Interpolating Between BSDEs and PINNs: Deep Learning for Elliptic and Parabolic Boundary Value Problems

Nikolas Nüsken * 1 and Lorenz Richter † 2,3,4

^{1}Department of Mathematics, King's College London, UK.
^{2}dida Datenschmiede GmbH, 10827 Berlin, Germany.
^{3}Institute of Mathematics, Freie Universität Berlin, 14195 Berlin, Germany.
^{4}Zuse Institute Berlin, 14195 Berlin, Germany.

Abstract. Solving high-dimensional partial differential equations is a recurrent challenge in economics, science and engineering. In recent years, a great number of computational approaches have been developed, most of them relying on a combination of Monte Carlo sampling and deep learning based approximation. For elliptic and parabolic problems, existing methods can broadly be classified into those resting on reformulations in terms of backward stochastic differential equations (BSDEs) and those aiming to minimize a regression-type L^2-error (physics-informed neural networks, PINNs). In this paper, we review the literature and suggest a methodology based on the novel diffusion loss that interpolates between BSDEs and PINNs. Our contribution opens the door towards a unified understanding of numerical approaches for high-dimensional PDEs, as well as for implementations that combine the strengths of BSDEs and PINNs. The diffusion loss furthermore bears close similarities to (least squares) temporal difference objectives found in reinforcement learning. We also discuss eigenvalue problems and perform extensive numerical studies, including calculations of the ground state for nonlinear Schrödinger operators and committor functions relevant in molecular dynamics.

1 Introduction

In this article, we consider approaches towards solving high-dimensional partial differential equations (PDEs) that are based on minimizing appropriate loss functions in the spirit of machine learning. For example, we aim at identifying approximate solutions to nonlinear parabolic boundary value problems of the form

$$(\partial_t + L)V(x,t) + h\big(x,t,V(x,t),\sigma^\top \nabla V(x,t)\big) = 0, \qquad (x,t) \in \Omega \times [0,T), \qquad (1.1a)$$

$$V(x,T) = f(x), \qquad x \in \Omega, \qquad (1.1b)$$

$$V(x,t) = g(x,t), \qquad (x,t) \in \partial\Omega \times [0,T], \qquad (1.1c)$$

on a spatial domain $\Omega \subset \mathbb{R}^d$ and time interval $[0,T]$, where $h \in C(\mathbb{R}^d \times [0,T] \times \mathbb{R} \times \mathbb{R}^d, \mathbb{R})$ specifies the nonlinearity, and $f \in C(\mathbb{R}^d, \mathbb{R})$ as well as $g \in C(\mathbb{R}^d \times [0,T], \mathbb{R})$ are given

---

*nikolas.nusken@kcl.ac.uk.
†Corresponding author. lorenz.richter@dida.do.

functions defining the terminal and boundary conditions. Moreover,

$$L = \frac{1}{2} \sum_{i,j=1}^{d} (\sigma\sigma^\top)_{ij}(x,t)\partial_{x_i}\partial_{x_j} + \sum_{i=1}^{d} b_i(x,t)\partial_{x_i} \qquad (1.2)$$

is an elliptic differential operator including the coefficient functions $b \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ and $\sigma \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^{d\times d})$, with σ assumed to be non-degenerate. We will later make use of the fact that L is the infinitesimal generator of the diffusion process defined by the stochastic differential equation (SDE)

$$\mathrm{d}X_s = b(X_s,s)\mathrm{d}s + \sigma(X_s,s)\mathrm{d}W_s, \qquad (1.3)$$

where W_s is a standard d-dimensional Brownian motion. Our approach exploiting the connection between (1.1) and (1.3) extends almost effortlessly to a wide range of nonlinear elliptic PDEs (see Section 4.1) and eigenvalue problems (see Section 4.2). We also note that the terminal condition (1.1b) can be replaced by an initial condition without loss of generality, using a time reversal transformation. Similarly, we may replace the Dirichlet boundary condition (1.1c) by its Neumann counterpart, that is, constraining the normal derivative $\partial_{\vec{n}} V$ on $\partial\Omega \times [0,T]$ in (1.1c).

A notorious challenge that appears in the numerical treatment of PDEs is the curse of dimensionality, suggesting that the computational complexity increases exponentially in the dimension of the state space. In recent years, however, multiple numerical [19, 36, 71] as well as theoretical studies [26, 41] have indicated that a combination of Monte Carlo methods and neural networks offers a promising way to overcome this problem. This paper centers around two strategies that allow for solving quite general nonlinear PDEs:

- **PINNs (Physics-Informed Neural Networks)** [71], also known under the name DGM (deep Galerkin method) [77] directly minimize the misfit between the left-hand sides and right-hand sides of (1.1), evaluated at appropriately chosen (random) points in the space-time domain $\Omega \times [0,T]$ and its boundary.

- **Deep BSDEs (Backward Stochastic Differential Equations)** [19] rely on a reformulation of (1.1) in terms of stochastic forward-backward dynamics, explicitly making use of the connection between the PDE (1.1) and the SDE (1.3). Deep BSDEs minimize the misfit in the terminal condition associated to the backward SDE.

We review those approaches (see Section 2) and – motivated by Itô's formula – introduce a novel optimization objective, called the diffusion loss $\mathcal{L}_{\mathrm{diffusion}}^{\mathfrak{t}}$ (see Section 3). Importantly, our construction depends on the auxiliary time parameter $\mathfrak{t} \in (0,\infty)$, allowing us to recover PINNs in the limit $\mathfrak{t} \to 0$ and deep BSDEs in the limit $\mathfrak{t} \to \infty$

$$\text{PINNs} \xleftarrow{\ \mathfrak{t}\to 0\ } \mathcal{L}_{\mathrm{diffusion}}^{\mathfrak{t}} \xrightarrow{\ \mathfrak{t}\to\infty\ } \text{deep BSDEs}$$

As will become clear below, PINNs may therefore be thought of as accumulating derivative information on the PDE solution locally (in time), whereas deep BSDEs constitute global approximation schemes (relying on entire trajectories). Moreover, the diffusion loss

is closely related to (least squares) temporal difference learning (see [78, Chapter 3] and Remark 3.2 below).[1]

The diffusion loss provides an interpolation between seemingly quite distinct methods. Besides this theoretical insight, we show experimentally that an appropriate choice of t can lead to a computationally favorable blending of PINNs and deep BSDEs, combining the advantages of both methods. In particular, the diffusion loss (with t chosen to be of moderate size) appears to perform well in scenarios where the domain Ω has a (possibly complex) boundary and/or the PDE under consideration contains a large number of second order derivatives (for instance, when σ is not sparse). We will discuss the trade-offs involved in Section 5.

The curse of dimensionality, BSDEs vs. PINNs. Traditional numerical methods for solving PDEs (such as finite difference and finite volume methods, see [1]) usually require a discretization of the domain Ω, incurring a computational cost that for a prescribed accuracy is expected to grow linearly in the number of grid points, and hence exponentially in the number of dimensions. The recently developed approaches towards beating this curse of dimensionality – BSDEs and PINNs alike[2] – replace the deterministic mesh by Monte Carlo sampling, in principle promising dimension-independent convergence rates [22]. Typically, the approximating function class is comprised of neural networks (expectantly providing adaptivity to low-dimensional latent structures [2]), although also tensor trains have shown to perform well [75]. In contrast to PINNs, methods based on BSDEs make essential use of the underlying diffusion (1.3) to generate the training data. According to our preliminary numerical experiments, it is not clear whether this use of structure indeed translates into computational benefits; we believe that additional research into this comparison is needed and likely to contribute both conceptually and practically to further advances in the field. The diffusion loss considered in this paper may be a first step in this direction, for a direct comparison between BSDEs and PINNs we refer to Tables 5.1(a) and 5.1(b) below.

Previous works. Attempts to approximate PDE solutions by combining Monte Carlo methods with neural networks date back to the 1990s, where some variants of residual minimizations have been suggested [42, 50–52, 79]. Recently, this idea gained popularity under the names physics-informed neural networks (PINNs, [71, 72]) and deep Galerkin methods (DGMs, [77]). Let us further refer to a comparable approach that has been suggested in [9], and, for the special case of solving the dynamic Schrödinger equation, to [16]. For theoretical analyses we shall for instance mention [59], which provides upper bounds on the generalization error, [17], which states an error analysis for PINNs in the approximation of Kolmogorov-type PDEs, [61], which investigates convergences depending on whether using exact or penalized boundary terms, and [82,83], which study convergence properties through the lens of neural tangent kernels. Some further numerical experiments have been conducted in [18,57] and multiple algorithmic improvements have been

---

[1]We would like to thank an anonymous referee for pointing out this connection.

[2]We note in passing that multilevel Picard approximations [7, 37, 39] represent another interesting and fairly different class of methods that however are beyond the scope of this paper.

suggested, e.g. in [81], which balances gradients during model training, [40], which considers adaptive activation functions to accelerate convergence, as well as [80], which investigates efficient weight-tuning.

BSDEs have first been introduced in the 1970s [11] and eventually studied more systematically in the 1990s [64]. For a comprehensive introduction elaborating on their connections to both elliptic and parabolic PDEs we refer for instance to [63]. Numerical attempts to exploit this connection aiming for approximations of PDE solutions have first been approached by backward-in-time iterations, originally relying on a set of basis functions and addressing parabolic equations on unbounded domains [13, 23, 63]. Those methods have been considered and further developed with neural networks in [3, 36] and endowed with tensor trains in [75]. A variational formulation termed deep BSDE has been first introduced in [19, 28], aiming at PDE solutions at a single point, with some variants following e.g. in [62, 70, 73]. For an analysis of the approximation error of the deep BSDE method we refer to [29] and for a fully nonlinear version based on second-order BSDEs to [5]. An extension to elliptic PDEs on bounded domains has been suggested in [48].

The diffusion loss as well as other methods that aim at solving high-dimensional PDEs can be related to well-known approaches in reinforcement learning and optimal control. In particular, the key definition (3.3a) below can be seen as a generalization of [87, Eq. (2.27)], derived in the context of (least squares) temporal difference learning [78, Chapter 3]. In contrast, the algorithms suggested in [31, 58] are in the spirit of Monte Carlo reinforcement techniques [78, Section 3.1.2], relying on an averaged rather than a pathwise construction principle for optimization objectives. We would also like to point out that introducing the artificial time parameter t is conceptually similar to the idea in [30] where the authors elevate an eigenvalue problem of elliptic type to a parabolic one using a related augmentation.

There are additional works aiming to solve linear PDEs specifically, mainly by exploiting the Feynman-Kac theorem and mostly considering parabolic equations [4, 10, 74]. For the special case of the Poisson equation, [24] considers an elliptic equation on a bounded domain. Linear PDEs often admit a variational formulation that suggests to minimize a certain energy functional – this connection has been used in [21, 46, 55] and we refer to [60] for some further analysis. Similar minimization strategies can be used when considering eigenvalue problems, where we mention [86] in the context of metastable diffusion processes. We also refer to [34, 43, 49, 67] for similar problems in quantum mechanics that often rest on particular neural network architectures. Nonlinear elliptic eigenvalue problems are addressed in [30, 67] by exploiting a connection to a parabolic PDE and a fixed point problem.

For rigorous results towards the capability of neural networks to overcome the curse of dimensionality we refer to [25, 38, 41], each analyzing certain special PDE cases. Adding to the methods referred to above, let us also mention [84] as an alternative approach exploiting weak PDEs formulations, as well as [54], which approximates operators by neural networks (however relying on training data from reference solutions), where a typical application is to map an initial condition to a solution of a PDE. For further references on approximating PDE solutions with neural networks we refer to the recent review articles [6, 12, 22, 44].

