. The authors thank Xuefeng Gao, Chun Liu, and Jiale Zha for discussions on the contents of this paper.

14. It should be noted that although the PDEs studied in this paper include *in form* the class of HJB equations as a special case, our assumption that the value of the nonlinear source term is known for each given input excludes the applicability to HJB in the model-free RL setting. However, Jia and Zhou (2022a, 2023) decompose the RL into policy evaluation and policy improvement/q-learning, and show that the latter can be reformulated as policy evaluation. In this sense, black-box HJB equations can also be solved essentially by policy evaluation, albeit in a very different way from this paper.

Appendix A. A Sufficient Condition for Assumption 2

We provide a concrete sufficient condition for Assumption 2.

Consider the operator $\mathcal{L}v := \Delta v - \nabla h \cdot \nabla v$ that corresponds to the Langevin SDE:

$$dX_t = -\nabla h(X_t)dt + \sqrt{2}dW_t.$$

Consider the norm defined in Section 5 with a weighting parameter β to be specified later:

$$\|v\|_{\beta, \text{mix}}^2 := \beta\|v\|_{\beta}^2 + \beta\|\nabla v\|_{\beta}^2 + \|\nabla^2 v\|_{\beta}^2.$$

For a triplet $\mathbf{U} = (V, G, H)$, we denote the corresponding norm as

$$\|\mathbf{U}\|_{\beta, \text{mix}}^2 := \beta\|V\|_{\beta}^2 + \beta\|G\|_{\beta}^2 + \|H\|_{\beta}^2,$$

and denote the solution mapping as

$$\mathbf{U} \mapsto v = \mathcal{S}(\mathbf{U}), \text{ such that } \partial_t v + \mathcal{L}v + f(x, V, G, H) = 0, \quad v(T, x) = g(x).$$

For any two input triplets $\mathbf{U}_i = (V_i, G_i, H_i)$, $i = 1, 2$. Denote $\delta f := f(x, \mathbf{U}_1) - f(x, \mathbf{U}_2)$, $v := \mathcal{S}(\mathbf{U}_1) - \mathcal{S}(\mathbf{U}_2)$. Then v satisfies the linear PDE

$$\partial_t v + \mathcal{L}v = -\delta f, \quad v(T, x) = 0. \tag{A.1}$$

We aim to prove the contraction property through the bound

$$\|v\|_{\beta, \text{mix}} \leq \gamma \|\mathbf{U}_1 - \mathbf{U}_2\|_{\beta, \text{mix}}$$

for some constant $\gamma < 1$. We need the following assumption in this section.

Assumption 5 (i) $h \in C_p^2(\mathbb{R}^d)$, the eigenvalues of $\nabla^2 h$ have a uniform lower bound $-k$ for some $k \geq 0$, and

$$\lim_{|x| \rightarrow \infty} \frac{h(x)}{\log |x|} = \infty. \tag{A.2}$$

(ii) The nonlinearity f is Lipschitz in the value, gradient, and Hessian variables:

$$|f(x, V, G, H) - f(x, V', G', H')| \leq L_v|V - V'| + L_g|G - G'| + L_h|H - H'|.$$

(iii) $v := \mathcal{S}(\mathbf{U}_1) - \mathcal{S}(\mathbf{U}_2) \in C_p^{1,3}([0, T] \times \mathbb{R}^d)$.

Under Assumption 5 (i), the potential h is a smooth confining potential, and the Langevin dynamics admits the invariant density, see, e.g., (Pavliotis, 2014, Proposition 4.2),

$$p(x) := \frac{1}{C} \exp\{-h(x)\},$$

where $C := \int_{\mathbb{R}^d} \exp\{-h(x)\}dx < \infty$.

From now on, we take $X_0 \sim p(\cdot)$, and then $X_t \sim p(\cdot)$ for all $t \geq 0$. Consequently, the norms have the following explicit form:

$$\|v\|_{L_2(p)}^2 := \int_{\mathbb{R}^d} |v(t, x)|^2 p(x) dx, \quad \|v\|_{\beta}^2 := \int_0^T e^{\beta t} \int_{\mathbb{R}^d} |v(t, x)|^2 p(x) dx dt.$$

The following theorem provides a sufficient condition for the contraction property used in the main text.

Theorem 11 *Suppose Assumption 5 holds. If $12L_h^2 < 1$, then there exists a sufficiently large $\beta \geq k$ such that*

$$\eta_\beta := 12 \left((L_v^2 + L_g^2) \frac{\beta + 1}{\beta^2} + L_h^2 \frac{\beta + 1}{\beta} \right) < 1, \quad (\text{A.3})$$

and set $\gamma = \sqrt{\eta_\beta}$. Then

$$\|\mathcal{S}(\mathbf{U}_1) - \mathcal{S}(\mathbf{U}_2)\|_{\beta, \text{mix}} \leq \gamma \|\mathbf{U}_1 - \mathbf{U}_2\|_{\beta, \text{mix}}.$$

That is, Assumption 2 holds.

The rest of this appendix is devoted to proving Theorem 11. We first analyze the norm of the value and the gradient. The following lemma holds in a general diffusion setting, where \mathcal{L} is the infinitesimal generator of the diffusion, $\mathcal{L}v = b \cdot \nabla v + \frac{1}{2} \text{Tr}(\sigma \sigma^\top \nabla^2 v)$.

Lemma 12 *Suppose Assumption 5 holds. Assume that $\sigma \sigma^\top$ is uniformly elliptic, i.e., there is a constant $\lambda > 0$, such that $x^\top \sigma \sigma^\top x \geq \lambda |x|^2$. Then the value and gradient have the following upper bound:*

$$\frac{\beta}{2} \|v\|_\beta^2 + \lambda \|\nabla v\|_\beta^2 \leq \frac{2}{\beta} \|\delta f\|_\beta^2. \quad (\text{A.4})$$

Proof Denote $Y_t := e^{\beta t} v(t, X_t)^2$. It follows from Ito's formula that

$$dY_t = \beta e^{\beta t} v^2 dt + 2e^{\beta t} v (\partial_t + \mathcal{L})v dt + e^{\beta t} \nabla v^\top \sigma \sigma^\top \nabla v dt + dM_t,$$

where M is a martingale. Taking expectation on both sides and noting $Y_T = 0$, $(\partial_t + \mathcal{L})v = -\delta f$, we have

$$-\mathbb{E}[Y_0] = \mathbb{E} \left[\int_0^T \beta e^{\beta t} v^2 - 2e^{\beta t} v \delta f + e^{\beta t} \nabla v^\top \sigma \sigma^\top \nabla v dt \right].$$

Thus,

$$\beta \|v\|_\beta^2 + \mathbb{E} \left[\int_0^T e^{\beta t} \nabla v^\top \sigma \sigma^\top \nabla v dt \right] = \mathbb{E} \left[\int_0^T 2e^{\beta t} v \delta f dt \right] - \mathbb{E}[Y_0].$$

It follows from the uniform ellipticity of $\sigma \sigma^\top$ that $\nabla v^\top \sigma \sigma^\top \nabla v \geq \lambda \|\nabla v\|^2$. This along with the inequality $2|v \delta f| \leq \frac{\beta}{2} |v|^2 + \frac{2}{\beta} |\delta f|^2$ yields

$$\beta \|v\|_\beta^2 + \lambda \|\nabla v\|_\beta^2 \leq \frac{\beta}{2} \|v\|_\beta^2 + \frac{2}{\beta} \|\delta f\|_\beta^2 - \mathbb{E}[Y_0].$$

However, $Y_0 \geq 0$; thus

$$\frac{\beta}{2} \|v\|_\beta^2 + \lambda \|\nabla v\|_\beta^2 \leq \frac{2}{\beta} \|\delta f\|_\beta^2.$$

The desired result then follows. ■

The preceding estimate controls the value component and provides an initial control on the gradient. To handle the Hessian component, we use the reversibility of the Langevin generator with respect to the stationary density.

Lemma 13 *Suppose Assumption 5 (i) holds. Let $r \in C_p^2(\mathbb{R}^d)$ and $g \in C_p^1(\mathbb{R}^d)$. Then*

$$\int_{\mathbb{R}^d} (\mathcal{L}r)g(x)p(x)dx = - \int_{\mathbb{R}^d} \nabla r \cdot \nabla gp(x)dx.$$

Proof Note that $\nabla p = \frac{1}{C} \exp\{-h(x)\} (-\nabla h) = -p\nabla h$. Thus,

$$\nabla p + p\nabla h = 0. \quad (\text{A.5})$$

It follows from the integration by parts (the remainder terms vanish due to (A.2)) that

$$\begin{aligned} \int_{\mathbb{R}^d} (\mathcal{L}r)g(x)p(x)dx &= \int_{\mathbb{R}^d} (\Delta r - \nabla h \cdot \nabla r)g(x)p(x)dx \\ &= - \int_{\mathbb{R}^d} \nabla r \cdot \nabla(gp)dx - \int_{\mathbb{R}^d} \nabla h \cdot \nabla r g p dx \\ &= - \int_{\mathbb{R}^d} \nabla r \cdot \nabla gp dx - \int_{\mathbb{R}^d} \nabla r \cdot (\nabla p + p\nabla h)g dx \\ &= - \int_{\mathbb{R}^d} \nabla r \cdot \nabla gp dx. \end{aligned}$$

where the last equality follows from the identity (A.5). ■

Next, we turn to the Hessian bound. The following lemma represents the norm of the Hessian $\nabla^2 v$.

Lemma 14 *Under Assumption 5, for $\mathcal{L}v = \Delta v - \nabla h \cdot \nabla v$, we have*

$$\frac{1}{2} \mathcal{L}(|\nabla v|^2) - \nabla v \cdot \nabla(\mathcal{L}v) = |\nabla^2 v|^2 + \nabla v^\top \nabla^2 h \nabla v. \quad (\text{A.6})$$

Proof Note that

$$\begin{aligned} \frac{1}{2} \Delta(|\nabla v|^2) &= \frac{1}{2} \sum_{i=1}^d \partial_i^2 \sum_{j=1}^d (\partial_j v)^2 = \frac{1}{2} \sum_{i,j=1}^d \partial_i (2\partial_j v \partial_{ij}^2 v) = \sum_{i,j=1}^d (\partial_{ij}^2 v \partial_{ij}^2 v + \partial_j v \partial_i (\partial_{ij}^2 v)) \\ &= |\nabla^2 v|^2 + \nabla v \cdot \nabla(\Delta v). \end{aligned}$$

Thus,

$$\begin{aligned} \frac{1}{2} \mathcal{L}(|\nabla v|^2) - \nabla v \cdot \nabla(\mathcal{L}v) &= \frac{1}{2} \Delta(|\nabla v|^2) - \frac{1}{2} \nabla h \cdot \nabla(|\nabla v|^2) - \nabla v \cdot \nabla(\Delta v - \nabla h \cdot \nabla v) \\ &= \frac{1}{2} \Delta(|\nabla v|^2) - \nabla v \cdot \nabla(\Delta v) - \frac{1}{2} \nabla h \cdot \nabla(|\nabla v|^2) + \nabla v \cdot \nabla(\nabla h \cdot \nabla v) \\ &= |\nabla^2 v|^2 - \frac{1}{2} \nabla h \cdot \nabla(|\nabla v|^2) + \nabla v \cdot \nabla(\nabla h \cdot \nabla v). \end{aligned} \quad (\text{A.7})$$

We now compute the second and third terms, respectively. Indeed,

$$-\frac{1}{2} \nabla h \cdot \nabla(|\nabla v|^2) = -\frac{1}{2} \sum_{i=1}^d \partial_i h \partial_i \left(\sum_{j=1}^d (\partial_j v)^2 \right) = - \sum_{i,j=1}^d \partial_i h \partial_j v \partial_{ij}^2 v, \quad (\text{A.8})$$

while

$$\begin{aligned}
 \nabla v \cdot \nabla(\nabla h \cdot \nabla v) &= \sum_{i=1}^d \partial_i v \partial_i \left(\sum_{j=1}^d \partial_j h \partial_j v \right) = \sum_{i,j=1}^d \partial_i v \partial_{ij}^2 h \partial_j v + \sum_{i,j=1}^d \partial_i h \partial_j v \partial_{ij}^2 v \\
 &= \nabla v^\top \nabla^2 h \nabla v + \sum_{i,j=1}^d \partial_i h \partial_j v \partial_{ij}^2 v.
 \end{aligned} \tag{A.9}$$

The desired result follows from (A.7), (A.8) and (A.9). \blacksquare

Integrating the above identity against the invariant density converts the local Hessian identity into an $L_2(p)$ estimate. The integration by parts formula above removes the total-divergence term.