Outline of the article. In Section 2 we review PINN and BSDE based approaches towards solving high-dimensional (parabolic) PDEs. In Section 3 we introduce the diffusion loss, show its validity for solving PDEs of the form (1.1), and prove that it interpolates between the PINN and BSDE losses. Section 4 develops extensions of the proposed methodology to elliptic PDEs and eigenvalue problems. In Section 5, we discuss implementation details as well as some further modifications of the losses under consideration. In Section 6 we present numerical experiments, including a committor function example from molecular dynamics and a nonlinear eigenvalue problem motivated by quantum physics. Finally, Section 7 concludes the paper with a summary and outlook.

2 Variational formulations of boundary value problems

In this section we consider boundary value problems such as (1.1) in a variational formulation. That is, we aim at approximating the solution V with some function $\varphi \in \mathcal{F}$ by minimizing suitable loss functionals

$$\mathcal{L} : \mathcal{F} \to \mathbb{R}_{\geq 0}, \tag{2.1}$$

which are zero if and only if the boundary value problem is fulfilled,

$$\mathcal{L}(\varphi) = 0 \iff \varphi = V. \tag{2.2}$$

Here $\mathcal{F} \subset C^{2,1}(\Omega \times [0, T], \mathbb{R}) \cap C(\overline{\Omega} \times [0, T], \mathbb{R})$ refers to an appropriate function class, usually consisting of deep neural networks. With a loss function at hand we can apply gradient-descent type algorithms to minimize (estimator versions of) $\mathcal{L}$, keeping in mind that different choices of losses lead to different statistical and computational properties and therefore potentially to different convergence speeds and robustness behaviors [62].

Throughout, we will work under the following assumption:

Assumption 2.1. The following hold:

1. The domain Ω is either bounded with piecewise smooth boundary, or $\Omega = \mathbb{R}^d$.

2. The boundary value problem (1.1) admits a unique classical solution $V \in C^{2,1}(\Omega \times [0, T], \mathbb{R}) \cap C(\overline{\Omega} \times [0, T], \mathbb{R})$. Moreover, the gradient of V satisfies a polynomial growth condition in x, that is,

$$|\nabla V(x, t)| \leq C(1 + |x|^q), \quad (x, t) \in \Omega \times [0, T] \tag{2.3}$$

   for some $C, q > 0$.

3. Complemented with a deterministic initial condition, the SDE (1.3) admits a unique strong solution, globally in time.

We note that the second assumption is trivially satisfied in the case when Ω is bounded.

2.1 The PINN loss

Losses based on PDE residuals go back to [50, 51] and have gained recent popularity under the name physics-informed neural networks (PINNs, [71]) or deep Galerkin methods (DGMs, [77]). The idea is to minimize an appropriate L^2-error between the left- and right-hand sides of (1.1a)-(1.1c), replacing V by its approximation φ. The derivatives of φ are computed analytically or via automatic differentiation and the data on which φ is evaluated is distributed according to some prescribed probability measure (often a uniform distribution). A precise definition is as follows.

Definition 2.1 (PINN Loss). *Let $\varphi \in \mathcal{F}$. The PINN loss consists of three terms,*

$$\mathcal{L}_{\mathrm{PINN}}(\varphi) = \alpha_{\mathrm{int}}\mathcal{L}_{\mathrm{PINN,int}}(\varphi) + \alpha_{\mathrm{T}}\mathcal{L}_{\mathrm{PINN,T}}(\varphi) + \alpha_{\mathrm{b}}\mathcal{L}_{\mathrm{PINN,b}}(\varphi), \tag{2.4}$$

where

$$\mathcal{L}_{\mathrm{PINN,int}}(\varphi) = \mathbb{E}\left[\left((\partial_t + L)\varphi(X,t) + h\big(X,t,\varphi(X,t),\sigma^\top\nabla\varphi(X,t)\big)\right)^2\right], \tag{2.5a}$$

$$\mathcal{L}_{\mathrm{PINN,T}}(\varphi) = \mathbb{E}\left[\left(\varphi(X^{(T)},T) - f(X^{(T)})\right)^2\right], \tag{2.5b}$$

$$\mathcal{L}_{\mathrm{PINN,b}}(\varphi) = \mathbb{E}\left[\left(\varphi(X^{\mathrm{b}},t^{\mathrm{b}}) - g(X^{\mathrm{b}},t^{\mathrm{b}})\right)^2\right]. \tag{2.5c}$$

Here, $\alpha_{\mathrm{int}}, \alpha_{\mathrm{T}}, \alpha_{\mathrm{b}} > 0$ are suitable weights balancing the interior, terminal and boundary constraints, and $(X,t) \sim \nu_{\Omega\times[0,T]}$, $X^{(T)} \sim \nu_\Omega$ and $(X^{\mathrm{b}},t^{\mathrm{b}}) \sim \nu_{\partial\Omega\times[0,T]}$ are distributed according to probability measures $\nu_{\Omega\times[0,T]} \in \mathcal{P}(\Omega\times[0,T])$, $\nu_\Omega \in \mathcal{P}(\Omega)$ and $\nu_{\partial\Omega\times[0,T]} \in \mathcal{P}(\partial\Omega\times[0,T])$ that are fully supported on their respective domains.

While uniform distributions are canonical choices for $\nu_{\Omega\times[0,T]}$, ν_Ω and $\nu_{\partial\Omega\times[0,T]}$, further research might reveal promising (possibly adaptive) alternatives that focus the sampling on specific regions of interest.

Remark 2.1 (Generalizations)**.** Clearly, the loss contributions (2.5a)-(2.5c) represent in one-to-one correspondence the constraints in (1.1a)-(1.1c). By that principle, the PINN loss can straightforwardly be generalized to other types of PDEs, see [44] and references therein. Let us further already mention that choosing appropriate weights $\alpha_{\mathrm{int}}, \alpha_{\mathrm{T}}, \alpha_{\mathrm{b}} > 0$ is crucial for algorithmic performance, but not straightforward. We will elaborate on this aspect in Section 5.

Remark 2.2 (Unbounded Domains)**.** In the case when $\Omega = \mathbb{R}^d$, the boundary contribution (2.5c) becomes obsolete and we set $\alpha_{\mathrm{b}} = 0$. Note that in this scenario, it is impossible to choose $\nu_{\Omega\times[0,T]}$ and ν_Ω to be uniform, and a choice to prioritize certain regions of Ω needs to be made. Although in theory the loss function (2.4) still admits the solution to (1.1a) as its unique minimizer, the lack of uniform coverage will make it challenging to obtain accurate approximations on sets which have small probability under $\nu_{\Omega\times[0,T]}$ and ν_Ω. Solving PDEs on large (and, in principle, unbounded) domains using the PINN loss hence requires a commensurate computational budget to ensure sufficient sampling for the expectations in (2.5a)-(2.5c). Analogous remarks apply to the BSDE and diffusion losses introduced below.

2.2 The BSDE loss

The BSDE loss makes use of a stochastic representation of the boundary value problem (1.1) given by a backward stochastic differential equation (BSDE) that is rooted in the correspondence between the differential operator L defined in (1.2) and the stochastic process X defined in (1.3). Indeed, according to [63], the PDE (1.1) is related to the system

$$\mathrm{d}X_s = b(X_s, s)\,\mathrm{d}s + \sigma(X_s, s)\,\mathrm{d}W_s, \qquad X_{t_0} = x_{\text{init}}, \tag{2.6a}$$

$$\mathrm{d}Y_s = -h(X_s, s, Y_s, Z_s)\,\mathrm{d}s + Z_s \cdot \mathrm{d}W_s, \quad Y_{T \wedge \tau} = k(X_{T \wedge \tau}, T \wedge \tau), \tag{2.6b}$$

where $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ is the first exit time from Ω and k subsumes the boundary conditions,

$$k(x, t) = \begin{cases} f(x), & t = T, \quad x \in \Omega, \\ g(x, t), & t \leq T, \quad x \in \partial\Omega. \end{cases} \tag{2.7}$$

Owing to the fact that X in (2.6a) is constrained at initial time t_0 and Y in (2.6b) at final time T, the processes $(X_s)_{t_0 \leq s \leq T}$ and $(Y_s)_{t_0 \leq s \leq T}$ are referred to as forward and backward, respectively. Given appropriate growth and regularity conditions on the coefficients b, σ, h and k, Itô's formula implies that the backward processes satisfy

$$Y_s = V(X_s, s), \quad Z_s = \sigma^\top \nabla V(X_s, s), \tag{2.8}$$

that is, they provide the solution to (1.1) and its derivative along the trajectories of the forward process $(X_s)_{t_0 \leq s \leq T}$, see [85]. Aiming for the approximation $\varphi \approx V$, we substitute (2.8) into (2.6b) and replace V by φ to obtain

$$\widetilde{Y}_{T \wedge \tau}(\varphi) = \varphi(X_{t_0}, t_0) - \int_0^{T \wedge \tau} h\big(X_s, s, \varphi(X_s, s), \sigma^\top \nabla\varphi(X_s, s)\big)\,\mathrm{d}s$$

$$+ \int_0^{T \wedge \tau} \sigma^\top \nabla\varphi(X_s, s) \cdot \mathrm{d}W_s. \tag{2.9}$$

Our notation $\widetilde{Y}_{T \wedge \tau}(\varphi)$ emphasizes the distinction from $Y_{T \wedge \tau}$ (which refers to the solution of (2.6)) and highlights the dependence on the particular choice $\varphi \in \mathcal{F}$. The key idea is now to penalize deviations from the terminal condition in (2.6b) via the loss

$$\mathcal{L}(\varphi) = \mathbb{E}\left[\left(k(X_{T \wedge \tau}, T \wedge \tau) - \widetilde{Y}_{T \wedge \tau}(\varphi)\right)^2\right], \tag{2.10}$$

see [28]. We summarize this construction as follows.

Definition 2.2 (BSDE Loss). *Let $\varphi \in \mathcal{F}$. The BSDE loss is defined as*

$$\mathcal{L}_{\text{BSDE}}(\varphi) = \mathbb{E}\Bigg[\Bigg(f(X_{\tau \wedge T})\mathbb{1}_{\{\tau \wedge T = T\}} + g(X_{\tau \wedge T}, \tau \wedge T)\mathbb{1}_{\{\tau \wedge T = \tau\}}$$

$$- \varphi(X_{t_0}, t_0) - \int_{t_0}^{\tau \wedge T} \sigma^\top \nabla\varphi(X_s, s) \cdot \mathrm{d}W_s$$

$$+ \int_{t_0}^{\tau \wedge T} h\big(X_s, s, \varphi(X_s, s), \sigma^\top \nabla\varphi(X_s, s)\big)\,\mathrm{d}s\Bigg)^2\Bigg], \tag{2.11}$$

where $(X_t)_{t_0 \leq t \leq \tau \wedge T}$ is a solution to (1.3) and $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ is the first exit time from Ω. Furthermore, the initial condition (X_0, t_0) is distributed according to a prescribed probability measure $\nu_{\Omega \times [0,T]}$ with full support on $\Omega \times [0, T]$.

Remark 2.3 (BSDE versus PINN). In contrast to the PINN loss (2.4), the BSDE loss (2.11) does not rely on a judicious tuning of the weights $\alpha_{\text{int}}, \alpha_{\text{T}}, \alpha_{\text{b}}$. On the other hand, implementations based on the BSDE loss face the challenge of simulating the hitting times $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ efficiently and accurately. We shall elaborate on this aspect in Section 5.1.

Remark 2.4 (Neumann and Periodic Boundary Conditions). Instead of the Dirichlet boundary condition (1.1c), Neumann or periodic boundary conditions may be considered, and the generalization of the PINN loss (as well as the diffusion loss introduced below) is straightforward. In the case of periodic boundary conditions, for instance, (2.5c) may be replaced by

$$\mathcal{L}_{\text{b}}(\varphi) = \mathbb{E}\left[\left(\varphi(X^b) - \varphi(\overline{X}^b)\right)^2\right] + \mathbb{E}\left[\left|\nabla\varphi(X^b) - \nabla\varphi(\overline{X}^b)\right|^2\right], \qquad (2.12)$$

where $X^b \sim \nu_{\partial\Omega}$, and $\overline{X}^b$ refers to the reflected/periodic counterpart. For the BSDE loss one could for instance incorporate homogeneous Neumann conditions by reflecting the stochastic process at the boundary [31], however, straightforward extensions to more general boundary conditions seem to be not available. We refer to [65], which might provide a starting point for the construction of further related BSDE based methods.