Lemma 15 *Under Assumption 5, we have*

$$\|\nabla^2 v\|_\beta^2 = \|\mathcal{L}v\|_\beta^2 - \int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^\top \nabla^2 h \nabla v p dx dt.$$

Proof It follows from the formula (A.6) that

$$\int_{\mathbb{R}^d} \frac{1}{2} \mathcal{L}(|\nabla v|^2) p dx - \int_{\mathbb{R}^d} \nabla v \cdot \nabla(\mathcal{L}v) p dx = \int_{\mathbb{R}^d} |\nabla^2 v|^2 p dx + \int_{\mathbb{R}^d} \nabla v^\top \nabla^2 h \nabla v p dx. \tag{A.10}$$

Lemma 13 yields that the first term on the left hand side of the above equation equals 0:

$$\int_{\mathbb{R}^d} \frac{1}{2} \mathcal{L}(|\nabla v|^2) p dx = - \int_{\mathbb{R}^d} \nabla(|\nabla v|^2) \cdot \nabla \left(\frac{1}{2} \right) p dx = 0.$$

The second term can be written as

$$- \int_{\mathbb{R}^d} \nabla v \cdot \nabla(\mathcal{L}v) p dx = \int_{\mathbb{R}^d} (\mathcal{L}v)^2 p dx.$$

Therefore, integrating (A.10) over $[0, T]$ with the weight $e^{\beta t}$, we have

$$\|\nabla^2 v\|_\beta^2 = \|\mathcal{L}v\|_\beta^2 - \int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^\top \nabla^2 h \nabla v p dx dt.$$

The desired result follows. \blacksquare

Lemma 15 gives the unfinished bound of Hessian $\nabla^2 v$, which contains the norm of $\mathcal{L}v$. To bound $\mathcal{L}v$, we have to analyze $\partial_t v$.

Lemma 16 *Under Assumption 5, we have*

$$\|\partial_t v\|_\beta^2 + \beta \|\nabla v\|_\beta^2 \leq \|\delta f\|_\beta^2.$$

Proof Multiply (A.1) by $\partial_t v$ and integrate over \mathbb{R}^d with weight p to get

$$\int_{\mathbb{R}^d} (\partial_t v)^2 p dx + \int_{\mathbb{R}^d} \mathcal{L}v \partial_t v p dx = \int_{\mathbb{R}^d} -\delta f \partial_t v p dx. \quad (\text{A.11})$$

For the second term on the left hand side of the above equation, by Lemma 13,

$$\int_{\mathbb{R}^d} \mathcal{L}v \partial_t v p dx = - \int_{\mathbb{R}^d} \nabla v \cdot \nabla (\partial_t v) p dx = -\frac{1}{2} \frac{d}{dt} \int_{\mathbb{R}^d} |\nabla v|^2 p dx.$$

Integrating over $[0, T]$ with weight $e^{\beta t}$ and using integration by parts with respect to t , we obtain

$$\begin{aligned} \int_0^T e^{\beta t} - \frac{1}{2} \frac{d}{dt} \int_{\mathbb{R}^d} |\nabla v|^2 p dx dt &= -\frac{1}{2} \int_0^T \frac{d}{dt} \left(e^{\beta t} \int_{\mathbb{R}^d} |\nabla v|^2 p dx \right) dt \\ &\quad + \frac{\beta}{2} \int_0^T e^{\beta t} \int_{\mathbb{R}^d} |\nabla v|^2 p dx dt \\ &= \frac{1}{2} \int_{\mathbb{R}^d} |\nabla v(0, \cdot)|^2 p dx + \frac{\beta}{2} \|\nabla v\|_{\beta}^2 \geq \frac{\beta}{2} \|\nabla v\|_{\beta}^2, \end{aligned}$$

where in the second equality, we used the fact that $v(T, x) = 0$. This along with (A.11) yields

$$\|\partial_t v\|_{\beta}^2 + \frac{\beta}{2} \|\nabla v\|_{\beta}^2 \leq \int_0^T e^{\beta t} \int_{\mathbb{R}^d} |\delta f \partial_t v| p dx dt \leq \frac{1}{2} \|\delta f\|_{\beta}^2 + \frac{1}{2} \|\partial_t v\|_{\beta}^2.$$

This proves the desired result. ■

Combining these lemmas, we have the upper bound for the norm of Hessian $\nabla^2 v$.

Theorem 17 *Under Assumption 5,*

$$\|\nabla^2 v\|_{\beta}^2 \leq 4\|\delta f\|_{\beta}^2 - (2\beta - k)\|\nabla v\|_{\beta}^2.$$

Proof It follows from Lemma 15 and Lemma 16 that

$$\begin{aligned} \|\nabla^2 v\|_{\beta}^2 &= \|\mathcal{L}v\|_{\beta}^2 - \int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^{\top} \nabla^2 h \nabla v p dx dt \\ &\leq 2\|\partial_t v\|_{\beta}^2 + 2\|\delta f\|_{\beta}^2 - \int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^{\top} \nabla^2 h \nabla v p dx dt \\ &\leq 4\|\delta f\|_{\beta}^2 - 2\beta\|\nabla v\|_{\beta}^2 - \int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^{\top} \nabla^2 h \nabla v p dx dt. \end{aligned}$$

Since the eigenvalues of $\nabla^2 h$ have a uniform lower bound $-k$, $\nabla^2 h + kI$ is positive semi-definite. Therefore,

$$\int_0^T e^{\beta t} \int_{\mathbb{R}^d} \nabla v^{\top} \nabla^2 h \nabla v p dx dt \geq -k\|\nabla v\|_{\beta}^2,$$

which yields

$$\|\nabla^2 v\|_{\beta}^2 \leq 4\|\delta f\|_{\beta}^2 - (2\beta - k)\|\nabla v\|_{\beta}^2.$$

This proves the desired result. ■

We now finish the proof of the sufficient condition by relating δf to the distance between the two input triplets.

Proof [Proof of Theorem 11] For the Langevin dynamics considered here, (A.4) holds with $\lambda = 2$. Hence

$$\beta \|v\|_\beta^2 \leq \frac{4}{\beta} \|\delta f\|_\beta^2, \quad \beta \|\nabla v\|_\beta^2 \leq \|\delta f\|_\beta^2.$$

Together with Theorem 17, this gives

$$\begin{aligned} \|v\|_{\beta, \text{mix}}^2 &= \beta \|v\|_\beta^2 + \beta \|\nabla v\|_\beta^2 + \|\nabla^2 v\|_\beta^2 \\ &\leq 4 \|\delta f\|_\beta^2 - (2\beta - k) \|\nabla v\|_\beta^2 + \beta \|\nabla v\|_\beta^2 + \frac{4}{\beta} \|\delta f\|_\beta^2 \\ &\leq 4 \left(1 + \frac{1}{\beta}\right) 3 (L_v^2 \|V_1 - V_2\|_\beta^2 + L_g^2 \|G_1 - G_2\|_\beta^2 + L_h^2 \|H_1 - H_2\|_\beta^2) - (\beta - k) \|\nabla v\|_\beta^2 \\ &\leq 12 \left((L_v^2 + L_g^2) \frac{\beta + 1}{\beta^2} + L_h^2 \frac{\beta + 1}{\beta} \right) \|\mathbf{U}_1 - \mathbf{U}_2\|_{\beta, \text{mix}}^2 - (\beta - k) \|\nabla v\|_\beta^2. \end{aligned}$$

For any $\beta \geq k$, the last term is nonpositive, and therefore

$$\|v\|_{\beta, \text{mix}}^2 \leq \eta_\beta \|\mathbf{U}_1 - \mathbf{U}_2\|_{\beta, \text{mix}}^2,$$

where η_β is defined in (A.3). If $\eta_\beta < 1$, taking square roots gives the contraction inequality with $\gamma = \sqrt{\eta_\beta} < 1$. Finally, if $12L_h^2 < 1$, then $\eta_\beta \rightarrow 12L_h^2$ as $\beta \rightarrow \infty$, so such a choice of β is available. ■

Appendix B. Addition Numerical Results with Weak Simulator

We examine the performance of our approach when only the weak simulator is available on the fully nonlinear PDE example in Subsection 6.3, as a robustness check. The results are summarized in Table 5. Overall, consistent with our theoretical results, the performance of ZOD-1 is slightly worse than that with ZOD-m. Indeed, in this case it is no longer true that the smaller bandwidth the better performance.

Table 5: **Summary of ZOD ablations for the fully nonlinear PDE** (6.6). Each row reports the mean final rRMSE over three random seeds.

Setting	Gradient Steps	Batch Size	Bandwidth ϵ	Strong Simulator	Value Error	Gradient Error	Hessian Error	Time (s)
Varying bandwidth	2048	65536	0.1	✗	3.854×10^{-2}	4.571×10^{-2}	2.320×10^0	1328
	2048	65536	0.2	✗	9.834×10^{-3}	3.168×10^{-2}	9.591×10^{-1}	1326
	2048	65536	0.25	✗	6.398×10^{-3}	2.781×10^{-2}	6.192×10^{-1}	1330
	2048	65536	0.5	✗	4.255×10^{-3}	3.697×10^{-2}	2.703×10^{-1}	1327
Varying batch size	2048	65536	0.25	✗	6.398×10^{-3}	2.781×10^{-2}	6.192×10^{-1}	1330
	2048	131072	0.25	✗	4.896×10^{-3}	2.481×10^{-2}	4.939×10^{-1}	2512
	2048	262144	0.25	✗	5.000×10^{-3}	1.982×10^{-2}	3.955×10^{-1}	4880

Appendix C. Proof of Statements

C.1 Proof of Lemma 1

Proof We prove the $d = 1$ case for illustration. It follows from Taylor's expansion that

$$\frac{Z}{\epsilon}u(t, x + \epsilon Z) = \frac{Z}{\epsilon} \left(u(t, x) + \nabla u(t, x)\epsilon Z + \frac{1}{2}\nabla^2 u(t, x)\epsilon^2 Z^2 + c(t, x, x + \epsilon Z)\epsilon^3 Z^3 \right). \quad (\text{C.1})$$

Thus,

$$\begin{aligned} \left| \mathbb{E} \left[\frac{Z}{\epsilon}u(t, x + \epsilon Z) \right] - \nabla u(t, x) \right| &= \left| \mathbb{E} \left[\frac{1}{2}\nabla^2 u(t, x)\epsilon Z^3 + c(t, x, x + \epsilon Z)\epsilon^2 Z^4 \right] \right| \\ &= \left| \mathbb{E}[c(t, x, x + \epsilon Z)Z^4] \right| \epsilon^2 \leq C\epsilon^2. \end{aligned}$$

Similarly, if u has the fourth-order Taylor expansion, then

$$\begin{aligned} &\frac{ZZ^\top - I_d}{\epsilon^2}u(t, x + \epsilon Z) \\ &= \frac{Z^2 - 1}{\epsilon^2}u(t, x) + \frac{Z^3 - Z}{\epsilon}\nabla u(t, x) + \frac{1}{2}(Z^4 - Z^2)\nabla^2 u(t, x) \\ &\quad + \frac{1}{6}\epsilon\nabla^3 u(t, x)(Z^5 - Z^3) + c'(t, x, x + \epsilon Z)(Z^6 - Z^4)\epsilon^2. \end{aligned} \quad (\text{C.2})$$

Consequently,

$$\left| \mathbb{E} \left[\frac{ZZ^\top - I_d}{\epsilon^2}u(t, x + \epsilon Z) \right] - \nabla^2 u(t, x) \right| = \left| \mathbb{E}[c'(t, x, x + \epsilon Z)(Z^6 - Z^4)] \right| \epsilon^2 \leq C\epsilon^2.$$

The variance parts follow from (C.1), (C.2) and the fact that c, c' have polynomial growth. ■

C.2 Proof of Lemma 2

Proof For notational simplicity, we suppress the variable t . The Taylor expansions of $u(x \pm \epsilon Z)$ are:

$$\begin{aligned} u(x + \epsilon Z) &= u(x) + \epsilon\nabla u^\top Z + \frac{\epsilon^2}{2}Z^\top\nabla^2 u(x)Z + O(\epsilon^3), \\ u(x - \epsilon Z) &= u(x) - \epsilon\nabla u^\top Z + \frac{\epsilon^2}{2}Z^\top\nabla^2 u(x)Z + O(\epsilon^3). \end{aligned}$$

Subtracting the second expansion from the first cancels all even-order terms:

$$u(x + \epsilon Z) - u(x - \epsilon Z) = 2\epsilon\nabla u^\top Z + O(\epsilon^3).$$

Substituting this into the estimator definition in Eq. (3.1):

$$\hat{g}(x; \epsilon) = \frac{2\epsilon\nabla u^\top Z + O(\epsilon^3)}{2\epsilon}Z = (\nabla u^\top Z)Z + O(\epsilon^2).$$

The expectation is $\mathbb{E}[(\nabla u^\top Z)Z] = \nabla u$, since $\mathbb{E}[ZZ^\top] = I_d$. The bias is therefore dominated by the $O(\epsilon^2)$ term.

For the variance, observe that the leading term of the estimator, $(\nabla u^\top Z)Z$, is independent of ϵ . Its variance is a constant determined by the moments of Z and the magnitude of ∇u :

$$\lim_{\epsilon \rightarrow 0} \text{Var}[\hat{g}(x; \epsilon)] = \text{Var}[(\nabla u^\top Z)Z].$$

This variance is finite and does not depend on ϵ . Thus, the total variance is bounded. \blacksquare

C.3 Proof of Lemma 3

Proof We suppress the variable t . Using the Taylor expansions up to the fourth order, the central difference numerator becomes:

$$u(x + \epsilon Z) - 2u(x) + u(x - \epsilon Z) = \epsilon^2 Z^\top \nabla^2 u Z + \frac{\epsilon^4}{12} \sum_{1 \leq i, j, k, l \leq d} \frac{\partial^4 u(x)}{\partial x_i \partial x_j \partial x_k \partial x_l} Z_i Z_j Z_k Z_l + o(\epsilon^4).$$

The symmetric construction cancels all the odd-order terms. Substituting this into Eq. (3.2) yields:

$$\hat{H}(x; \epsilon) = \left(\frac{1}{2} Z^\top \nabla^2 u Z + \frac{\epsilon^2}{24} \sum_{1 \leq i, j, k, l \leq d} \frac{\partial^4 u(x)}{\partial x_i \partial x_j \partial x_k \partial x_l} Z_i Z_j Z_k Z_l + o(\epsilon^2) \right) (ZZ^\top - I_d).$$

Thus, the variance is bounded when ϵ tends to 0. For the bias part, we take the expectation of the leading term. Using the property of Gaussian moments that $\mathbb{E}[(Z^\top \nabla^2 u Z)(ZZ^\top - I_d)] = 2H$ for a symmetric matrix $\nabla^2 u$ (derived from Isserlis' theorem), we find

$$\begin{aligned} \mathbb{E}[\hat{H}(x; \epsilon)] &= \frac{1}{2} \mathbb{E} \left[(Z^\top \nabla^2 u Z)(ZZ^\top - I_d) \right] + O(\epsilon^2) \\ &= \frac{1}{2} (2H) + O(\epsilon^2) = \nabla^2 u + O(\epsilon^2). \end{aligned}$$

The bias is therefore of order $O(\epsilon^2)$. \blacksquare

C.4 Proof of Proposition 5

Proof The proof of the results for the weak simulator case is the same as the Proposition 4. We only prove the strong simulator case.

For simplicity, we use $R(t, x)$ to represent $R^{t,x}(\mathbf{U}_n)$. We only prove the variance result for \hat{h} ; the other parts are similar. Due to the regularity conditions of b and σ , it follows from Protter (2005, Theorem 40) and Kunita (2019, Theorem 3.4.2) that $X_s^{t,x}$ is four times continuously differentiable in x , and for every $p > 2$, there is a constant $C_p > 0$ such that for any multi-index \mathbf{i} with $0 < |\mathbf{i}| \leq 4$, it holds that

$$\sup_{x \in \mathbb{R}^d, 0 \leq t \leq T} \mathbb{E} \left[\sup_{t \leq s \leq T} |\partial_x^{\mathbf{i}} X_s^{t,x}|^p \right] \leq C_p. \quad (\text{C.3})$$

Therefore, $R(t, x)$ is also four times continuously differentiable in x and admits the Taylor expansion

$$\begin{aligned} R(t, x + \epsilon Z) &= R(t, x) + \epsilon \nabla R(t, x)^\top Z + \frac{\epsilon^2}{2} Z^\top \nabla^2 R(t, x) Z + \frac{\epsilon^3}{6} \nabla^3 R(t, x)(Z, Z, Z) \\ &\quad + \frac{\epsilon^4}{24} \nabla^4 R(t, x + \theta_1 \epsilon Z)(Z, Z, Z, Z), \\ R(t, x - \epsilon Z) &= R(t, x) - \epsilon \nabla R(t, x)^\top Z + \frac{\epsilon^2}{2} Z^\top \nabla^2 R(t, x) Z - \frac{\epsilon^3}{6} \nabla^3 R(t, x)(Z, Z, Z) \\ &\quad + \frac{\epsilon^4}{24} \nabla^4 R(t, x - \theta_2 \epsilon Z)(Z, Z, Z, Z), \end{aligned}$$

where $\theta_1, \theta_2 \in (0, 1)$ and

$$\begin{aligned} \nabla^3 R(t, x)(Z, Z, Z) &:= \sum_{1 \leq i, j, k \leq d} \frac{\partial_x^3 R(t, x)}{\partial x_i \partial x_j \partial x_k} Z_i Z_j Z_k, \\ \nabla^4 R(t, x)(Z, Z, Z, Z) &:= \sum_{1 \leq i, j, k, l \leq d} \frac{\partial_x^4 R(t, x)}{\partial x_i \partial x_j \partial x_k \partial x_l} Z_i Z_j Z_k Z_l. \end{aligned}$$

Denote $\hat{h}(t, x) := \frac{ZZ^\top - I_d}{2\epsilon^2} (R(t, x + \epsilon Z) + R(t, x - \epsilon Z) - 2R(t, x))$. Then

$$\begin{aligned} \hat{h}(t, x) &= \frac{ZZ^\top - I_d}{2} Z^\top \nabla^2 R(t, x) Z \\ &\quad + \frac{\epsilon^2}{48} (ZZ^\top - I_d) [\nabla^4 R(t, x + \theta_1 \epsilon Z) + \nabla^4 R(t, x - \theta_2 \epsilon Z)] (Z, Z, Z, Z). \end{aligned}$$

It follows from the regularity conditions of g, \tilde{f} and (C.3) that for every $p > 0$ there are constants $C_p > 0$ and $q > 0$ such that for all $t, x \in [0, T] \times \mathbb{R}^d$,

$$\mathbb{E} [|\nabla^2 R(t, x)|^p] \leq C_p (1 + |x|^q), \quad \mathbb{E} [|\nabla^4 R(t, x)|^p] \leq C (1 + |x|^q).$$

This along with the moments bound of X_t yields

$$\begin{aligned} \mathbb{E}[|\hat{h}|^2] &\leq 2\mathbb{E} \left[\left| \frac{ZZ^\top - I_d}{2} Z^\top \nabla^2 R(t, X_t) Z \right|^2 \right] \\ &\quad + 2\mathbb{E} \left[\left| \frac{\epsilon^2}{48} (ZZ^\top - I_d) [\nabla^4 R(t, X_t + \theta_1 \epsilon Z) + \nabla^4 R(t, X_t - \theta_2 \epsilon Z)] (Z, Z, Z, Z) \right|^2 \right] \\ &\leq \mathbb{E} \left[\left| ZZ^\top - I_d \right|^2 |\nabla^2 R(t, X_t)|^2 |Z|^4 \right] \\ &\quad + \frac{\epsilon^2}{24} \mathbb{E} \left[\left| ZZ^\top - I_d \right|^2 |\nabla^4 R(t, X_t + \theta_1 \epsilon Z) + \nabla^4 R(t, X_t - \theta_2 \epsilon Z)|^2 |Z|^8 \right] \\ &\leq C, \end{aligned}$$

for some constant $C > 0$. This proves the desired result. \blacksquare

C.5 Proof of Theorem 6

Proof By the triangle inequality, we decompose the total error at step n into the computational error introduced at the current step and the propagated error from the previous step:

$$\|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} \leq \|\mathbf{U}_n - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}} + \|\mathcal{S}(\mathbf{U}_{n-1}) - \mathbf{u}^*\|_{\beta, \text{mix}}.$$

The first term on the right-hand side is exactly the definition of the one-step error E_n . For the second term, using the fixed-point property $\mathbf{u}^* = \mathcal{S}(\mathbf{u}^*)$ and the Contraction Assumption 2, we have:

$$\|\mathcal{S}(\mathbf{U}_{n-1}) - \mathbf{u}^*\|_{\beta, \text{mix}} = \|\mathcal{S}(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{u}^*)\|_{\beta, \text{mix}} \leq \gamma \|\mathbf{U}_{n-1} - \mathbf{u}^*\|_{\beta, \text{mix}}.$$

Combining these inequalities yields the recursive relationship:

$$\|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} \leq \gamma \|\mathbf{U}_{n-1} - \mathbf{u}^*\|_{\beta, \text{mix}} + E_n.$$

Thus,

$$\begin{aligned} \|\mathbf{U}_n - \mathbf{u}^*\|_{\beta, \text{mix}} &\leq \gamma (\gamma \|\mathbf{U}_{n-2} - \mathbf{u}^*\|_{\beta, \text{mix}} + E_{n-1}) + E_n \\ &= \gamma^2 \|\mathbf{U}_{n-2} - \mathbf{u}^*\|_{\beta, \text{mix}} + \gamma E_{n-1} + E_n \\ &\leq \gamma^n \|\mathbf{U}_0 - \mathbf{u}^*\|_{\beta, \text{mix}} + \sum_{k=1}^n \gamma^{n-k} E_k. \end{aligned}$$