Remark 2.5 (Relationship to Earlier Work). The idea of approximating solutions to PDEs by solving BSDEs has been studied extensively [13, 23, 63], where first approaches were regression based, relying on iterations backwards in time. These ideas do not seem to be straightforwardly applicable to the case when Ω is bounded, as the trajectories of the forward process (2.6a) are not all of the same length (but see [14, 32]). A global variational strategy using neural networks has first been introduced in [19], where however in contrast to Definition 2.2, the initial condition (X_0, t_0) is deterministic (and fixed) and only parabolic problems on $\Omega = \mathbb{R}^d$ are considered. Moreover, slightly different choices for the approximations are chosen, namely V is only approximated at $t_0 = 0$ and ∇V instead of V is learnt by one neural network per time point (instead of using only one neural network with t as an additional input).

3   The diffusion loss

In this section we introduce a novel loss that interpolates between the PINN and BSDE losses from Section 2 using an auxiliary time parameter $\mathfrak{t} \in (0, \infty)$. As for the BSDE loss, the connection between the SDE (1.3) and its infinitesimal generator (1.2) plays a major role: Itô's formula

$$V(X_T, T) - V(X_0, 0) = \int_0^T (\partial_s + L) V(X_s, s) \, ds + \int_0^T \sigma^\top \nabla V(X_s, s) \cdot dW_s \qquad (3.1)$$

motivates the following variational formulation of the boundary value problem (1.1).

Definition 3.1 (Diffusion Loss). *Let $\varphi \in \mathcal{F}$ and $\mathfrak{t} \in (0, \infty)$. The diffusion loss consists of three terms,*

$$\mathcal{L}^{\mathfrak{t}}_{\text{diffusion}}(\varphi) = \alpha_{\text{int}} \mathcal{L}^{\mathfrak{t}}_{\text{diffusion,int}}(\varphi) + \alpha_{\text{T}} \mathcal{L}^{\mathfrak{t}}_{\text{diffusion,T}}(\varphi) + \alpha_{\text{b}} \mathcal{L}^{\mathfrak{t}}_{\text{diffusion,b}}(\varphi), \qquad (3.2)$$

where

$$\mathcal{L}^{\mathfrak{t}}_{\text{diffusion,int}}(\varphi) = \mathbb{E}\left[\left(\varphi(X_{\mathcal{T}}, \mathcal{T}) - \varphi(X_{t_0}, t_0) - \int_{t_0}^{\mathcal{T}} \sigma^{\top} \nabla \varphi(X_s, s) \cdot \mathrm{d}W_s \right.\right.$$
$$\left.\left. + \int_{t_0}^{\mathcal{T}} h\big(X_s, s, \varphi(X_s, s), \sigma^{\top} \nabla \varphi(X_s, s)\big) \, \mathrm{d}s\right)^2\right], \qquad (3.3a)$$

$$\mathcal{L}^{\mathfrak{t}}_{\text{diffusion,T}}(\varphi) = \mathbb{E}\left[\left(\varphi(X^{(T)}, T) - f(X^{(T)})\right)^2\right], \qquad (3.3b)$$

$$\mathcal{L}^{\mathfrak{t}}_{\text{diffusion,b}}(\varphi) = \mathbb{E}\left[\left(\varphi(X^{\text{b}}, t^{\text{b}}) - g(X^{\text{b}}, t^{\text{b}})\right)^2\right] \qquad (3.3c)$$

encode the constraints (1.1a)-(1.1c) balanced by the weights $\alpha_{\text{int}}, \alpha_{\text{T}}, \alpha_{\text{b}} > 0$, and the process $(X_t)_{t_0 \leq t \leq \mathcal{T}}$ is a solution to (1.3) with initial condition $(X_0, t_0) \sim \nu_{\Omega \times [0,T]}$ and maximal trajectory length $\mathfrak{t} > 0$. The stopping time $\mathcal{T} := (t_0 + \mathfrak{t}) \wedge \tau \wedge T$ is a shorthand notation, referring to the (random) final time associated to a realization of the path X as it either hits the parabolic boundary $\partial \Omega \times \{T\}$ or reaches the maximal time $t_0 + \mathfrak{t}$. As in Definition 2.1, $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ is the exit time from Ω, and $(X^{\text{b}}, t^{\text{b}}) \sim \nu_{\partial \Omega \times [0,T]}$, $X^{(T)} \sim \nu_{\Omega}$ are distributed according to probability measures that are fully supported on their respective domains.

Remark 3.1 (Comparison to the BSDE and PINN Losses). In contrast to the PINN loss from Definition 2.1, the data inside the domain Ω is not sampled according to a prescribed probability measure ν_{Ω}, but along trajectories of the diffusion (1.3). Consequently, second derivatives of φ do not have to be computed explicitly, but are approximated using the driving Brownian motion (and – implicitly – Itô's formula). A main difference to the BSDE loss from Definition 2.2 is that the simulated trajectories have a maximal length $\mathfrak{t}$, which might be beneficial computationally if the final time T or the exit time τ is large (with high probability). Additionally, the sampling of extra boundary data circumvents the problem of accurately simulating those exit times (see Remark 2.3). Both aspects will be further discussed in Section 5. We refer to Fig. 3.1 for a graphical illustration of the data required to compute the three losses.

Remark 3.2 (Reinforcement Learning). The form of (3.3a) can directly be related to (least squares) temporal difference objectives [78, Section 3.2], the PDE (1.1a) being a continuous-time analog to the Bellman equation that governs optimal control. We refer to [87, Section 2] for a lucid exposition of the relevant ideas in the context of (elliptic) PDEs. Interestingly, however, the background on control and reinforcement learning (although providing further intuition) is not strictly necessary, as the diffusion loss applies to the general parabolic PDE in (1.1a).[3] The diffusion loss can hence be thought of as a generalization of the approach in [87], see in particular Eq. (2.27) therein, with $\gamma = 0$.
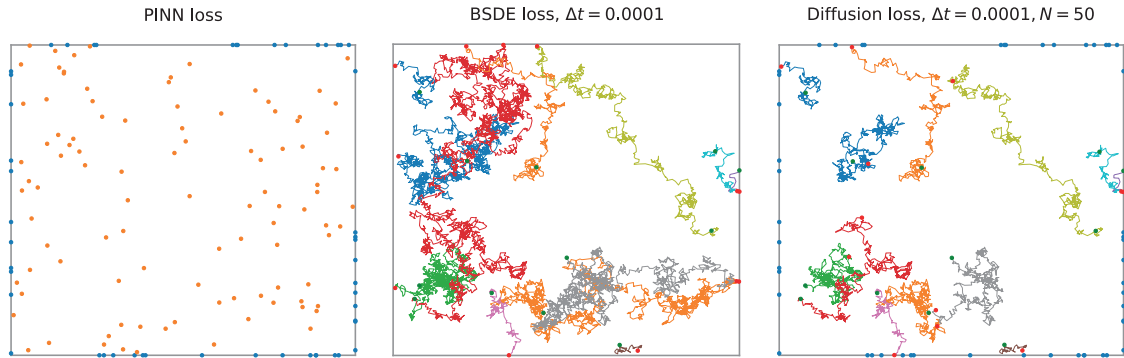
Figure 3.1: We illustrate the training data used for the three losses inside the unit square $\Omega = (0,1)^2$. The PINN loss in the left panel takes i.i.d. data points that are sampled from prescribed probability distributions in the domain Ω and on the boundary $\partial\Omega$ (in this case from uniform distributions). The BSDE loss (middle panel) uses trajectories associated to the SDE (1.3) that are started at random points X_0 (green points) and run until they hit the boundary (red points). The trajectories for the diffusion loss have a maximal length t and therefore frequently start and end inside Ω, as displayed in the right panel. The blue points for the PINN and diffusion losses indicate the additionally sampled boundary data.

The following proposition shows that the loss $\mathcal{L}^{\text{t}}_{\text{diffusion}}$ is indeed suitable for the boundary value problem (1.1).

Proposition 3.1. *Consider the diffusion loss as defined in (3.2), and assume that b and σ are globally Lipschitz continuous in x, uniformly in $t \in [0, T]$. Furthermore, assume the following Lipschitz and boundedness conditions on f, g and h:*

$$|f(x)| \leq C(1 + |x|^p),$$
$$|g(x,t)| \leq C(1 + |x|^p),$$
$$|h(t,x,y,z)| \leq C(1 + |x|^p + |y| + |z|),$$
$$|h(t,x,y,z) - h(t,x,y',z)| \leq C|y - y'|,$$
$$|h(t,x,y,z) - h(t,x,y,z')| \leq C|z - z'|$$

for appropriate constants $C, p \geq 0$ and all $x, y, z \in \Omega$, $t \in [0, T]$. Finally, assume that Assumption 2.1 is satisfied. Then for $\varphi \in \mathcal{F}$ the following are equivalent:

1. *The diffusion loss vanishes on φ,*

$$\mathcal{L}^{\text{t}}_{\text{diffusion}}(\varphi) = 0. \tag{3.4}$$

2. *φ fulfills the boundary value problem (1.1).*

Proof. Denoting by X_s the unique strong solution to (1.3), an application of Itô's lemma to $\varphi(X_s, s)$ yields

$$\varphi(X_{\mathcal{T}}, \mathcal{T}) = \varphi(X_{t_0}, t_0) + \int_{t_0}^{\mathcal{T}} (\partial_s + L)\varphi(X_s, s)\, \mathrm{d}s + \int_{t_0}^{\mathcal{T}} \sigma^\top \nabla \varphi(X_s, s) \cdot \mathrm{d}W_s, \tag{3.5}$$

---

[3]This PDE contains control-related Hamilton-Jacobi-Bellman PDEs as special cases, see, for instance [68].

almost surely. Assuming that φ fulfills the PDE (1.1a), it follows from the definition in (3.3a) that $\mathcal{L}^t_{\text{diffusion,int}}(\varphi) = 0$. Similarly, the boundary conditions (1.1b) and (1.1c) imply that $\mathcal{L}^t_{\text{diffusion,T}}(\varphi) = \mathcal{L}^t_{\text{diffusion,b}}(\varphi) = 0$. Consequently, we see that $\mathcal{L}^t_{\text{diffusion}}(\varphi) = 0$.

For the converse direction, observe that $\mathcal{L}^t_{\text{diffusion}}(\varphi) = 0$ implies that

$$
\varphi(X_{\mathcal{T}}, \mathcal{T}) = \varphi(X_{t_0}, t_0) + \int_{t_0}^{\mathcal{T}} \sigma^\top \nabla \varphi(X_s, s) \cdot \mathrm{d}W_s
$$
$$
- \int_{t_0}^{\mathcal{T}} h\big(X_s, s, \varphi(X_s, s), \sigma^\top \nabla \varphi(X_s, s)\big) \, \mathrm{d}s, \tag{3.6}
$$

almost surely, and that the same holds with φ replaced by V. We proceed by defining the processes $\widetilde{Y}_s := \varphi(X_s, s)$ and $\widetilde{Z}_s := \sigma^\top \nabla \varphi(X_s, s)$, as well as $Y_s := V(X_s, s)$ and $Z_s := \sigma^\top \nabla V(X_s, s)$. By the assumptions on φ, b and σ, the processes $Y, Z, \widetilde{Y}$ and $\widetilde{Z}$ are progressively measurable with respect to the filtration generated by $(W_t)_{t \geq 0}$ and moreover square-integrable. Furthermore, the relation (3.6) shows that the pairs (Y, Z) and $(\widetilde{Y}, \widetilde{Z})$ satisfy a BSDE with terminal condition $\xi := \varphi(X_{\mathcal{T}}, \mathcal{T})$ on the random time interval $[t_0, \mathcal{T}]$. Well-posedness of the BSDE (see [63, Theorems 1.2 and 3.2]) implies that $Y = \widetilde{Y}$ and $Z = \widetilde{Z}$, almost surely. Conditional on t_0 and X_{t_0}, we also have $V(X_{t_0}, t_0) = Y^{X_{t_0}, t_0} = \widetilde{Y}^{X_{t_0}, t_0} = \varphi(X_{t_0}, t_0)$, where the superscripts denote conditioning on the initial time t_0 and corresponding initial condition X_{t_0}, see [63, Theorems 2.4 and 4.3]. Hence, we conclude that $\varphi = V$, $\nu_\Omega \times [0, T]$-almost surely, and the result follows from the continuity of φ and V and the assumption that $\nu_{\Omega \times [0,T]}$ has full support. $\square$

We have noted before that the diffusion loss combines aspects from the BSDE and PINN losses. In fact, it turns out that the diffusion loss can be interpreted as a specific type of interpolation between the two. The following proposition makes this observation precise.