This proves the desired result. ■

C.6 Proof of Lemma 7

Proof We derive the result for the gradient component only; the others follow similarly. Let $Y_g = \frac{Z}{\epsilon} R^{t, X_t + \epsilon Z}$ denote the stochastic gradient target. The gradient loss is:

$$L_g(\varphi_g, \mathbf{U}_{n-1}) = \beta \mathbb{E} \left[\int_0^T e^{\beta t} |Y_g - \varphi_g(t, X_t)|^2 dt \right].$$

Note that

$$\mathbb{E} \left[|\varphi_g(t, X_t) - Y_g|^2 \right] = \mathbb{E} \left[\mathbb{E} \left[|\varphi_g(t, X_t) - G^\epsilon(t, X_t) + G^\epsilon(t, X_t) - Y_g|^2 \middle| t, X_t \right] \right].$$

Since $G^\epsilon(t, X_t) = \mathbb{E}[Y_g | t, X_t]$, the cross-term in the expansion of the square above vanishes. Thus

$$\begin{aligned} L_g(\varphi_g, \mathbf{U}_{n-1}) &= \beta \mathbb{E} \left[\int_0^T e^{\beta t} |\varphi_g(t, X_t) - G^\epsilon(t, X_t)|^2 dt \right] + \beta \mathbb{E} \left[\int_0^T e^{\beta t} |G^\epsilon(t, X_t) - Y_g|^2 dt \right] \\ &=: \beta \|\varphi_g - G^\epsilon\|_\beta^2 + C, \end{aligned}$$

where C is a constant independent of φ . Summing the components for V, G, H yields the result. ■

C.7 Proof of Theorem 9

We first prove Theorem 9-(a), the case with the weak simulator. For simplicity, we suppress the notation \mathbf{U}_{n-1} and use $L(\varphi), \tilde{L}^{(B)}(\varphi)$ for total loss and empirical total loss, respectively. The following lemma provides the upper bound of E_n in terms of the statistical error, approximation error and the bias due to the ZOD estimator.

Lemma 18 *The one step approximation error $E_n = \|\mathbf{U}_n - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}$ has the upper bound*

$$E_n^2 \leq 4 \sup_{\varphi \in \mathcal{H}} |L(\varphi) - \tilde{L}^{(B)}(\varphi)| + 2 \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2.$$

Proof Denote $\mathbf{U}_n^{\mathcal{H}} \in \arg \min_{\varphi \in \mathcal{H}} L(\varphi)$. It follows from the decomposition in Lemma 7 that

$$\begin{aligned} E_n^2 &= \|\mathbf{U}_n - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \\ &\leq 2 \|\mathbf{U}_n - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \\ &\leq 2(L(\mathbf{U}_n) - L(\mathcal{S}^\epsilon(\mathbf{U}_{n-1}))) + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \\ &\leq 2(L(\mathbf{U}_n) - L(\mathbf{U}_n^{\mathcal{H}})) + 2(L(\mathbf{U}_n^{\mathcal{H}}) - L(\mathcal{S}^\epsilon(\mathbf{U}_{n-1}))) + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \\ &= 2(L(\mathbf{U}_n) - L(\mathbf{U}_n^{\mathcal{H}})) + 2 \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2, \end{aligned} \tag{C.4}$$

where in the last equality we used the fact $L(\mathbf{U}_n^{\mathcal{H}}) - L(\mathcal{S}^\epsilon(\mathbf{U}_{n-1})) = \|\mathbf{U}_n^{\mathcal{H}} - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 = \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2$. For the first term on the right hand side of the above equation, note that

$$\begin{aligned} L(\mathbf{U}_n) - L(\mathbf{U}_n^{\mathcal{H}}) &= L(\mathbf{U}_n) - \tilde{L}^{(B)}(\mathbf{U}_n) + \tilde{L}^{(B)}(\mathbf{U}_n) - \tilde{L}^{(B)}(\mathbf{U}_n^{\mathcal{H}}) + \tilde{L}^{(B)}(\mathbf{U}_n^{\mathcal{H}}) - L(\mathbf{U}_n^{\mathcal{H}}) \\ &\leq L(\mathbf{U}_n) - \tilde{L}^{(B)}(\mathbf{U}_n) + \tilde{L}^{(B)}(\mathbf{U}_n^{\mathcal{H}}) - L(\mathbf{U}_n^{\mathcal{H}}) \\ &\leq 2 \sup_{\varphi \in \mathcal{H}} |\tilde{L}^{(B)}(\varphi) - L(\varphi)|, \end{aligned}$$

where the first inequality follows from $\tilde{L}^{(B)}(\mathbf{U}_n) - \tilde{L}^{(B)}(\mathbf{U}_n^{\mathcal{H}}) \leq 0$ due to $\mathbf{U}_n \in \arg \min_{\varphi \in \mathcal{H}} \tilde{L}^{(B)}(\varphi)$. This along with (C.4) yields the desired result. \blacksquare

The ZOD error It follows from Assumption 4 and Proposition 4 that

$$\|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \leq C\epsilon^4.$$

The statistical error Now we focus on the statistical error $\sup_{\varphi \in \mathcal{H}} |L(\varphi) - \tilde{L}^{(B)}(\varphi)|$. Denote the time discretization version of the loss function with respect to the time grid $\mathcal{G} := \{0 = t_0 < t_1 < \dots < t_N = T\}$ as

$$L^{\mathcal{G}}(\varphi) := L_v^{\mathcal{G}}(\varphi_v) + L_g^{\mathcal{G}}(\varphi_g) + L_h^{\mathcal{G}}(\varphi_h)$$

where

$$\begin{aligned} L_v^{\mathcal{G}}(\varphi_v) &:= \beta \mathbb{E} \left[\sum_{j=0}^{N-1} e^{\beta t_j} \left(\varphi_v(t_j, X_{t_j}) - \tilde{R}^{t_j, X_{t_j}}(\mathbf{U}_{n-1}) \right)^2 \Delta t_j \right], \\ L_g^{\mathcal{G}}(\varphi_g) &:= \beta \mathbb{E} \left[\sum_{j=0}^{N-1} e^{\beta t_j} \left| \varphi_g(t_j, X_{t_j}) - \frac{Z}{\epsilon} \tilde{R}^{t_j, X_{t_j} + \epsilon Z}(\mathbf{U}_{n-1}) \right|^2 \Delta t_j \right], \\ L_h^{\mathcal{G}}(\varphi_h) &:= \mathbb{E} \left[\sum_{j=0}^{N-1} e^{\beta t_j} \left| \varphi_h(t_j, X_{t_j}) - \frac{ZZ^\top - I_d}{\epsilon^2} \tilde{R}^{t_j, X_{t_j} + \epsilon Z}(\mathbf{U}_{n-1}) \right|^2 \Delta t_j \right]. \end{aligned}$$

Noting that $\mathbb{E}[\tilde{L}^{(B)}(\varphi)] = L^{\mathcal{G}}(\varphi)$, we have

$$\sup_{\varphi \in \mathcal{H}} \left| L(\varphi) - \tilde{L}^{(B)}(\varphi) \right| \leq \sup_{\varphi \in \mathcal{H}} \left| \tilde{L}^{(B)}(\varphi) - L^{\mathcal{G}}(\varphi) \right| + \sup_{\varphi \in \mathcal{H}} \left| L^{\mathcal{G}}(\varphi) - L(\varphi) \right|. \quad (\text{C.5})$$

The second term on the right hand side of (C.5) is the time discretization error, which is bounded by the following lemma.

Lemma 19 *Under Assumptions 3, 4, 1, there is a constant C independent of ϵ and N such that*

$$\sup_{\varphi \in \mathcal{H}} \left| L^{\mathcal{G}}(\varphi) - L(\varphi) \right| \leq C \epsilon^{-4} N^{-1}.$$

Proof We aim to bound the difference between the continuous-time population loss $L(\varphi)$ and its discrete-time approximation $L^{\mathcal{G}}(\varphi)$. We prove the Hessian loss for illustration, and the value and gradient components are treated in the same way and are of lower order in ϵ . Let the estimator be denoted by

$$\xi^\epsilon(Z) := \frac{ZZ^\top - I_d}{\epsilon^2}, \quad \tilde{f}(t, x) := f(t, x, \mathbf{U}_{n-1}(t, x)).$$

The continuous and discrete losses are defined as:

$$\begin{aligned} L_h(\varphi_h) &:= \int_0^T \mathbb{E} \left[e^{\beta t} \left| \varphi_h(t, X_t) - \xi^\epsilon(Z) R^{t, X_t + \epsilon Z}(\mathbf{U}_{n-1}) \right|^2 \right] dt, \\ L_h^{\mathcal{G}}(\varphi_h) &:= \sum_{i=0}^{N-1} \mathbb{E} \left[e^{\beta t_i} \left| \varphi_h(t_i, X_{t_i}) - \xi^\epsilon(Z) \tilde{R}^{t_i, X_{t_i} + \epsilon Z}(\mathbf{U}_{n-1}) \right|^2 \right] \Delta t_i, \end{aligned}$$

where $\tilde{R}^{t,x}(\mathbf{U}_{n-1})$ uses the discrete reward approximation

$$\tilde{R}^{t,x}(\mathbf{U}_{n-1}) = g(X_T^{t,x}) + \sum_i f(t_i, X_{t_i}^{t,x}, \mathbf{U}_{n-1}(t_i, X_{t_i}^{t,x})) \Delta t_i,$$

instead of the exact $R^{t,x}(\mathbf{U}_{n-1})$

$$R^{t,x}(\mathbf{U}_{n-1}) = g(X_T^{t,x}) + \int_t^T f(s, X_s^{t,x}, \mathbf{U}_{n-1}(s, X_s^{t,x})) ds.$$

Denote

$$\Delta(t, y) := \tilde{R}^{t,y}(\mathbf{U}_{n-1}) - R^{t,y}(\mathbf{U}_{n-1}).$$

Introduce the intermediate grid loss $\bar{L}_h^{\mathcal{G}}$ obtained by replacing the time integral in L_h with the grid sum while keeping the exact reward R .

$$\bar{L}_h(\varphi_h)^{\mathcal{G}} := \sum_{i=0}^{N-1} \mathbb{E} \left[e^{\beta t_i} \left| \varphi_h(t_i, X_{t_i}) - \xi^\epsilon(Z) R^{t_i, X_{t_i} + \epsilon Z}(\mathbf{U}_{n-1}) \right|^2 \right] \Delta t_i.$$

We have the following inner and outer error decompositions

$$|L_h(\varphi_h) - L_h^{\mathcal{G}}(\varphi_h)| \leq \underbrace{\left| L_h^{\mathcal{G}}(\varphi_h) - \bar{L}_h^{\mathcal{G}}(\varphi_h) \right|}_{\text{Inner Discretization Error}} + \underbrace{\left| \bar{L}_h^{\mathcal{G}}(\varphi_h) - L_h(\varphi_h) \right|}_{\text{Outer Discretization Error}}.$$

Therefore, there are two types of error – the inner and the outer discretization errors. For the inner discretization error, it suffices to bound

$$\begin{aligned} & \left| \mathbb{E} \left[\left\| \varphi_h(t_i, X_{t_i}) - \xi^\epsilon(Z) R^{t_i, X_{t_i} + \epsilon Z} \right\|^2 \right] - \mathbb{E} \left[\left\| \varphi_h(t_i, X_{t_i}) - \xi^\epsilon(Z) \tilde{R}^{t_i, X_{t_i} + \epsilon Z} \right\|^2 \right] \right| \\ & \leq \mathbb{E} \left[\left| 2\varphi_h(t_i, X_{t_i}) - \xi^\epsilon(Z) (\tilde{R}^{t_i, X_{t_i} + \epsilon Z} + R^{t_i, X_{t_i} + \epsilon Z}) \right| |\xi^\epsilon(Z)| |\Delta(t_i, X_{t_i} + \epsilon Z)| \right]. \end{aligned} \quad (\text{C.6})$$

Due to the polynomial growth condition and the moment bounds of X_t , it suffices to bound $\|\Delta(t_i, X_{t_i} + \epsilon Z)\|_{L^2}$. To this end, recall that $R^{t,y} = g(X_T^{t,y}) + \int_t^T \tilde{f}(s, X_s^{t,y}) ds$. The terminal part cancels and

$$\Delta(t_i, y) = \sum_{k=i}^{N-1} \int_{t_k}^{t_{k+1}} \left[\tilde{f}(t_k, X_{t_k}^{t_i, y}) - \tilde{f}(s, X_s^{t_i, y}) \right] ds.$$

By Assumption 4, Itô's formula gives

$$d\tilde{f}(s, X_s^{t_i, y}) = A_s ds + B_s dW_s, \quad \mathbb{E} [|A_s|^2 + |B_s|^2] \leq C(1 + |y|^q)$$

for some $q \geq 0$, uniformly in s . Stochastic Fubini therefore yields

$$\Delta(t_i, y) = - \sum_{k=i}^{N-1} \int_{t_k}^{t_{k+1}} (t_{k+1} - u) A_u du - \sum_{k=i}^{N-1} \int_{t_k}^{t_{k+1}} (t_{k+1} - u) B_u dW_u.$$

The drift term has L^2 norm at most $CN(T/N)^2 \leq C/N$. For the martingale term, Itô's isometry and orthogonality of martingale increments give

$$\mathbb{E} \left[\sum_{k=i}^{N-1} \int_{t_k}^{t_{k+1}} (t_{k+1} - u) B_u dW_u \right]^2 \leq CNh^3 \leq \frac{C}{N^2}.$$

Hence

$$\|\Delta(t_i, y)\|_{L^2} \leq \frac{C(1 + |y|^q)}{N}.$$

Conditioning on Z , taking $y = X_{t_i} + \epsilon Z$, and using the moment bounds for X_{t_i} together with Gaussian moments,

$$\|\Delta(t_i, X_{t_i} + \epsilon Z)\|_{L^2} \leq \frac{C}{N}.$$

This along with (C.6) yields

$$\sup_{\varphi_h \in \mathcal{H}} |L_h^{\mathcal{G}}(\varphi_h) - \bar{L}_h^{\mathcal{G}}(\varphi_h)| \leq \frac{C}{N\epsilon^4}.$$

Now we bound the outer discretization error:

$$\left| \int_0^T l(t) dt - \sum_{i=0}^{N-1} l(t_i) \Delta t_i \right|, \quad \text{where } l(t) = e^{\beta t} \mathbb{E} \left[\|\varphi_h(t, X_t) - \xi^\epsilon(Z) R^{t, X_t + \epsilon Z}\|^2 \right].$$

It suffices to show that the time derivative $\frac{d}{dt} l(t)$ is bounded. Indeed,

$$\begin{aligned} l(t) &= e^{\beta t} \mathbb{E} \left[\|\varphi_h(t, X_t) - \xi^\epsilon(Z) R^{t, X_t + \epsilon Z}\|^2 \right] \\ &= e^{\beta t} \left(\mathbb{E} \left[\|\varphi_h(t, X_t)\|^2 \right] + \mathbb{E} \left[\|\xi^\epsilon(Z) R^{t, X_t + \epsilon Z}\|^2 \right] - 2\mathbb{E} \left[\langle \varphi_h(t, X_t), \xi^\epsilon(Z) R^{t, X_t + \epsilon Z} \rangle \right] \right). \end{aligned}$$

Since φ_h is smooth and bounded, we need only to analyze the second and third terms.

For the second term, using the conditional expectation, we have

$$I_2(t) := \mathbb{E} \left[\|\xi^\epsilon(Z) R^{t, X_t + \epsilon Z}\|^2 \right] = \mathbb{E} \left[\|\xi^\epsilon(Z)\|^2 \mathbb{E} \left[(R^{t, X_t + \epsilon Z})^2 \mid Z, X_t \right] \right].$$

Denote $u(t, x) := \mathbb{E}[(R^{t, x})^2]$. Under Assumption 4, the derivatives of u satisfy the polynomial growth condition. Therefore, the time derivative of the expectation is:

$$\begin{aligned} \frac{d}{dt} \mathbb{E} \left[\|\xi^\epsilon(Z) R^{t, X_t + \epsilon Z}\|^2 \right] &= \mathbb{E} \left[\|\xi^\epsilon(Z)\|^2 \frac{d}{dt} u(t, X_t + \epsilon Z) \right] \\ &= \mathbb{E} \left[\|\xi^\epsilon(Z)\|^2 (\partial_t + \mathcal{L}) F_Z(t, x) \Big|_{x=X_t} \right], \end{aligned}$$

where $F_Z(t, x) = u(t, x + \epsilon Z)$. This confirms that $|\frac{d}{dt} I_2(t)| \leq C\epsilon^{-4}$.

Similarly, for the cross term involving $\mathbb{E}[\langle \varphi_h, \xi^\epsilon(Z) R^{t, X_t + \epsilon Z} \rangle]$, the derivative involves terms related to $\mathbb{E}[R^{t, x}]$. Using Assumption 4, the derivative is bounded by $C\epsilon^{-2}$.

Combining these bounds, the total discretization error is bounded by:

$$\sup_{\varphi \in \mathcal{H}} |L_h^{\mathcal{G}}(\varphi) - L_h(\varphi)| \leq C (\epsilon^{-4} N^{-1} + \epsilon^{-2} N^{-1}) \leq C \epsilon^{-4} N^{-1}.$$

This proves the desired result. ■

The first term on the right hand side of (C.5) is the classical statistical error, and we aim to bound it via the Rademacher complexity. Typically, the Rademacher complexity framework works for bounded loss functions. We first truncate the loss function and denote

G as the implicit function and W as the underlying random variable, such that $L^{\mathcal{G}}(\varphi) = \mathbb{E}[G(\varphi; W)]$ and $\tilde{L}^{(B)}(\varphi) = \frac{1}{B} \sum_{i=1}^B G(\varphi; W_i)$. Write the grid loss as

$$G(\varphi; W) = \sum_{j=0}^{N-1} \Delta t_j \ell_j(\varphi; W),$$

where ℓ_j is the one-point squared value-gradient-Hessian loss at time t_j . For each grid index, define the local truncation event

$$\Omega_{K,j} := \left\{ \sup_{0 \leq t \leq T} |X_t| < K \right\} \cap \{|Z| < K\} \cap \left\{ \sup_{t_j \leq s \leq T} |X_s^{t_j, X_{t_j} + \epsilon Z}| < K \right\}.$$

We clip the grid loss term by term:

$$G^{(K)}(\varphi; W) := \sum_{j=0}^{N-1} \Delta t_j \ell_j(\varphi; W) \mathbf{1}_{\Omega_{K,j}}.$$

By Assumption 4, there exists a finite exponent $q \geq 1$, depending only on the polynomial growth orders of $g(x)$, $J_1(t, x; \varphi)$, $J_2(t, x; \varphi)$, and $\tilde{f}(t, x; \varphi)$ uniformly over $\varphi \in \mathcal{H}$, such that on $\Omega_{K,j}$ the one-point value, gradient, and Hessian targets are bounded by $C\epsilon^{-2}(1 + K^q)$. As a consequence of this truncation, $G^{(K)}$ has the following bound

$$|G^{(K)}(\varphi; W)| \leq C\epsilon^{-4}(1 + K^q).$$

Next, we aim to derive the probability of the truncation sets. To this end, we need the following lemmas. To derive the tail probability of the trajectory of X , we will use the following Bernstein martingale inequality from Dzhaparidze and Van Zanten (2001).

Lemma 20 *Let M_t be a continuous square integrable martingale. Then*

$$\mathbb{P} \left(\sup_{0 \leq t \leq T} |M_t| \geq z, \langle M \rangle_T \leq L \right) \leq 2 \exp \left(-\frac{1}{2} \frac{z^2}{L} \right).$$

Based on this lemma, we state the tail probability of the trajectory of X .

Lemma 21 (Sub-Gaussian Tail of Base Trajectory) *Under Assumption 1, let X be the solution to the SDE (2.2) starting from X_0 . Then there exist constants $C_1, C_2 > 0$ depending only on T, L_b, σ_0 and X_0 , such that for any $K > 0$:*

$$\mathbb{P} \left(\sup_{0 \leq t \leq T} |X_t| > K \right) \leq C_1 \exp(-C_2 K^2).$$

Proof The SDE in integral form is given by:

$$X_t = X_0 + \int_0^t b(s, X_s) ds + M_t, \quad \text{where } M_t = \sqrt{2} \int_0^t \sigma(s, X_s) dW_s.$$

Note that M_t is a continuous square-integrable martingale. Taking the norm and using the linear growth condition we get

$$\begin{aligned} |X_t| &\leq |X_0| + \int_0^t L_b(1 + |X_s|) ds + |M_t| \\ &= (|X_0| + L_b T + |M_t|) + L_b \int_0^t |X_s| ds. \end{aligned}$$

Let $M_T^* := \sup_{0 \leq t \leq T} |M_t|$. Applying Grönwall's inequality yields a path-wise bound:

$$\sup_{0 \leq t \leq T} |X_t| \leq (|X_0| + L_b T + M_T^*) e^{L_b T}.$$

Therefore, the event $\{\sup_{0 \leq t \leq T} |X_t| > K\}$ implies:

$$M_T^* > K e^{-L_b T} - (|X_0| + L_b T).$$

For sufficiently large K , the RHS is approximately proportional to K . Thus, it suffices to bound the tail of M_T^* . We invoke Bernstein's inequality for continuous martingales (Lemma 20). Under the bounded diffusion assumption, the quadratic variation is deterministically bounded:

$$\langle M \rangle_T = \int_0^T \|\sigma(s, X_s)\|^2 ds \leq T \sigma_{\max}^2.$$

Setting $L = T \sigma_{\max}^2$, we obtain:

$$\mathbb{P}(M_T^* \geq z) \leq 2 \exp\left(-\frac{z^2}{2T \sigma_{\max}^2}\right).$$

Combining this with the Grönwall bound proves the lemma. ■

Recall the ψ_2 -Orlicz norm (sub-Gaussian norm) is defined as $\|X\|_{\psi_2} = \inf\{C > 0 : \mathbb{E}[\exp(|X|^2/C^2)] \leq 2\}$.

As a direct consequence of Lemma 21, the supremum of the base trajectory satisfies:

$$\left\| \sup_{0 \leq t \leq T} |X_t| \right\|_{\psi_2} \leq C_{base}, \quad (\text{C.7})$$

where C_{base} is a constant depending only on the SDE coefficients and T .

We now extend this analysis to the branching paths used in the ZOD estimation. Let $\mathcal{G} = \{t_1, \dots, t_N\}$ be the time discretization grid. For each j , let $X_s^{t_j, \xi_j}$ denote the conditional path starting at time t_j from $\xi_j = X_{t_j} + \epsilon Z$. The path satisfies:

$$X_s^{t_j, \xi_j} = \xi_j + \int_{t_j}^s b(u, X_u^{t_j, \xi_j}) du + \sqrt{2} \int_{t_j}^s \sigma(u, X_u^{t_j, \xi_j}) dW_u, \quad s \in [t_j, T].$$

The following lemma bounds the branching trajectories.