Proposition 3.2 (Relation of the Diffusion Loss to the PINN and BSDE Losses)**.** *Let $\varphi \in \mathcal{F}$. Assuming that the measures $\nu_{\Omega \times [0,T]}$ in Definitions 2.1 and 3.1 coincide, we have that*

$$
\frac{\mathcal{L}^t_{\text{diffusion,int}}(\varphi)}{t^2} \to \mathcal{L}_{\text{PINN,int}}(\varphi) \tag{3.7}
$$

as $t \to 0$. Moreover, if $\nu_{\Omega \times [0,T]}$ refers to the same measure in Definitions 2.1 and 2.2, then

$$
\mathcal{L}^t_{\text{diffusion,int}}(\varphi) \to \mathcal{L}_{\text{BSDE}}(\varphi) \tag{3.8}
$$

as $t \to \infty$.

Proof. Itô's formula shows that $\mathcal{L}^t_{\text{diffusion,int}}$ can be expressed as

$$
\mathcal{L}^t_{\text{diffusion,int}}(\varphi) = \mathbb{E}\Bigg[\Bigg(\int_{t_0}^{\mathcal{T}} (\partial_s + L)\varphi(X_s, s) \, \mathrm{d}s
$$
$$
+ \int_{t_0}^{\mathcal{T}} h\big(X_s, s, \varphi(X_s, s), \sigma^\top \nabla \varphi(X_s, s)\big) \, \mathrm{d}s\Bigg)^2\Bigg], \tag{3.9}
$$

which implies the limit (3.7) by dominated convergence, noting that $\mathcal{T} \to t_0$ as $\mathfrak{t} \to 0$, almost surely. The relation (3.8) follows immediately from the definition of $\mathcal{L}_{\mathrm{BSDE}}$ by noting that $\mathcal{T} \to \tau \wedge T$ as $\mathfrak{t} \to \infty$, almost surely. $\qquad\square$

4 Extensions to elliptic PDEs and eigenvalue problems

In this section we show that the ideas reviewed and developed in the previous sections in the context of the parabolic boundary value problem (1.1) can straightforwardly be extended to the treatment of elliptic PDEs and certain eigenvalue problems (and in fact, the diffusion loss for parabolic problems could have been derived using [87, Section 2.2.3] as a starting point). To begin with, it is helpful to notice that the boundary value problem (1.1) can be written in the following slightly more abstract form.

Remark 4.1 (Compact Notation). Consider the operator $\mathcal{A} := \partial_t + L$, the space-time domain $\Omega_{xt} := \Omega \times [0, T)$ with (forward) boundary[4] $\overrightarrow{\partial} \Omega_{xt} := \Omega \times \{T\} \cup \partial\Omega \times [0, T]$, and the augmented variable $z = (x, t)^\top \in \Omega \times [0, T]$. Then problem (1.1) can be presented as

$$\mathcal{A}V(z) + h\big(z, V(z), \sigma^\top \nabla_x V(z)\big) = 0, \quad z \in \Omega_{xt}, \tag{4.1a}$$

$$V(z) = k(z), \qquad\qquad z \in \overrightarrow{\partial} \Omega_{xt} \tag{4.1b}$$

with k defined as in (2.7). Relying on (4.1), we can equivalently define the BSDE loss as

$$\mathcal{L}_{\mathrm{BSDE}}(\varphi) = \mathbb{E}\Bigg[\bigg(k(X_{\tau_{xt}}, \tau_{xt}) - \varphi(X_{t_0}, t_0) - \int_{t_0}^{\tau_{xt}} \sigma^\top \nabla\varphi(X_s, s) \cdot \mathrm{d}W_s$$

$$+ \int_{t_0}^{\tau_{xt}} h\big(X_s, s, \varphi(X_s, s), \sigma^\top \nabla\varphi(X_s, s)\big) \, \mathrm{d}s\bigg)^2\Bigg], \tag{4.2}$$

where $\tau_{xt} = \inf\{t > 0 : X_t \notin \Omega_{xt}\}$ is the first exit time from Ω_{xt}. The PINN and diffusion losses can similarly be rewritten in terms of the space-time domain Ω_{xt} and exit time τ_{xt}.

4.1 Elliptic boundary value problems

Removing the time dependence from the solution (and from the coefficients b and σ determining L) we obtain the elliptic boundary value problem

$$LV(x) + h\big(x, V(x), \sigma^\top \nabla V(x)\big) = 0, \quad x \in \Omega, \tag{4.3a}$$

$$V(x) = g(x), \qquad\qquad x \in \partial\Omega \tag{4.3b}$$

with the nonlinearity $h \in C(\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d, \mathbb{R})$. In analogy to (2.6), the corresponding backward equation is given by

$$\mathrm{d}Y_s = -h(X_s, Y_s, Z_s) \, \mathrm{d}s + Z_s \cdot \mathrm{d}W_s, \quad Y_\tau = g(X_\tau), \tag{4.4}$$

---

[4]We notice in passing that the (ordinary) topological boundary of Ω_{xt} is given by $\partial\Omega_{xt} = \Omega \times \{0, T\} \cup \partial\Omega \times [0, T]$, so that $\partial\Omega_{xt} = \overrightarrow{\partial}\Omega_{xt} \cup \{0\} \times \Omega$.

where $\tau = \{t > 0 : X_t \notin \Omega\}$ is the first exit time from Ω. Given suitable boundedness and regularity assumptions on h and assuming that τ is almost surely finite, one can show existence and uniqueness of solutions Y and Z, which, as before, represent the solution V and its gradient along trajectories of the forward process [63, Theorem 4.6].

Therefore, the BSDE, PINN and diffusion losses can be applied to (4.3) with minor modifications: Owing to the fact that there is no terminal condition, we set $f = 0$ in (2.11), as well as $\alpha_T = 0$ in (2.4) and (3.2), making (2.5b) and (3.3b) obsolete. With the same reasoning, we set $T = \infty$, incurring $\tau \wedge T = \tau$ and $\mathcal{T} = (t_0 + \mathfrak{t}) \wedge \tau$; these simplifications are relevant for the expressions (2.11) and (3.3a). Proposition 3.1 and its proof can straightforwardly be generalized to the elliptic setting. An algorithm for solving elliptic PDEs of the type (4.3) in the spirit of the BSDE loss has been suggested in [48], using the same approximation framework as in [19] (cf. Remark 2.5). We note that the solutions to linear elliptic PDEs often admit alternative variational characterizations in terms of energy functionals [21]. An approach using the Feynman-Kac formula has been considered in [24].

4.2 Elliptic eigenvalue problems

We can extend the algorithmic approaches from Sections 2 and 3 to eigenvalue problems of the form

$$LV(x) = \lambda V(x), \quad x \in \Omega, \tag{4.5a}$$
$$V(x) = 0, \quad\quad\quad x \in \partial\Omega, \tag{4.5b}$$

corresponding to the choice $h(x, y, z) = -\lambda y$ in the elliptic PDE (4.3). Note, however, that h now depends on the unknown eigenvalue $\lambda \in \mathbb{R}$. Furthermore, we can consider nonlinear eigenvalue problems,

$$LV(x) + h\big(x, V(x), \sigma^\top \nabla V(x)\big) = \lambda V(x), \quad x \in \Omega, \tag{4.6a}$$
$$V(x) = 0, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad x \in \partial\Omega \tag{4.6b}$$

with a general nonlinearity $h \in C(\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d, \mathbb{R})$.

For the linear problem (4.5) it is known that, given suitable boundedness and regularity assumption on b and σ, there exists a unique principal eigenvalue with strictly positive eigenfunction in Ω, see [8, Theorem 2.3]. This motivates us to consider the above losses, now depending on λ, as well as enhanced with an additional term, preventing the trivial solution $V \equiv 0$. We define

$$\mathcal{L}^{\text{eigen}}(\varphi, \lambda) = \mathcal{L}_\lambda(\varphi) + \alpha_c \mathcal{L}_c(\varphi), \tag{4.7}$$

where $\mathcal{L}_\lambda(\varphi)$ stands for either the PINN, the BSDE, or the diffusion loss (with the nonlinearity h depending on λ), $\mathcal{L}_c(\varphi) = (\varphi(x_c) - 1)^2$, and $\alpha_c > 0$ is a weight. Here $x_c \in \Omega$ is chosen deterministically, preferably not too close to the boundary $\partial\Omega$. Clearly, the term $\mathcal{L}_c$ encourages $\varphi(x_c) = 1$, thus discouraging $\varphi \equiv 0$. We note that $V(x_c) = 1$ can be imposed on solutions to (4.5) without loss of generality, since the eigenfunctions are determined up

to a multiplicative constant only. Avoiding $\phi \equiv 0$ for nonlinear eigenvalue problems of the form (4.6) needs to be addressed on a case-by-case basis; we present an example in Section 6.4.2.

The idea is now to minimize $\mathcal{L}^{\text{eigen}}(\varphi, \lambda)$ with respect to $\varphi \in \mathcal{F}$ and $\lambda \in \mathbb{R}$ simultaneously, while constraining the function φ to be non-negative. According to the following proposition this is a valid strategy to determine the first eigenpair.

Proposition 4.1. *Let Ω be bounded, and assume that L is uniformly elliptic, that is, there exist constants $c_0, C_0 > 0$ such that*

$$c_0 |\xi|^2 \leq \sum_{i,j=1}^{d} (\sigma \sigma^\top)(x) \xi_i \xi_j \leq C_0 |\xi|^2 \tag{4.8}$$

for all $\xi \in \mathbb{R}^d$. Moreover, assume that b is bounded. Let $\varphi \in \mathcal{F}$ with $\varphi \geq 0$ and assume that $\mathcal{L}_\lambda(\varphi) = 0$ if and only if (4.5) is satisfied. Then the following are equivalent:

(1) φ is the principal eigenfunction for (4.5) with principal eigenvalue λ and normalization $\varphi(x_c) = 1$.

(2) $\mathcal{L}^{\text{eigen}}(\varphi, \lambda)$ vanishes on the pair (φ, λ),

$$\mathcal{L}^{\text{eigen}}(\varphi, \lambda) = 0. \tag{4.9}$$

Remark 4.2. The assumption that $\mathcal{L}_\lambda(\varphi)$ is equivalent to (4.5) is satisfied for any 'reasonable' loss function. For the diffusion loss, Proposition 3.1 establishes this condition whenever the coefficients in (4.5) are regular enough.

Proof. It is clear that (1) implies (2) by the construction of (4.7). For the converse direction, notice that (4.9) implies $\varphi(x_c) = 1$ as well as (4.5), that is, φ is an eigenfunction with eigenvalue λ. In conjunction with the constraint $\varphi \geq 0$, it follows by [8, Theorem 2.3] that φ is the principal eigenfunction. □

An alternative approach towards (4.5) can be found in [30], where the eigenvalue problem is connected to a parabolic PDE and formulated as a fixed point problem.