Lemma 22 (Uniform Tail for Branching Paths) *Under Assumption 1, there exists a constant $C > 0$ independent of j, N such that for any $K > 0$:*

$$\sup_{0 \leq j < N} \mathbb{P} \left(\sup_{t_j \leq s \leq T} |X_s^{t_j, X_{t_j} + \epsilon Z}| > K \right) \leq 2 \exp(-CK^2).$$

The same bounds hold with $X_{t_j} - \epsilon Z$ and X_{t_j} in place of $X_{t_j} + \epsilon Z$.

Proof Fix an index $j \in \{1, \dots, N\}$. Let $X_s^{(j)} := X_s^{t_j, X_{t_j} + \epsilon Z}$. By applying Grönwall's inequality under the linear growth condition on b , we obtain the pathwise bound:

$$\sup_{s \in [t_j, T]} |X_s^{(j)}| \leq e^{L_b T} \left(|X_{t_j}| + \epsilon |Z| + L_b T + \sup_{s \in [t_j, T]} |M_s^{(j)}| \right),$$

where $M_s^{(j)} = \sqrt{2} \int_{t_j}^s \sigma(u, X_u^{(j)}) dW_u$.

We analyze the Orlicz ψ_2 -norm of each term on the right-hand side of the above:

1. **Base Path:** By (C.7), we have $\|\sup |X_t|\|_{\psi_2} \leq C_{base}$.
2. **Perturbation:** Since $Z \sim \mathcal{N}(0, I_d)$, it is sub-Gaussian with $\|Z\|_{\psi_2} \leq C_{Gauss}$.
3. **Martingale:** Since the diffusion σ is uniformly bounded, the quadratic variation is bounded by $\sigma_{\max}^2 T$. By Bernstein's inequality for continuous martingales, $\sup_s |M_s^{(j)}|$ is sub-Gaussian with norm bounded by a constant C_{mart} independent of the path history.

By the triangle inequality for the ψ_2 -norm, the supremum of the j -th branching path is sub-Gaussian:

$$\left\| \sup_{t_j \leq s \leq T} |X_s^{(j)}| \right\|_{\psi_2} \leq C_{path}, \tag{C.8}$$

where C_{path} is independent of j and N .

The tail bound follows directly from (C.8) and the definition of the ψ_2 -norm. The proof for the branch started at $X_{t_j} - \epsilon Z$ is identical. \blacksquare

Combining Lemma 21, the Gaussian tail of Z , and Lemma 22 gives the probability of the local truncation events $\Omega_{K,j}^c$ used in the proof of Theorem 9.

Lemma 23 *Under Assumption 1, there exist constants $C_1, C_2 > 0$, independent of j and N , such that*

$$\sup_{0 \leq j < N} \mathbb{P}(\Omega_{K,j}^c) \leq C_1 \exp(-C_2 K^2).$$

We are ready to obtain the upper bound of the first term on the right hand side of (C.5). It can be decomposed as

$$\begin{aligned}
 \sup_{\varphi \in \mathcal{H}} \left| \tilde{L}^{(B)}(\varphi) - L^{\mathcal{G}}(\varphi) \right| &= \sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B G(\varphi; W_i) - \mathbb{E}[G(\varphi; W)] \right| \\
 &\leq \sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B G^{(K)}(\varphi; W_i) - \mathbb{E}[G^{(K)}(\varphi; W)] \right| \\
 &\quad + \sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B \left(G^{(K)}(\varphi; W_i) - G(\varphi; W_i) \right) \right| \\
 &\quad + \sup_{\varphi \in \mathcal{H}} \mathbb{E} \left[\left| G^{(K)}(\varphi; W) - G(\varphi; W) \right| \right].
 \end{aligned} \tag{C.9}$$

Let

$$R_K(W) := \sup_{\varphi \in \mathcal{H}} |G(\varphi; W) - G^{(K)}(\varphi; W)|.$$

Since the networks in \mathcal{H} are uniformly bounded, the supremum over φ in the local squared loss is bounded pointwise by the reward targets:

$$\sup_{\varphi \in \mathcal{H}} \ell_j(\varphi; W) \leq C \left(1 + |Y_{v,j}|^2 + \|Y_{g,j}^\epsilon\|^2 + \|Y_{h,j}^\epsilon\|^2 \right),$$

where

$$Y_{v,j} := \tilde{R}^{t_j, X_{t_j}}(\mathbf{U}_{n-1}), \quad Y_{g,j}^\epsilon := \frac{Z}{\epsilon} \tilde{R}^{t_j, X_{t_j} + \epsilon Z}(\mathbf{U}_{n-1}),$$

and

$$Y_{h,j}^\epsilon := \frac{ZZ^\top - I_d}{\epsilon^2} \tilde{R}^{t_j, X_{t_j} + \epsilon Z}(\mathbf{U}_{n-1}).$$

The assumption 4 and the polynomial growth of g imply that, for some $q \geq 0$,

$$|R^{t_j, y}(\mathbf{U}_{n-1})| + |\tilde{R}^{t_j, y}(\mathbf{U}_{n-1})| \leq C \left(1 + \sup_{t_j \leq s \leq T} |X_s^{t_j, y}|^q \right).$$

Together with Assumption 1 and the Gaussian moments of Z , this gives

$$\sup_{0 \leq j < N} \mathbb{E} \left[\left(\sup_{\varphi \in \mathcal{H}} \ell_j(\varphi; W) \right)^2 \right] \leq C \epsilon^{-8}.$$

Hence, by Cauchy–Schwarz applied to each grid summand, Lemma 23, and $\sum_{j=0}^{N-1} \Delta t_j = T$,

$$\begin{aligned}
 \mathbb{E} R_K(W) &\leq \sum_{j=0}^{N-1} \Delta t_j \mathbb{E} \left[\sup_{\varphi \in \mathcal{H}} \ell_j(\varphi; W) \mathbf{1}_{\Omega_{K,j}^c} \right] \\
 &\leq \sum_{j=0}^{N-1} \Delta t_j \left(\mathbb{E} \left[\left(\sup_{\varphi \in \mathcal{H}} \ell_j(\varphi; W) \right)^2 \right] \right)^{1/2} \mathbb{P}(\Omega_{K,j}^c)^{1/2} \\
 &\leq C \epsilon^{-4} \exp(-CK^2).
 \end{aligned} \tag{C.10}$$

In particular,

$$\sup_{\varphi \in \mathcal{H}} \mathbb{E} \left[|G^{(K)}(\varphi; W) - G(\varphi; W)| \right] \leq \mathbb{E} R_K(W) \leq C\epsilon^{-4} \exp(-CK^2).$$

Therefore, by Markov's inequality, with probability at least $1 - \delta$,

$$\frac{1}{B} \sum_{i=1}^B R_K(W_i) + \mathbb{E} R_K(W) \leq C\delta^{-1} \epsilon^{-4} \exp(-CK^2).$$

For the first term on the right hand side of (C.9), we use the Rademacher complexity. Denote $S = (W_1, \dots, W_B)$ and define

$$\Phi(S) := \sup_{\varphi} \left| \frac{1}{B} \sum_{i=1}^B G^{(K)}(\varphi; W_i) - \mathbb{E} \left[G^{(K)}(\varphi; W) \right] \right|.$$

Let $S' = (W_1, \dots, W'_j, \dots, W_B)$ differ from S only at j -th component. It follows from Jensen's inequality and the upper bound of $G^{(K)}$ that

$$|\Phi(S) - \Phi(S')| \leq \sup_{\varphi \in \mathcal{H}} \frac{1}{B} \left| G^{(K)}(\varphi; W_j) - G^{(K)}(\varphi; W'_j) \right| \leq \frac{C\epsilon^{-4}(1+K^q)}{B}.$$

By McDiarmid's inequality,

$$\mathbb{P} \left(\Phi(S) - \mathbb{E}[\Phi(S)] > C\epsilon^{-4}(1+K^q) \sqrt{\frac{\log(1/\delta)}{B}} \right) < \delta. \quad (\text{C.11})$$

Now we use the Rademacher complexity to bound $\mathbb{E}[\Phi]$. Let W' be the independent copy of W , and $\{\sigma_i\}_{i=1}^B$ are i.i.d. random variables with $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$. Then

$$\begin{aligned} \mathbb{E}[\Phi(S)] &= \mathbb{E} \left[\sup_{\varphi} \left| \frac{1}{B} \sum_{i=1}^B G^{(K)}(\varphi; W_i) - \mathbb{E} \left[G^{(K)}(\varphi; W) \right] \right| \right] \\ &= \mathbb{E} \left[\sup_{\varphi \in \mathcal{H}} \left| \mathbb{E}_{W'} \left[\frac{1}{B} \sum_{i=1}^B \left(G^{(K)}(\varphi; W'_i) - G^{(K)}(\varphi; W_i) \right) \right] \right| \right] \\ &= \mathbb{E} \left[\sup_{\varphi \in \mathcal{H}} \left| \mathbb{E}_{W', \sigma} \left[\frac{1}{B} \sum_{i=1}^B \sigma_i \left(G^{(K)}(\varphi; W'_i) - G^{(K)}(\varphi; W_i) \right) \right] \right| \right] \\ &\leq \mathbb{E}_{W, W', \sigma} \left[\sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B \sigma_i G^{(K)}(\varphi; W'_i) \right| \right] + \mathbb{E}_{W, W', \sigma} \left[\sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B \sigma_i G^{(K)}(\varphi; W_i) \right| \right] \\ &\leq 2\mathcal{R}_B(G^{(K)}(\mathcal{H}) \cup -G^{(K)}(\mathcal{H})) \end{aligned} \quad (\text{C.12})$$

where $G^{(K)}(\mathcal{H}) = \{G^{(K)}(\varphi; \cdot) | \varphi \in \mathcal{H}\}$. For the Rademacher complexity in terms of the parameterized function class, we have the following lemma by Jiao et al. (2024).

Lemma 24 *Let \mathcal{G} be a parameterized function class $\mathcal{G} = \{f(\cdot; \theta) : \mathbb{R}^d \rightarrow \mathbb{R} \mid \theta \in \Theta\}$, where $\Theta = \{\theta \mid \theta \in \mathbb{R}^p, \|\theta\|_\infty < R\}$. If there are constants C_1, C_2 , such that*

$$\begin{aligned} \sup_{x \in \mathbb{R}^d} |f(x; \theta_1) - f(x; \theta_2)| &\leq C_1 \|\theta_1 - \theta_2\|_\infty, \\ \sup_{\theta \in \Theta} \sup_{x \in \mathbb{R}^d} |f(x; \theta)| &\leq C_2, \end{aligned}$$

then

$$\mathcal{R}_m(\mathcal{G}) \leq \frac{4}{\sqrt{m}} + \frac{6\sqrt{p}C_2}{\sqrt{m}} \sqrt{\log(2RC_1 p \sqrt{m})}.$$

Based on this lemma, it follows from Assumption 3 that

$$\begin{aligned} &\mathcal{R}_B(G^{(K)}(\mathcal{H}) \cup -G^{(K)}(\mathcal{H})) \\ &\leq \frac{4}{\sqrt{B}} + \frac{6\sqrt{p}C(1 + \epsilon^{-4}(1 + K^q))}{\sqrt{B}} \sqrt{\log(2RC\epsilon^{-2}(1 + K^q)p\sqrt{B})}, \end{aligned} \quad (\text{C.13})$$

where C is a constant depending only on the coefficients of (2.1). The desired result follows now from (C.11), (C.10), (C.12) and (C.13) by choosing $K = C(\log B + \log(1/\delta))^{1/2}$ with C large enough, which gives the factor $L_{B,\delta}^q$ after increasing q once more if necessary.