5   From losses to algorithms

In this section we discuss some details regarding implementation aspects. For convenience, let us start by stating a prototypical algorithm based on the losses introduced in Sections 2 and 3.

Function approximation.   In this paper, we rely on neural networks to provide the parametrization $\mathbb{R}^p \ni \theta \mapsto \varphi_\theta$ referred to in Algorithm 1 (but note that alternative function classes might offer specific benefits, see, for instance [75]). Standard feed-forward neural networks are given by

$$\varphi_\theta : \mathbb{R}^d \to \mathbb{R}, \quad \varphi_\theta(x) = A_L \varrho(A_{L-1} \varrho(\cdots \varrho(A_1 x + b_1) \cdots) + b_{L-1}) + b_L \tag{5.1}$$

Algorithm 1: Approximation of the solution V to the boundary value problem (1.1).

Choose a parametrization $\mathbb{R}^p \ni \theta \mapsto \varphi_\theta$.

Initialize φ_θ (with a parameter vector $\theta \in \mathbb{R}^p$).

Choose an optimization method `descent`, a batch size $K \in \mathbb{N}$ and a learning rate
$\eta > 0$. For PINN and diffusion losses choose weights $\alpha_{\text{int}}, \alpha_{\text{b}}, \alpha_{\text{T}} > 0$ and batch
sizes $K_{\text{b}}, K_{\text{T}} \in \mathbb{N}$. For BSDE and diffusion losses choose a step-size $\Delta t > 0$, for the
diffusion loss choose a trajectory length $\mathfrak{t} > 0$.

repeat

    Choose a loss function $\mathcal{L}$ from either (2.4), (2.11) or (3.2).

    Simulate data according to the chosen loss.

    Compute $\widehat{\mathcal{L}}(\varphi_\theta)$ as a Monte Carlo version of $\mathcal{L}$.

    Compute $\nabla_\theta \widehat{\mathcal{L}}(\varphi_\theta)$ using automatic differentiation.

    Update parameters: $\theta \leftarrow \theta - \eta \, \texttt{descent}(\nabla_\theta \widehat{\mathcal{L}}(\varphi_\theta))$.

until *convergence*;

Result: $\varphi_\theta \approx V$.

---

with a collection of matrices $A_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and vectors $b_l \in \mathbb{R}^{n_l}$ comprising the learnable parameters, $\theta = (A_l, b_l)_{1 \le l \le L}$. Here L denotes the depth of the network, and we have $n_0 = d$ as well as $n_L = 1$. The nonlinear activation function $\varrho : \mathbb{R} \to \mathbb{R}$ is to be applied componentwise.

Additionally, we define the *DenseNet* [21, 35] as a variant of the feed-forward neural network (5.1), containing additional skip connections,

$$\varphi_\theta^{\text{DenseNet}}(x) = A_L x_L + b_L, \tag{5.2}$$

where x_L is specified recursively as

$$y_{l+1} = \varrho(A_l x_l + b_l), \quad x_{l+1} = (x_l, y_{l+1})^\top, \quad x_1 = x \tag{5.3}$$

with $A_l \in \mathbb{R}^{n_l \times \sum_{i=0}^{l-1} n_i}$ and $b_l \in \mathbb{R}^l$ for $1 \le l \le L-1$, $n_0 = d$. Again, the collection of matrices A_l and vectors b_l comprises the learnable parameters.

Comparison of the losses (practical challenges). The PINN, BSDE and diffusion losses differ in the way training data is generated (see Fig. 3.1 and Table 5.1(a)); hence, the corresponding implementations face different challenges (see Table 5.1(b)).

First, the BSDE and diffusion losses rely on trajectorial data obtained from the SDE (1.3), in contrast to the PINN loss (cf. the first row in Table 5.1(a)). As a consequence, the BSDE and diffusion losses do not require the computation of second-order derivatives, as those are approximated implicitly using Itô's formula and the SDE (1.3), cf. the first row in Table 5.1(a). From a computational perspective, the PINN loss therefore faces a significant overhead in high dimensions when the diffusion coefficient σ is not sparse (as the expression (2.5a) involves d^2 second-order partial derivatives). We notice in passing that an approach similar to the diffusion loss circumventing this problem has been proposed in [77, Section 3]. On the other hand, evaluating the BSDE and diffusion losses

Table 5.1: Comparison of the different losses.

	PINN	BSDE	Diffusion
SDE simulation		✗	✗
Boundary data	✗		✗

	PINN	BSDE	Diffusion
Hessian computations	✗		
Exit time computations		✗	
Weight tuning	✗		✗
Long runtimes		✗	
Discretization		✗	✗

(a) The three losses can be characterized by how training data is generated.

(b) In this table we list potential challenges and drawbacks for the corresponding losses.

requires discretizing the SDE (1.3), incurring additional numerical errors (cf. the last row in Table 5.1(b) and the discussion below in Section 5.1).

Second, the PINN and diffusion losses incorporate boundary and final time constraints (see (1.1c) and (1.1b)) explicitly by sampling additional boundary data (see (2.5b), (2.5c), (3.3b), (3.3c) and cf. the second row in Table 5.1(a)). On the one hand, this approach necessitates choosing the weights $\alpha_{\mathrm{int}}, \alpha_{\mathrm{b}}, \alpha_{\mathrm{T}} > 0$; it is by now well established that while algorithmic performance depends quite sensitively on a judicious tuning of these weights, general and principled guidelines to address this issue are not straightforward (see, however, [80–83]). Weight-tuning, on the other hand, is not required for implementations relying on the BSDE loss, as the boundary data is accounted for implicitly by the hitting event $\{(X_t, t) \notin \Omega \times [0, T]\}$ and the corresponding first two terms on the right-hand side of (2.11). The hitting times $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ may however be large, leading to a computational overhead in the generation of the training data (but see Section 5.2.1), and are generally hard to compute accurately (but see Section 5.1).

5.1   Simulation of diffusions and their exit times

The BSDE and diffusion losses rely on trajectorial data obtained from the stochastic process defined in (1.3). In practice, we approximate this SDE on a time grid $t_0 \le t_1 \le \cdots \le t_N$, for instance using the Euler-Maruyama scheme [47]

$$\widetilde{X}_{n+1} = \widetilde{X}_n + b(\widetilde{X}_n, t_n)\Delta t + \sigma(\widetilde{X}_n, t_n)\sqrt{\Delta t}\, \xi_{n+1}, \qquad (5.4)$$

or, to be precise, by its stopped version

$$\widehat{X}_{n+1} = \widehat{X}_n + \left(b(\widehat{X}_n, t_n)\Delta t + \sigma(\widehat{X}_n, t_n)\sqrt{\Delta t}\, \xi_{n+1} \right) \mathbb{1}_{\mathcal{C}_{n+1}} \qquad (5.5)$$

with step condition $\mathcal{C}_n := \{\widetilde{X}_n \in \Omega\} \vee \{t_n \le T\}$ and time-increment $t_{n+1} = t_n + \Delta t\, \mathbb{1}_{\mathcal{C}_{n+1}}$, where Δt is the step-size and $\xi_{n+1} \sim \mathcal{N}(0, \mathrm{Id}_{d \times d})$ are standard normally distributed random variables. We can then straightforwardly construct Monte Carlo estimator versions of either the BSDE or the diffusion loss. For example, the discrete version of the domain

part of the diffusion loss (3.3a) reads

$$
\begin{aligned}
&\widehat{\mathcal{L}}_{\text{diffusion,int}}^{(K,N)}(\varphi) \\
&= \frac{1}{K} \sum_{k=1}^{K} \Bigg(\varphi\big(\widehat{X}_N^{(k)}, t_N^{(k)}\big) - \varphi\big(\widehat{X}_0^{(k)}, t_0^{(k)}\big) - \sum_{n=0}^{N-1} \sigma^\top \nabla \varphi\big(\widehat{X}_n^{(k)}, t_n^{(k)}\big) \cdot \xi_{n+1}^{(k)} \sqrt{\Delta t}\, \mathbb{1}_{\mathcal{C}_n^{(k)}} \\
&\qquad + \sum_{n=0}^{N-1} h\left(\widehat{X}_n^{(k)}, t_n^{(k)}, \varphi\big(\widehat{X}_n^{(k)}, t_n^{(k)}\big), \sigma^\top \nabla \varphi\big(\widehat{X}_n^{(k)}, t_n^{(k)}\big) \right) \Delta t\, \mathbb{1}_{\mathcal{C}_n^{(k)}} \Bigg)^2, \qquad (5.6)
\end{aligned}
$$

where K is the sample size, $N = \mathfrak{t}/\Delta t$ is the maximal discrete-time trajectory length, and $(\widehat{X}_n^{(k)})_{n=1\ldots N}^{k=1\ldots K}$ are independent copies of the iterates from (5.5). The Monte Carlo version of the BSDE loss can be formed analogously.

Given suitable growth and regularity assumptions on the coefficients, the discretization errors of the forward and backward processes are of order $\sqrt{\Delta t}$ [47, 85], cf. also [29] for a numerical analysis on the original version of the BSDE loss. However, the stopped Euler-Maruyama scheme (5.5) incurs additional errors due to the approximation of the first exit times from Ω. In the two left panels of Fig. 5.1 we illustrate this problem by displaying multiple "last locations" obtained according to (5.5), using two different step-sizes Δt. Clearly, all these points should in principle lie on the boundary, (naively) requiring a computationally costly small step-size.

Sophisticated approaches towards the accurate simulation of exit times for diffusions discretizing exit times have been put forward, see, e.g. [15,33,58]. For our purposes, however, it is not essential to compute the exit times, as long as the simulated trajectories are stopped accurately. We therefore suggest the following two attempts that aim at improving the sampling of boundary data:

1. Rescaling: Start X_0 randomly in Ω, simulate the trajectory and stop once the boundary has been crossed, however scale the last time step in such a way that the trajectory exactly ends on $\partial\Omega$.

2. Time-reversal: Start X_0 on the boundary $\partial\Omega$ and simulate the trajectory for a given time T (unless it hits the boundary again before time T, in this case stop the trajectory
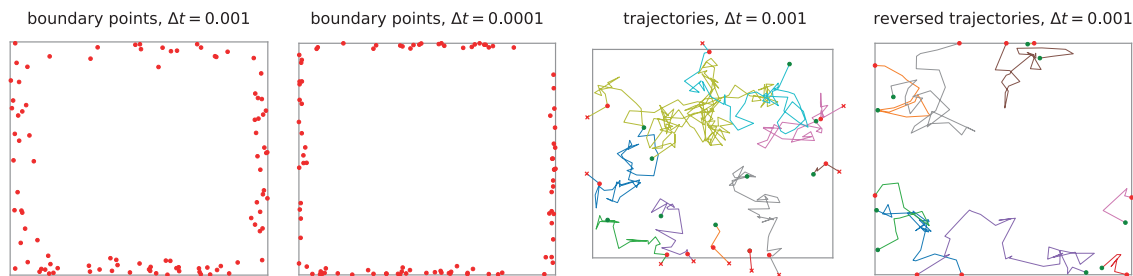

Figure 5.1: Illustration of the boundary data in the BSDE method.

accordingly or resimulate). Then reverse the process such that the reversed process exactly ends on the boundary.

An illustration of these two strategies can be found in the two right panels of Fig. 5.1. In our numerical experiments we have found that both rescaling and time-reversal improves the performance of the BSDE method somewhat, but further research is needed.

5.2 Further modifications of the losses

In the following we discuss modifications of the PINN, BSDE and diffusion losses, relating also to versions that have appeared in the literature before.

5.2.1 Forward control

We can modify the SDE-based BSDE and diffusion losses by including control terms $v \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ in the forward process (1.3), yielding the controlled diffusion

$$dX_s^v = \big(b(X_s^v,s) + \sigma(X_s^v,s)v(X_s^v,s)\big)ds + \sigma(X_s^v,s)\,dW_s. \tag{5.7}$$

Applying Itô's formula we may obtain losses similar to those considered in Sections 2 and 3. For instance, alternative versions of the diffusion loss take the form

$$\mathcal{L}_{\text{diffusion,int}}^{t,v}(\varphi)$$
$$= \mathbb{E}\Bigg[\bigg(\varphi(X_{\mathcal{T}}^v,\mathcal{T}) - \varphi(X_{t_0}^v,t_0) - \int_{t_0}^{\mathcal{T}} \sigma^\top \nabla\varphi(X_s,s)\cdot dW_s$$
$$+ \int_{t_0}^{\mathcal{T}} \Big[h(X_s^v,s,\varphi(X_s^v,s),\sigma^\top\nabla\varphi(X_s^v,s)) - v(X_s^v,s)\cdot\sigma^\top\nabla\varphi(X_s^v,s)\Big]ds\bigg)^2\Bigg], \quad (5.8)$$

replacing (3.3a). We note in passing that Proposition 3.1 extends straightforwardly to $\mathcal{L}_{\text{diffusion,int}}^{t,v}$ under the assumption that v satisfies appropriate Lipschitz and growth conditions.

Similar considerations apply for the BSDE loss, noting that for solutions to the generalized BSDE system [62]

$$dX_s^v = \big(b(X_s^v,s) + \sigma(X_s^v,s)v(X_s^v,s)\big)ds + \sigma(X_s^v,s)\,dW_s, \qquad X_{t_0}^v = x_{\text{init}}, \tag{5.9a}$$
$$dY_s^v = -h(X_s^v,s,Y_s^v,Z_s^v)\,ds + v(X_s^v,s)\cdot Z_s^v\,ds + Z_s^v\cdot dW_s, \quad Y_T^v = k(X_{T\wedge\tau}^v, T\wedge\tau), \tag{5.9b}$$

we still have the relations

$$Y_s^v = V(X_s^v,s), \quad Z_s^v = \sigma^\top\nabla V(X_s^v,s) \tag{5.10}$$

for suitable $v \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$, analogously to (2.8). This immediately incurs the family

of losses

$$
\begin{aligned}
&\mathcal{L}_{\mathrm{BSDE}}^{v}(\varphi) \\
&= \mathbb{E}\Bigg[\Bigg(f(X_{\tau\wedge T}^{v})\mathbb{1}_{\tau\wedge T=T} + g(X_{\tau\wedge T}^{v}, \tau\wedge T)\mathbb{1}_{\tau\wedge T=\tau} - \varphi(X_{t_0}^{v}, t_0) - \int_{t_0}^{\tau\wedge T} \sigma^{\top}\nabla\varphi(X_s^{v}, s)\cdot \mathrm{d}W_s \\
&\qquad + \int_{t_0}^{\tau\wedge T}\Big(h(X_s^{v}, s, \varphi(X_s, s), \sigma^{\top}\nabla\varphi(X_s, s)) - v(X_s^{v}, s)\cdot\sigma^{\top}\nabla\varphi(X_s^{v}, s)\Big)\mathrm{d}s\Bigg)^2\Bigg], \quad (5.11)
\end{aligned}
$$

parameterized by $v \in C(\mathbb{R}^d \times [0, T], \mathbb{R}^d)$.

Adding a control to the forward process can be understood as driving the data generating process into regions of interest, for instance possibly alleviating the problem that exit times might be large (see Table 5.1(b) and the corresponding discussion). Identifying suitable forward controls might be an interesting topic for future research (we refer to [62] for some systematic approaches in this respect relating to Hamilton-Jacobi-Bellman PDEs).

5.2.2 Approximating the gradient of the solution

The constraints imposed by the BSDE system (2.6) can be enforced by losses that are slightly different from Definition 2.2. Going back to [19], we can for instance use the fact that the backward process Y can be written in a forward way, yielding the discrete-time process

$$
\widehat{Y}_{n+1} = \widehat{Y}_n - h\big(\widehat{X}_n, t_n, \widehat{Y}_n, \widehat{Z}_n\big)\Delta t + \widehat{Z}_n\cdot\xi_{n+1}\sqrt{\Delta t}. \quad (5.12)
$$

The scheme (5.12) is explicit, the unknowns being $\widehat{Y}_0$ and $\widehat{Z}_n$, for $n \in \{0, \dots, N-1\}$. This motivates approximating the single parameter $y_0 \approx \widehat{Y}_0 \in \mathbb{R}$ as well as the vector fields $\phi \approx \sigma^{\top}\nabla V \in C(\mathbb{R}^d \times [0, T], \mathbb{R}^d)$, rather than V directly. This approach gives rise to the loss

$$
\begin{aligned}
\mathcal{L}_{\mathrm{BSDE}-2}(\phi, y_0) = \mathbb{E}\Bigg[\Bigg(&f(X_{\tau\wedge T})\mathbb{1}_{\tau\wedge T=T} + g(X_{\tau\wedge T}, \tau\wedge T)\mathbb{1}_{\tau\wedge T=\tau} - y_0 \\
&- \int_0^{\tau\wedge T}\phi(X_s, s)\cdot\mathrm{d}W_s + \int_0^{\tau\wedge T} h(X_s, s, Y_s, \phi(X_s, s))\,\mathrm{d}s\Bigg)^2\Bigg]. \quad (5.13)
\end{aligned}
$$

In this setting X_0 has to be chosen deterministically; we note that a potential drawback is thus that the solution is only expected to be approximated accurately in regions that can be reached by the forward process X_t (starting at X_0) with sufficiently high probability. It has been shown in [62] that alternative losses (like the log-variance loss) can be considered whenever the nonlinearity h only depends on the solution through its gradient, in which case the extra parameter y_0 can be omitted.

5.2.3   Penalizing deviations from the discrete scheme

Another approach that is rooted in the discrete-time backward process (5.12) has been suggested in [70] for problems on unbounded domains. It relies on the idea to penalize deviations from (5.12), for each $n \in \{0, \dots, N-1\}$ (cf. also [36, 75], where however an implicit scheme and backward iterations are used). Aiming for $\widehat{Y}_n \approx \varphi(\widehat{X}_n, t_n)$, $\widehat{Z}_n \approx \sigma^\top \nabla \varphi(\widehat{X}_n, t_n)$, this motivates the loss

$$\widehat{\mathcal{L}}_{\text{BSDE}-3}^{(K,N)}(\varphi) = \alpha_{\text{int}} \widehat{\mathcal{L}}_{\text{BSDE}-3,\text{int}}^{(K,N)}(\varphi) + \alpha_{\text{b}} \widehat{\mathcal{L}}_{\text{BSDE}-3,\text{b}}^{(K,N)}(\varphi) \tag{5.14}$$

with interior part

$$\widehat{\mathcal{L}}_{\text{BSDE}-3,\text{int}}^{(K,N)}(\varphi) = \frac{1}{K} \sum_{k=1}^{K} \sum_{n=0}^{N-1} \left(\varphi\big(\widehat{X}_{n+1}^{(k)}\big) - \varphi\big(\widehat{X}_n^{(k)}\big) + h\left(\widehat{X}_n^{(k)}, \varphi\big(\widehat{X}_n^{(k)}\big), \sigma^\top \nabla \varphi\big(\widehat{X}_n^{(k)}\big)\right) \Delta t \right.$$
$$\left. - \sigma^\top \nabla \varphi\big(\widehat{X}_n^{(k)}\big) \xi_{n+1} \sqrt{\Delta t} \right)^2 \tag{5.15}$$

and boundary term

$$\widehat{\mathcal{L}}_{\text{BSDE}-3,\text{b}}^{(K,N)}(\varphi) = \frac{1}{K} \sum_{k=1}^{K} \left(\varphi\big(\widehat{X}_N^{(k)}\big) - g\big(\widehat{X}_N^{(k)}\big) \right)^2. \tag{5.16}$$

A generalization to equations posed on bounded domains is straightforward. We also note that in contrast to the diffusion loss, (5.14) does not seem to naturally derive from a continuous-time formulation. Interestingly, $\widehat{\mathcal{L}}_{\text{BSDE}-3}^{(K,N)}$ can be related to the diffusion loss via Jensen's inequality,

$$\widehat{\mathcal{L}}_{\text{diffusion,int}}^{(K,N)}(\varphi) \leq N \widehat{\mathcal{L}}_{\text{BSDE}-3,\text{int}}^{(K,N)}(\varphi). \tag{5.17}$$

Yet another approach that is based on a discrete backward scheme is the following. Let us initialize $\widehat{Y}_0 = \varphi(\widehat{X}_0, 0)$ and simulate

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h\big(\widehat{X}_n, \widehat{Y}_n, \sigma^\top \nabla \varphi(\widehat{X}_n, t_n)\big) \Delta t + \sigma^\top \nabla \varphi(\widehat{X}_n, t_n) \cdot \xi_{n+1} \sqrt{\Delta t} \tag{5.18}$$

for $n \in \{0, \dots, N-1\}$, where, similarly to (5.13), but in contrast to (5.15), only $\widehat{Z}_n$ is replaced by its approximation $\sigma^\top \nabla \varphi(\widehat{X}_n, t_n)$ while $\widehat{Y}_n$ is retained from previous iteration steps. Again penalizing deviations from the discrete-time scheme, we can now introduce the loss

$$\widehat{\mathcal{L}}_{\text{BSDE}-4}^{(K,N)}(\varphi) = \frac{\alpha_1}{K} \sum_{k=1}^{K} \sum_{n=0}^{N} \left(\varphi\big(\widehat{X}_n^{(k)}, t_n\big) - \widehat{Y}_n^{(k)} \right)^2 + \frac{\alpha_2}{K} \sum_{k=1}^{K} \left(\varphi\big(X_b^{(k)}\big) - g\big(X_b^{(k)}\big) \right)^2. \tag{5.19}$$

Both in $\mathcal{L}_{\text{BSDE}-3}$ and $\mathcal{L}_{\text{BSDE}-4}$ the deterministic initial condition $\widehat{X}_0$ at $t = 0$ can be replaced by random choices $(\widehat{X}_{t_0}, t_0) \sim \nu_{\Omega \times [0,T]}$, adjusting the sums in (5.15) and (5.19) accordingly.

6   Numerical experiments

In this section we provide several numerical examples of high-dimensional parabolic and elliptic PDEs that shall demonstrate the performances of Algorithm 1 using the different loss functions introduced previously. We focus on the PINN, BSDE and diffusion losses from Sections 2 and 3 since their modified versions from Section 5.2 in general led to similar or worse performances. If not specified otherwise, the approximation of φ relies on a DenseNet architecture defined in (5.2) with ReLU activation function and four hidden layers with $d + 20$, d, d, d units respectively (recall that d specifies the dimension). The optimization is carried out using the Adam optimizer [45] with standard parameters and learning rate $\eta = 0.001$. Throughout, we take $K_{\mathrm{int}} = 200$ samples inside the domain Ω and (for the PINN and diffusion losses) $K_{\mathrm{b}} = 50$ samples on the boundary, per gradient step. For the SDE discretization we choose a step-size of $\Delta t = 0.001$. The weight configurations for the PINN and diffusion losses are optimized manually; we then only report the results for the optimal settings (see the discussion on weight-tuning in Section 5). We refer to the code at `https://github.com/lorenzrichter/path-space-PDE-solver`.

6.1   Nonlinear toy problems

Let us start with a nonlinear toy problem for which analytical reference solutions are available. Throughout this subsection, the domain of interest is taken to be the unit ball $\Omega = \{ x \in \mathbb{R}^d : |x| < 1 \}$.

6.1.1   Elliptic problem with Dirichlet boundary data

We first consider an elliptic boundary value problem of the form (4.3). Let $\gamma \in \mathbb{R}$ and choose

$$b(x,t) = \mathbf{0}, \quad \sigma(x,t) = \sqrt{2}\,\mathrm{Id}_{d \times d}, \quad g(x) = e^{\gamma}, \tag{6.1a}$$

$$h(x,y,z) = -2\gamma y(\gamma |x|^2 + d) + \sin\left(e^{2\gamma |x|^2} - y^2\right). \tag{6.1b}$$

It is straightforward to verify that

$$V(x) = e^{\gamma |x|^2} \tag{6.2}$$

is the unique solution to (4.3).