We now prove Theorem 9-(b) with the strong simulator. The only parts that differ from the preceding one-point proof are the ZOD target, the discretization estimate, and the envelope used in the empirical-process bound. The first entry in the minimum defining $\Gamma_{\epsilon,B,\delta}$ follows from the same truncation and Rademacher-complexity argument as in the weak-simulator case. We therefore focus below on the second entry of the minimum under the strong-simulator coupling. In the remainder of this proof, all the rewards

$$R^{t,x+\epsilon Z}(\mathbf{U}_{n-1}), \quad R^{t,x-\epsilon Z}(\mathbf{U}_{n-1}), \quad R^{t,x}(\mathbf{U}_{n-1})$$

are evaluated under the same Brownian path within one strong-simulator query. The exact multi-point derivative targets are the two central differences

$$\frac{Z}{2\epsilon} (R^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) - R^{t,x-\epsilon Z}(\mathbf{U}_{n-1})),$$

and

$$\frac{ZZ^\top - I_d}{2\epsilon^2} (R^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) + R^{t,x-\epsilon Z}(\mathbf{U}_{n-1}) - 2R^{t,x}(\mathbf{U}_{n-1})).$$

By Gaussian symmetry and linearity, their conditional means are G^ϵ and H^ϵ . Let L_m be the corresponding population loss and let $\tilde{L}_m^{(B)}$ be its quadrature empirical version. By the same conditional-expectation decomposition as in Lemma 7,

$$L_m(\varphi, \mathbf{U}_{n-1}) = \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta,\text{mix}}^2 + C_m,$$

where C_m is independent of φ . Hence the ERM argument in Lemma 18 gives

$$\begin{aligned} E_n^2 &\leq 4 \sup_{\varphi \in \mathcal{H}} |L_m(\varphi) - \tilde{L}_m^{(B)}(\varphi)| + 2 \inf_{\varphi \in \mathcal{H}} \|\varphi - \mathcal{S}^\epsilon(\mathbf{U}_{n-1})\|_{\beta,\text{mix}}^2 \\ &\quad + 2 \|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta,\text{mix}}^2. \end{aligned} \quad (\text{C.14})$$

By Proposition 5, the multi-point estimators have the same $O(\epsilon^2)$ bias as the one-point estimators under the stated smoothness assumptions. Therefore,

$$\|\mathcal{S}^\epsilon(\mathbf{U}_{n-1}) - \mathcal{S}(\mathbf{U}_{n-1})\|_{\beta, \text{mix}}^2 \leq C\epsilon^4. \quad (\text{C.15})$$

It remains to bound the statistical error term in (C.14). Let $L_m^\mathcal{G}$ be the time-discretized version of L_m . As before,

$$\sup_{\varphi \in \mathcal{H}} \left| L_m(\varphi) - \tilde{L}_m^{(B)}(\varphi) \right| \leq \sup_{\varphi \in \mathcal{H}} \left| \tilde{L}_m^{(B)}(\varphi) - L_m^\mathcal{G}(\varphi) \right| + \sup_{\varphi \in \mathcal{H}} \left| L_m^\mathcal{G}(\varphi) - L_m(\varphi) \right|. \quad (\text{C.16})$$

We first control the discretization term. To separate the two sources of time discretization, let $\bar{L}_m^\mathcal{G}$ denote the time-grid population loss obtained from L_m by replacing the time integral with the grid sum while keeping the exact rewards $R^{t,x}(\mathbf{U}_{n-1})$ unchanged. Then

$$\sup_{\varphi \in \mathcal{H}} \left| L_m^\mathcal{G}(\varphi) - L_m(\varphi) \right| \leq \sup_{\varphi \in \mathcal{H}} \left| L_m^\mathcal{G}(\varphi) - \bar{L}_m^\mathcal{G}(\varphi) \right| + \sup_{\varphi \in \mathcal{H}} \left| \bar{L}_m^\mathcal{G}(\varphi) - L_m(\varphi) \right|.$$

Write

$$\Delta(t, x) := \tilde{R}^{t,x}(\mathbf{U}_{n-1}) - R^{t,x}(\mathbf{U}_{n-1}).$$

Because the three rewards in the multi-point estimator are generated by the same Brownian path and the same quadrature grid, the discretization error enters through the finite differences

$$\frac{Z}{2\epsilon} \{ \Delta(t, x + \epsilon Z) - \Delta(t, x - \epsilon Z) \}$$

and

$$\frac{ZZ^\top - I_d}{2\epsilon^2} \{ \Delta(t, x + \epsilon Z) + \Delta(t, x - \epsilon Z) - 2\Delta(t, x) \}.$$

Under the regularity assumptions $b, \sigma \in C_b^{0,5}$, $f(\cdot, \cdot, \mathbf{U}_{n-1}(\cdot, \cdot)) \in C_p^{1,5}$, and $g \in C_p^4$, using the same method as in the proof of Lemma 19, there are constants $C > 0, q > 0$ such that

$$\|\partial_x^r \Delta(t, x)\|_{L^2} \leq \frac{C}{N} (1 + |x|^q).$$

Applying Taylor's expansion yields

$$\begin{aligned} \left\| \frac{Z}{2\epsilon} \{ \Delta(t, X_t + \epsilon Z) - \Delta(t, X_t - \epsilon Z) \} \right\|_{L^2} &\leq \frac{C}{N}, \\ \left\| \frac{ZZ^\top - I_d}{2\epsilon^2} \{ \Delta(t, X_t + \epsilon Z) + \Delta(t, X_t - \epsilon Z) - 2\Delta(t, X_t) \} \right\|_{L^2} &\leq \frac{C}{N}. \end{aligned}$$

Using the same method as in the proof of Lemma 19 gives

$$\sup_{\varphi \in \mathcal{H}} \left| L_m^\mathcal{G}(\varphi) - \bar{L}_m^\mathcal{G}(\varphi) \right| \leq \frac{C}{N}.$$

The outer discretization error is treated in the same way, leading to

$$\sup_{\varphi \in \mathcal{H}} \left| L_m^\mathcal{G}(\varphi) - L_m(\varphi) \right| \leq \frac{C}{N}. \quad (\text{C.17})$$

For the first term on the right hand side of (C.16), denote and note that

$$\begin{aligned}
 Y_g^{\epsilon,m}(t,x) &:= \frac{Z}{2\epsilon} \left(\tilde{R}^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) - \tilde{R}^{t,x-\epsilon Z}(\mathbf{U}_{n-1}) \right) \\
 &= Z \int_{-1}^1 \nabla_x \tilde{R}^{t,x+s\epsilon Z}(\mathbf{U}_{n-1})^\top Z \frac{ds}{2}, \\
 Y_h^{\epsilon,m}(t,x) &:= \frac{ZZ^\top - I_d}{2\epsilon^2} \left(\tilde{R}^{t,x+\epsilon Z}(\mathbf{U}_{n-1}) + \tilde{R}^{t,x-\epsilon Z}(\mathbf{U}_{n-1}) - 2\tilde{R}^{t,x}(\mathbf{U}_{n-1}) \right) \\
 &= \frac{ZZ^\top - I_d}{2} \int_{-1}^1 (1-|s|) Z^\top \nabla_x^2 \tilde{R}^{t,x+s\epsilon Z}(\mathbf{U}_{n-1}) Z ds.
 \end{aligned} \tag{C.18}$$

Set

$$\tilde{f}(s,x) := f(s,x, \mathbf{U}_{n-1}(s,x)), \quad D_s^{(r)} := \partial_x^r X_s^{t,x}, \quad r = 1, \dots, 4.$$

For $D^{(1)}$, differentiating the SDE gives a linear equation with bounded drift and multiplicative diffusion coefficients:

$$dD_s^{(1)} = \nabla_x b(s, X_s^{t,x}) D_s^{(1)} ds + \sqrt{2} \sum_{\ell=1}^d \nabla_x \sigma_\ell(s, X_s^{t,x}) D_s^{(1)} dW_s^\ell.$$

For $r = 2, 3, 4$, $D^{(r)}$ satisfies

$$dD_s^{(r)} = A_s^{(r)} D_s^{(r)} ds + \sqrt{2} \sum_{\ell=1}^d B_{\ell,s}^{(r)} D_s^{(r)} dW_s^\ell + F_s^{(r)} ds + \sqrt{2} \sum_{\ell=1}^d H_{\ell,s}^{(r)} dW_s^\ell, \tag{C.19}$$

where $A_s^{(r)}$ and $B_{\ell,s}^{(r)}$ are bounded, and the forcing terms depend polynomially on lower-order derivatives:

$$\begin{aligned}
 |F_s^{(2)}| + \sum_{\ell} |H_{\ell,s}^{(2)}| &\leq C |D_s^{(1)}|^2, \\
 |F_s^{(3)}| + \sum_{\ell} |H_{\ell,s}^{(3)}| &\leq C \left(|D_s^{(1)}|^3 + |D_s^{(1)}| |D_s^{(2)}| \right), \\
 |F_s^{(4)}| + \sum_{\ell} |H_{\ell,s}^{(4)}| &\leq C \left(|D_s^{(1)}|^4 + |D_s^{(1)}|^2 |D_s^{(2)}| + |D_s^{(2)}|^2 + |D_s^{(1)}| |D_s^{(3)}| \right).
 \end{aligned}$$

Applying BDG and Grönwall first to $D^{(1)}$ and then inductively to (C.19) yields, for every $p \geq 2$,

$$\sup_{t,x} \left\| 1 + \sum_{r=1}^4 \sup_{t \leq s \leq T} |D_s^{(r)}| \right\|_{L^p} \leq C e^{cp}. \tag{C.20}$$

Since

$$\tilde{R}^{t_j,x}(\mathbf{U}_{n-1}) = g(X_T^{t_j,x}) + \sum_{i=j}^{N-1} \tilde{f}(t_i, X_{t_i}^{t_j,x}) \Delta t_i,$$

Faà di Bruno's formula along with the fact that $g \in C_p^4$, $\tilde{f} \in C_p^{1,5}$ and the bound (C.20) implies that for every $p > 0$, there are constant $C, c > 0$ and $q > 0$ such that

$$\sup_j \left\| |\nabla_x \tilde{R}^{t_j,x}(\mathbf{U}_{n-1})| + |\nabla_x^2 \tilde{R}^{t_j,x}(\mathbf{U}_{n-1})| \right\|_{L^p} \leq C e^{cp} (1 + |x|^q).$$

Combining (C.18) with

$$\|Z\|_{L^p} \leq Cp^{1/2}, \quad \|ZZ^\top - I_d\|_{L^p} \leq Cp,$$

and the moment bound of X_t gives

$$\sup_{0 < \epsilon \leq 1} \sup_j \left\| |Y_g^{\epsilon, m}(t_j, X_{t_j})| + |Y_h^{\epsilon, m}(t_j, X_{t_j})| \right\|_{L^p} \leq Ce^{c_p}, \quad p \geq 2. \quad (\text{C.21})$$

Define the truncation event

$$\Omega_{K, m, j} := \left\{ \sup_{t_j \leq t \leq T} |X_t^{t_j, X_{t_j}}| < K \right\} \cap \{|Y_g^{\epsilon, m}(t_j, X_{t_j})| \leq K\} \cap \{|Y_h^{\epsilon, m}(t_j, X_{t_j})| \leq K\}.$$

With $q_K = \max\{2, \lfloor \alpha \log(1 + K) \rfloor\}$, Markov's inequality and (C.21) give

$$\begin{aligned} \mathbb{P}(|Y_g^{\epsilon, m}(t_j, X_{t_j})| > K) &\leq \left(\frac{Ce^{c_p q_K}}{K} \right)^{q_K}, \\ \mathbb{P}(|Y_h^{\epsilon, m}(t_j, X_{t_j})| > K) &\leq \left(\frac{Ce^{c_p q_K}}{K} \right)^{q_K}. \end{aligned}$$