We consider $d = 50$ and set $\gamma = 1$. For the PINN and diffusion losses the optimized weights are given by $\alpha_{\mathrm{int}} = 10^{-5}$, $\alpha_{\mathrm{b}} = 1$ and $\alpha_{\mathrm{int}} = 0.1$, $\alpha_{\mathrm{b}} = 1$, respectively. We sample the data uniformly and take a maximal (discrete-time) trajectory length of $N = 20$ (i.e. $t = 0.02$) for the diffusion loss. In Fig. 6.1 (left panel) we display the average relative errors $|\varphi(x) - V(x)|/V(x)$ as a function of $r = |x|$. Due to volume distortion in high dimensions, very few samples are drawn close to the center of the ball, and hence the numerical results appear to be unreliable for $r \leq 0.8$. While this effect could be alleviated by changing the measure ν_Ω accordingly, we content ourselves here with a comparison for $r \in [0.8, 1]$. In the right panel we display the L^2 error during the training iterations
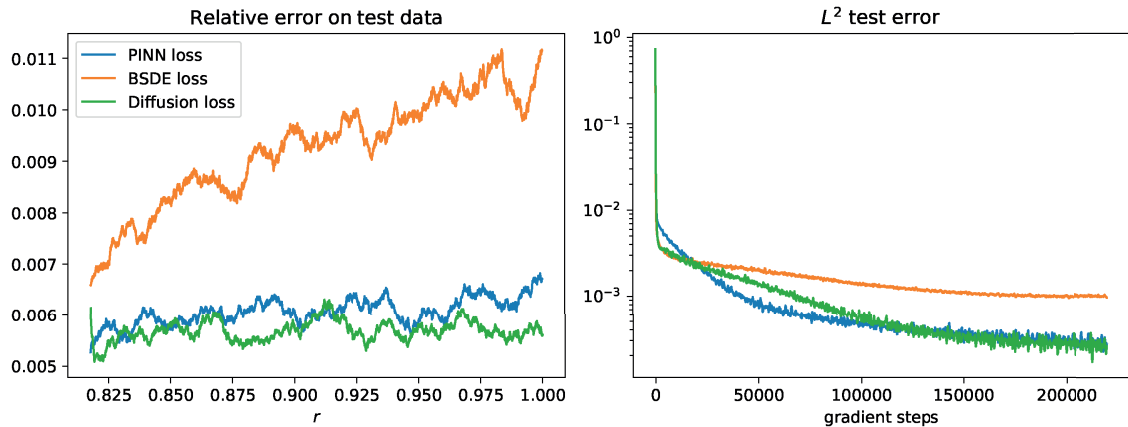
Figure 6.1: Left: Average relative errors as a function of $r = |x|$ evaluated on uniformly sampled data for the three losses smoothed with a moving average over 500 data points. For the training the diffusion loss relies on trajectory lengths of $\mathfrak{t} = 0.02$. Right: L^2 error during the training iterations evaluated on uniformly sampled test data.

evaluated on uniformly sampled test data. We observe that the PINN and diffusion losses yield similar results and that the BSDE loss performs worse, in particular close to the boundary. We attribute this effect to the challenges inherent in the simulation of hitting times, see Section 5.1.

In the diffusion loss as stated in Definition 3.1 we are free to choose the length $\mathfrak{t}$ of the forward trajectories, which affects the generated training data. Let us therefore investigate how different choices of $\mathfrak{t}$ influence the performance of Algorithm 1. To this end, we consider again the elliptic problem from Section 6.1.1 and vary $\mathfrak{t}$. To be precise, let us fix different step-sizes Δt and vary the Euler steps N (recalling that $\mathfrak{t} = N\Delta t$), once choosing the weight $\alpha_{\text{int}} = 0.1$, as before, and once by considering $\alpha_{\text{int}} = 10$. For the former choice we can see in Fig. 6.2(a) that larger trajectories tend to be better until a plateau is reached,

(a) Weight $\alpha_{\text{int}} = 0.1$

(b) Weight $\alpha_{\text{int}} = 10$
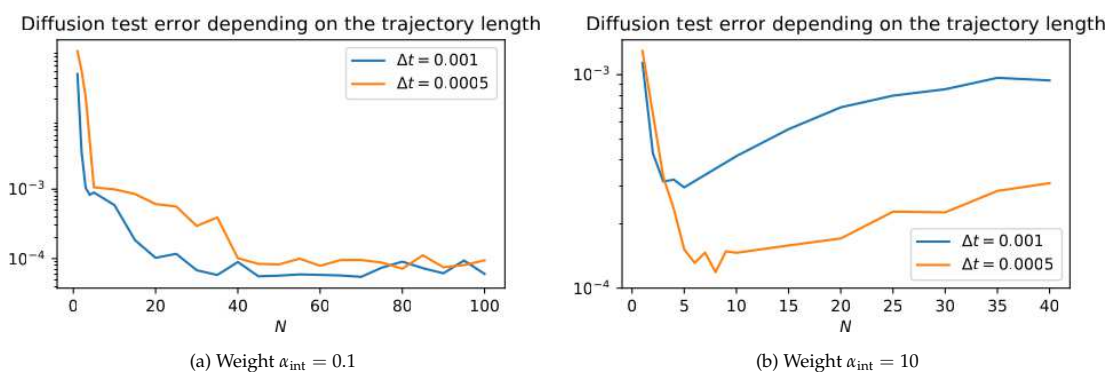
Figure 6.2: We display the L^2 error that one attains when using different choices of the maximal Euler steps N in the diffusion loss for different discretization step-sizes Δt.

whereas for the latter it turns out that there seems to be an optimal choice of the trajectory length, as displayed in Fig. 6.2(b).

6.1.2 Elliptic problem requiring the full Hessian matrix

We consider the setting specified in (6.1), replacing however the diffusion coefficient and the nonlinearity by

$$
\sigma(x,t) = \sqrt{\frac{2}{d}} \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix},
$$

$$
h(x,y,z) = -2\gamma y \left(\gamma \sum_{i,j=1}^{d} x_i x_j + d \right) + \sin\left(e^{2\gamma|x|^2} - y^2\right),
$$

(6.3)

respectively. Again, we can check that $V(x) = e^{\gamma|x|^2}$ is the unique solution to the corresponding boundary value problem. Since σ is not diagonal anymore, the differential operator (1.2) contains the full Hessian matrix of second-order derivatives of the candidate solution φ. As discussed in Section 5 (see Table 5.1(b)) this particularly impacts the runtime of the PINN method since all derivatives need to be computed explicitly. For the BSDE and diffusion losses, on the other hand, second-order derivatives are implicitly approximated using the underlying Brownian motion and we therefore do not expect significantly longer runtimes.

Let us consider $d = 20$ and $\gamma = 1$ and as before choose $N = 20$ (i.e. $\mathfrak{t} = 0.02$) for the diffusion loss. In Fig. 6.3 we display the L^2 error during the training process, plotted against the number of gradient steps (left panel) and against the runtime (right panel). As expected, the PINN loss takes significantly longer. This effect should become even more pronounced with growing state space dimension d.
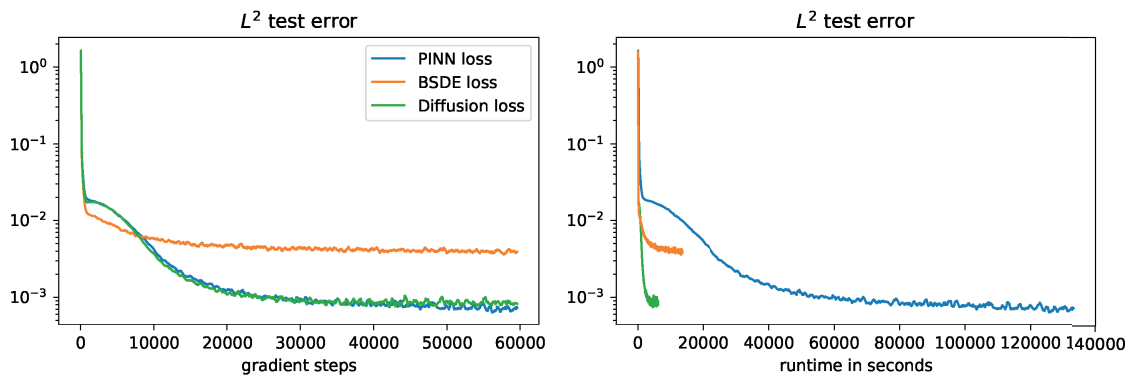

Figure 6.3: L^2 error during the training process evaluated on test data for the three losses, plotted against the number of gradient steps (left panel) and against the runtime (right panel). The diffusion loss relies on trajectory lengths of $\mathfrak{t} = 0.02$.

6.1.3 Parabolic problem with Neumann boundary data

Let us now consider the parabolic problem (1.1), with the Dirichlet boundary condition (1.1c) replaced by its Neumann counterpart

$$\partial_{\vec{n}} V(x,t) = g^N(x,t), \quad (x,t) \in \partial\Omega \times [0,T]. \tag{6.4}$$

Here, $\partial_{\vec{n}} V := \nabla V \cdot \vec{n}$ refers to the (outward facing) normal derivative at the boundary $\partial\Omega$. We take

$$b(x,t) = \mathbf{0}, \quad \sigma(x,t) = \sqrt{2}\,\mathrm{Id}_{d\times d}, \quad f(x) = e^{\gamma|x|^2+T}, \quad g^N(x,t) = 2\gamma e^{\gamma+t}, \tag{6.5a}$$

$$h(x,t,y,z) = -y\big(2\gamma(2\gamma|x|^2 + d) + 1\big) + \sin\big(e^{2\gamma|x|^2+2t} - y^2\big). \tag{6.5b}$$

In this case,

$$V(x,t) = e^{\gamma|x|^2+t} \tag{6.6}$$

provides the unique solution.

We choose $d = 20$ and $\gamma = 1$. For the diffusion loss we take $N = 25$ (i.e. t = 0.025) for the trajectory length. In the left and central panels of Fig. 6.4 we display the approximated solutions along the curve $\{(\kappa,\dots,\kappa)^\top : \kappa \in [-1/\sqrt{d}, 1/\sqrt{d}]\}$ for two different times. We can see that both the diffusion and the PINN loss work well, with small advantages for the PINN loss. The right panel displays the L^2 test error over the iterations and confirms this observation.
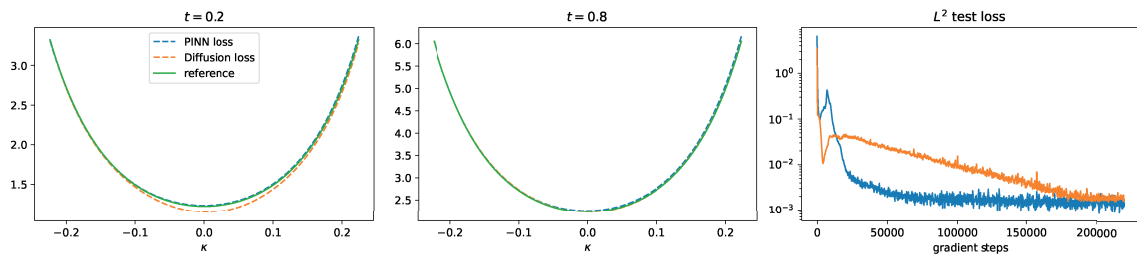

Figure 6.4: Left and central panel: Approximations along a curve for two different times using the diffusion and PINN losses. Right: L^2 test error along the training iterations.