Choosing $\alpha > 0$ small enough and K large enough yields

$$\sup_{1 \leq j \leq N} \mathbb{P}(\Omega_{K, m, j}^c) \leq C_1 \exp(-C_2(\log(1 + K))^2). \quad K \geq 2, \quad (\text{C.22})$$

Write the time-discretized sample loss as

$$G_m(\varphi; W) = \sum_{j=0}^{N-1} \Delta t_j \ell_{m, j}(\varphi; W),$$

where $\ell_{m, j}$ is the squared value-gradient-Hessian loss at the grid time t_j . Clip the grid loss term by the term:

$$G_m^{(K)}(\varphi; W) := \sum_{j=0}^{N-1} \Delta t_j \ell_{m, j}(\varphi; W) \mathbf{1}_{\Omega_{K, m, j}}.$$

Hence, after summing over the time grid and using $\sum_{j=0}^{N-1} \Delta t_j = T$,

$$|G_m^{(K)}(\varphi; W)| \leq C(1 + K^q),$$

where q depends on the polynomial growth condition of g and f . Recall the decomposition (C.9), by McDiarmid's inequality and the parameterized Rademacher bound,

$$\begin{aligned} &\sup_{\varphi \in \mathcal{H}} \left| \frac{1}{B} \sum_{i=1}^B G_m^{(K)}(\varphi; W_i) - \mathbb{E}[G_m^{(K)}(\varphi; W)] \right| \\ &\leq C(1 + \sqrt{\log(1/\delta)}) \frac{\mathcal{P}(d)(1 + K^q)}{\sqrt{B}} \sqrt{\log(CRp(1 + K^q)\sqrt{B})}. \end{aligned}$$

with probability at least $1 - \delta$. Let

$$R_K(W) := \sup_{\varphi \in \mathcal{H}} |G_m(\varphi; W) - G_m^{(K)}(\varphi; W)|.$$

Since the networks in \mathcal{H} are uniformly bounded,

$$\sup_{\varphi \in \mathcal{H}} \ell_{m,j}(\varphi; W) \leq C (1 + |Y_v|^2 + \|Y_g^{\epsilon,m}\|^2 + \|Y_h^{\epsilon,m}\|^2).$$

Therefore,

$$\begin{aligned} \mathbb{E}R_K(W) &\leq \sum_{j=0}^{N-1} \Delta t_j \mathbb{E} \left[\sup_{\varphi \in \mathcal{H}} \ell_{m,j}(\varphi; W) \mathbf{1}_{\Omega_{K,m,j}^c} \right] \\ &\leq C \sum_{j=0}^{N-1} \Delta t_j \|1 + |Y_v|^2 + \|Y_g^{\epsilon,m}\|^2 + \|Y_h^{\epsilon,m}\|^2\|_{L^2} \mathbb{P}(\Omega_{K,m,j}^c)^{1/2}. \end{aligned}$$

It follows from the moment bound (C.21), the moments of X_{t_j} , the truncation tail (C.22), and the identity $\sum_{j=0}^{N-1} \Delta t_j = T$ that

$$\mathbb{E}R_K(W) \leq C \exp(-C(\log(1+K))^2).$$

Markov's inequality therefore gives, with probability at least $1 - \delta$,

$$\frac{1}{B} \sum_{i=1}^B R_K(W_i) + \mathbb{E}R_K(W) \leq C\delta^{-1} \exp(-C(\log(1+K))^2).$$

Choosing $K = \exp(C\sqrt{\log B + \log(1/\delta)})$, we obtain

$$\begin{aligned} &\sup_{\varphi \in \mathcal{H}} \left| \tilde{L}_m^{(B)}(\varphi) - L_m^{\mathcal{G}}(\varphi) \right| \\ &\leq C \left(1 + \sqrt{\log(1/\delta)}\right) \frac{\mathcal{P}(d)}{B^{1/2}} \exp\left(C\sqrt{\log B + \log(1/\delta)}\right). \end{aligned} \tag{C.23}$$

Combining (C.14), (C.15), (C.16), (C.17), and (C.23) proves the strong-simulator bound.

C.8 Proof of Theorem 10

The conclusion follows by taking $\delta = \delta/n$ in Theorem 9, and combining with the conclusion in Theorem 6.

References

K. Balasubramanian and S. Ghadimi. Zeroth-Order Nonconvex Stochastic Optimization: Handling Constraints, High Dimensionality, and Saddle Points. *Foundations of Computational Mathematics*, 22(1):35–76, Feb. 2022.

- D. Baños. The Bismut-Elworthy-Li formula for mean-field stochastic differential equations. *Annales de L'Institut Henri Poincaré Section (B) Probability and Statistics*, 54(1):220–233, 2018.
- A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1):115–133, 1994.
- P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *The Annals of Statistics*, 33(4), Aug. 2005.
- C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *Discrete and Continuous Dynamical Systems-B*, 28(6):3697–3746, 2023.
- D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.
- A. Bonito, R. DeVore, G. Petrova, and J. W. Siegel. Convergence and error control of consistent PINNs for elliptic PDEs. *IMA Journal of Numerical Analysis*, 46(1):90–148, 2026.
- W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, 2017.
- N. Doumèche, G. Biau, and C. Boyer. On the convergence of PINNs. *Bernoulli*, 31(3):2127–2151, 2025.
- K. Dzhaparidze and H. Van Zanten. On Bernstein-type inequalities for martingales. *Stochastic Processes and their Applications*, 93(1):109–117, May 2001.
- W. E and B. Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.
- W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high dimensional parabolic partial differential equations and backward stochastic differential equations. *Communication in Mathematical Statistics*, 5(4):349–380, 2017.
- S. Fischer and I. Steinwart. Sobolev norm learning rates for regularized least-squares algorithms. *Journal of Machine Learning Research*, 21(205):1–38, 2020.
- W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions*. Springer, 2006.
- D. Gazoulis, I. Gkanis, and C. G. Makridakis. On the stability and convergence of physics informed neural networks. *IMA Journal of Numerical Analysis*, page draf090, 2025.
- M. Germain, H. Pham, and X. Warin. Neural networks-based algorithms for stochastic control and PDEs in finance. In C.-A. Lehalle and A. Capponi, editors, *Machine Learning and Data Sciences for Financial Markets: A Guide to Contemporary Practices*. Cambridge University Press, 2023.

- P. Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53. Springer, 2004.
- J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115:8505–8510, 2018.
- J. Han, W. Hu, J. Long, and Y. Zhao. Deep Picard iteration for high-dimensional nonlinear PDEs. *SIAM Journal of Scientific Computing*, 48:1–24, 2026.
- D. He, S. Li, W. Shi, X. Gao, J. Zhang, J. Bian, L. Wang, and T.-Y. Liu. Learning physics-informed neural networks without stacked back-propagation. In *International Conference on Artificial Intelligence and Statistics*, pages 3034–3047. PMLR, 2023.
- K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5): 551–560, 1990.
- Z. Hu, Z. Yang, Y. Wang, G. E. Karniadakis, and K. Kawaguchi. Bias-variance trade-off in physics-informed neural networks with randomized smoothing for high-dimensional PDEs. *SIAM Journal on Scientific Computing*, 47(4):C846–C872, 2025.
- B. Hugel and A. Savine. Differential machine learning. *SSRN 3591734*, 2020.
- C. Huré, H. Pham, and X. Warin. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 89:1547–1579, 2020.
- Y. Jia and X. Y. Zhou. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(275):1–50, 2022a.
- Y. Jia and X. Y. Zhou. Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research*, 23(154): 1–55, 2022b.
- Y. Jia and X. Y. Zhou. q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.
- A. Jiao, H. He, R. Ranade, J. Pathak, and L. Lu. One-shot learning for solution operators of partial differential equations. *arXiv preprint arXiv:2104.05512*, 2021.
- Y. Jiao, Y. Lai, Y. Lo, Y. Wang, and Y. Yang. Error analysis of deep Ritz methods for elliptic equations. *Analysis and Applications*, 22(01):57–87, 2024.
- I. Karatzas and S. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 2014.
- Y.-J. Kim and C. Gu. Smoothing spline gaussian regression: More scalable computation via efficient approximation. *Journal of The Royal Statistical Society Series B: Statistical Methodology*, 66(2):337–356, 2004.
- H. Kunita. *Stochastic Flows and Jump-Diffusions*, volume 92 of *Probability Theory and Stochastic Modelling*. Springer Singapore, Singapore, 2019.

- J. H. Lagergren, J. T. Nardini, G. Michael Lavigne, E. M. Rutter, and K. B. Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2234), 2020.
- W. Lefebvre, G. Loeper, and H. Pham. Differential learning methods for solving fully nonlinear PDEs. *Digital Finance*, 51(1):183–2229, 2023.
- Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deepnet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021a.
- Y. Lu, J. Lu, and M. Wang. A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations. In *Conference on Learning Theory*, pages 3196–3241. PMLR, 2021b.
- Y. Lu, H. Chen, J. Lu, L. Ying, and J. Blanchet. Machine learning for elliptic PDEs: Fast rate generalization bound, neural scaling law and minimax optimality. In *International Conference on Learning Representations*, 2022.
- J. Ma and J. Zhang. Representation theorems for backward stochastic differential equations. *The Annals of Applied Probability*, 12(4):1390–1418, 2002.
- W. Mou. Statistical guarantees for continuous-time policy evaluation: Blessing of ellipticity and new tradeoffs. *arXiv preprint arXiv:2502.04297*, 2025.
- W. Mou and Y. Zhu. On Bellman equations for continuous-time policy evaluation: high-order discretization and function approximation. *SIAM Journal on Mathematics of Data Science*, 2025.
- Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.
- G. A. Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*, volume 60 of *Texts in Applied Mathematics*. Springer New York, New York, NY, 2014.
- S. Peng. A nonlinear feynman-kac formula and applications. In *Proceedings of Symposium of System Sciences and Control Theory*, pages 173–184. World Scientific, 1992.
- H. Pham. *Continuous-Time Stochastic Control and Optimization with Financial Applications*, volume 61. Springer Science & Business Media, 2009.
- P. E. Protter. *Stochastic Integration and Differential Equations*, volume 21 of *Stochastic Modelling and Applied Probability*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

- M. Raissi, P. Perdikarais, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.
- S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.
- Z. Shi, Z. Hu, M. Lin, and K. Kawaguchi. Stochastic Taylor derivative estimator: Efficient amortization for arbitrary differential operators. *Advances in Neural Information Processing Systems*, 37:122316–122353, 2024.
- J. W. Siegel. Optimal approximation rates for deep ReLU neural networks on Sobolev and Besov spaces. *Journal of Machine Learning Research*, 24(357):1–52, 2023.
- J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.
- G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.
- M. J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge University Press, 1 edition, Feb. 2019.
- C. Wang, S. Li, D. He, and L. Wang. Is L^2 physics informed loss always suitable for training physics informed neural network? *Advances in Neural Information Processing Systems*, 35:8278–8290, 2022.
- H. Wang, T. Zariphopoulou, and X. Y. Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198):1–34, 2020.
- J. Yong and X. Y. Zhou. *Stochastic Controls: Hamiltonian Systems and HJB Equations*, volume 43. Springer Science & Business Media, 1999.
- W. Zhang, Z. Hu, W. Cai, and G. Karniadaki. Deep neural networks for solving high-dimensional parabolic partial differential equations. *arXiv:2601.13256*, 2026.
- M. Zhou, J. Han, and J. Lu. 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, 2021.
- Y. Zhu, Y. Zhang, and H. Zhang. Optimal-PhiBE: A PDE-based model-free framework for continuous-time reinforcement learning. *arXiv preprint arXiv:2506.05208*, 2025.