6.2 Committor functions

Committor functions are important quantities in molecular dynamics as they specify likely transition pathways as well as transition rates between (potentially metastable) regions or conformations of interest [20,56]. Since for most practical applications those functions are high-dimensional and hard to compute, there have been recent attempts to approach this problem using neural networks [46,53,76]. Based on the fact that committor functions fulfill elliptic boundary value problems, we can rely on the methods discussed in this paper.

For an $\mathbb{R}^d$-valued stochastic process $(X_t)_{t\geq 0}$ with continuous sample paths and two disjoint open sets $A, B \subset \mathbb{R}^d$, the committor function V computes the probability of X hitting A before B when starting in $x \in \mathbb{R}^d$, that is,

$$V(x) = \mathbb{P}(\tau_B < \tau_A | X_0 = x) = \mathbb{E}[\mathbb{1}_B(X_\tau)|X_0 = x]. \tag{6.7}$$

Here, $\tau_A = \inf\{t > 0 : X_t \in A\}$ and $\tau_B = \inf\{t > 0 : X_t \in B\}$ refer to the hitting times corresponding to the sets A and B. In the case when $(X_t)_{t\geq 0}$ is given as the unique strong solution to the SDE (1.3), it can be shown via the Kolmogorov backward PDE [66, Section 2.5] that V fulfills the elliptic boundary value problem

$$LV = 0, \quad V|_{\partial A} = 0, \quad V|_{\partial B} = 1, \tag{6.8}$$

where L as in (1.2) refers to the associated infinitesimal generator, see, for instance, [20]. In the notation of (4.3) we have $\Omega = \mathbb{R}^d \setminus (A \cup B)$, $h = 0$ and $g(x) = \mathbb{1}_B(x)$.

Following [32, Section V.A], we consider $(X_t)_{t\geq 0}$ to be a standard Brownian motion starting at $x \in \mathbb{R}^d$, that is, $X_t = x + W_t$, corresponding to $b = \mathbf{0}$ and $\sigma = \mathrm{Id}_{d\times d}$ in (1.3). The sets A and B are defined as

$$A = \{x \in \mathbb{R}^d : |x| < a\}, \quad B = \{x \in \mathbb{R}^d : |x| > b\} \tag{6.9}$$

with $b > a > 0$. Hence, in this case the committor function describes the statistics of leaving a spherical shell through one of its boundaries. The solution takes the form

$$V(x) = \frac{a^2 - |x|^{2-d}a^2}{a^2 - b^{2-d}a^2} \tag{6.10}$$

for $d \geq 3$. Let us consider $d = 10$ as well as $a = 1$, $b = 2$. We take a DenseNet with the tanh activation function and compare the three losses against each other. For the diffusion loss we take $N = 50$ (i.e. $t = 0.05$) for the trajectory length. In Fig. 6.5 we display the approximated solutions along a curve $\{(\kappa, \ldots, \kappa)^\top : \kappa \in [a/\sqrt{d}, b/\sqrt{d}]\}$ in the left panel,
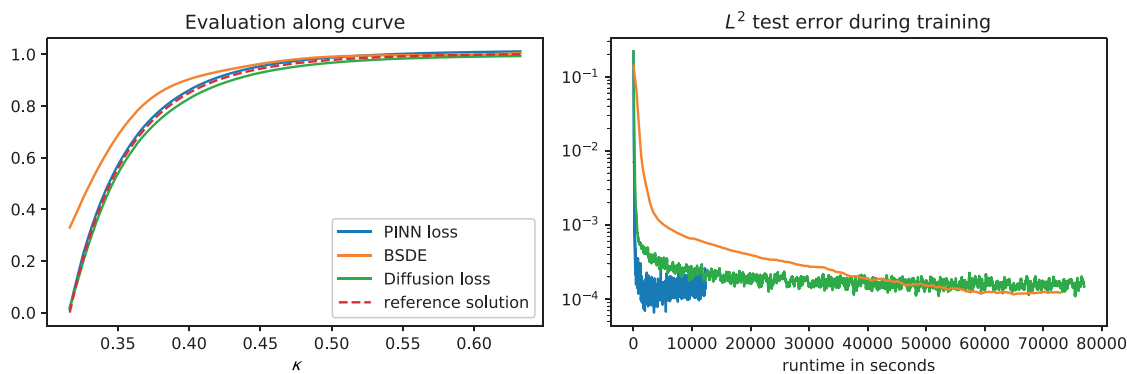

Figure 6.5: Left: approximations of the 10-dimensional committor function evaluated along a curve. Right: moving average of the test L^2 error along the training iterations.
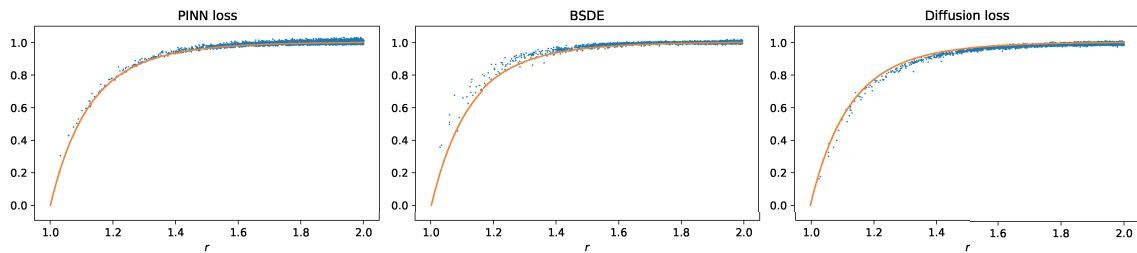
Figure 6.6: We plot the approximated committor functions evaluated at 10000 points uniformly sampled from the domain Ω (blue dots) and compare those to the reference solution (orange line) as a function of $r = |x|$.

realizing that in particular the PINN and diffusion losses lead to good approximations. This can also be observed in the right panel, where we plot a moving average of the L^2 error on test data based on a moving window of length 200. The BSDE loss appears to be especially error-prone close to the left end-point of the curve displayed in Fig. 6.5, that is, close to the inner shell. Due to the volume distortion in high dimensions, few samples are drawn according to ν_Ω for small values of $|x|$, see Fig. 6.6. We hence conclude that in this example, the BSDE loss suffers particularly from the relative sparsity of the samples, possibly in conjunction with numerical errors made while estimating the hitting times at the inner shell (see Section 5.1).

6.3 Parabolic Allen-Cahn equation on an unbounded domain

The Allen-Cahn equation in $d = 100$ has been suggested as a benchmark problem in [19]. It is an example of a parabolic PDE posed on an unbounded domain,

$$(\partial_t + L)V(x,t) + V(x,t) - V^3(x,t) = 0, \quad (x,t) \in \mathbb{R}^d \times [0,T], \qquad (6.11a)$$

$$V(x,T) = f(x), \qquad\qquad\qquad x \in \mathbb{R}^d \qquad (6.11b)$$

with $f(x) = (2 + (2/5)|x|^2)^{-1}$ and $T = 3/10$. We restrict attention to the ball $\{x \in \mathbb{R}^d : |x| < r\}$ with radius $r = 7$, from which we sample the initial points of the trajectories for the BSDE and diffusion losses, or the training data for the PINN loss, respectively. Instead of using the uniform distribution on this set, we consider sampling uniformly on a box around the origin with side length 2 and multiplying each data point x by $r/|x|$. In contrast to uniform sampling, this approach generates more samples close to the origin, which we observe to slightly improve the accuracy of the obtained solutions. For the diffusion loss we choose $N = 25$ (i.e. $\mathfrak{t} = 0.025$) for the trajectory length. We compare our approximations to a reference solution at $x_0 = (0, \ldots, 0)^\top$ for different times $t \in [0, T]$ that is provided by a branching diffusion method specified in [19]. In Fig. 6.7 we see that all three attempts match this reference solution, with very minor advantages (for instance at the right end point) for the PINN and diffusion losses. In Table 6.1 we display the computation times until convergence, realizing that the BSDE loss needs significantly longer, which is in accordance with Remark 3.1 and might be explained by the fact that $\mathfrak{t} < T$. We note that the computation times are longer in comparison to e.g. [19] since we aim for
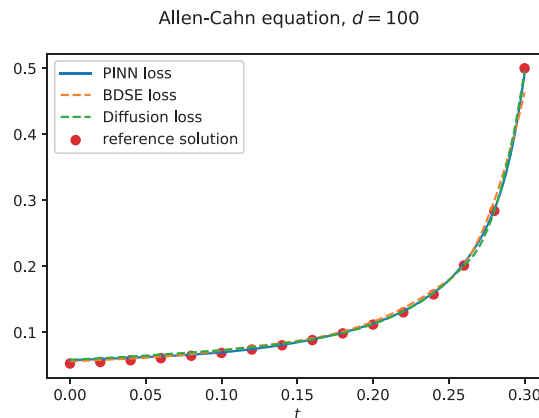
Figure 6.7: Approximation of the solution to an Allen-Cahn equation in $d = 100$ using different losses compared to a reference solution at $x_0 = (0, \ldots, 0)^\top$ for different times $t \in [0, T]$.

Table 6.1: Computation times until convergence.

Computation time	
PINN loss	325.46 min
BSDE loss	4280.68 min
Diffusion loss	194.38 min

a solution on a given domain, whereas other attempts only strive for approximating the solution at a single point.

6.4 Elliptic eigenvalue problems

In this section we provide two examples for the approximation of principal eigenvalues and corresponding eigenfunctions. The first one is a linear problem and therefore Proposition 4.1 assures that the minimization of an appropriate loss as in (4.7) leads to the desired solution. The second example is a nonlinear eigenvalue problem, for which we can numerically show that our algorithm still provides the correct solution.

6.4.1 Fokker-Planck equation

As suggested in [30], we aim at computing the principal eigenpair associated to a Fokker-Planck operator, defined by

$$LV = -\Delta V - \nabla \cdot (V \nabla \Psi) \tag{6.12}$$

for $V : \Omega \to \mathbb{R}$ on the domain $\Omega = [0, 2\pi]^d$, and where

$$\Psi(x) = \sin \left(\sum_{i=1}^{d} c_i \cos(x_i) \right)$$
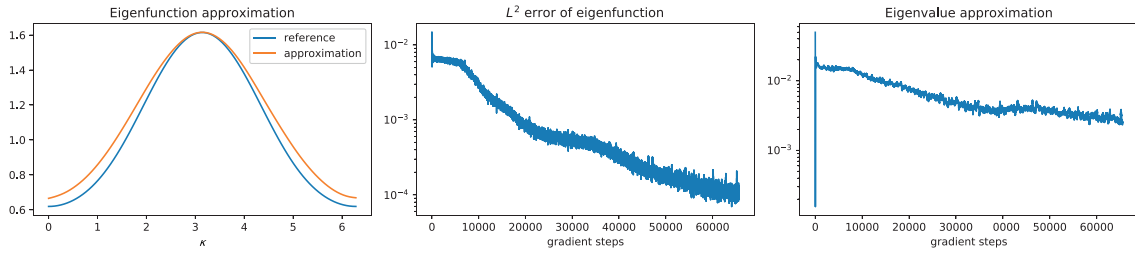
Figure 6.8: Left: Approximation and reference of the eigenfunction corresponding to the principal eigenvalue of the Fokker-Planck operator (6.12). Middle: L^2 error with respect to test data along the training iterations. Right: Moving average of the absolute value of the approximated eigenvalue along the gradient steps.

is a potential with constants $c_i \in [0.1, 1]$, assuming periodic boundary conditions. This results in the eigenvalue problem

$$\Delta V(x) + \nabla \Psi(x) \cdot \nabla V(x) + \Delta \Psi(x) V(x) = -\lambda V(x), \qquad (6.13)$$

and

$$V(x) = e^{-\Psi(x)} \qquad (6.14)$$

is an eigenfunction associated to the principal eigenvalue $\lambda = 0$, see [66, Section 4.7]. We choose $c_i = 0.1$, $i = 1, \dots, d$, and approach this problem in dimension $d = 5$ following Section 4.2, i.e. by minimizing the loss (4.7), where for $\mathcal{L}$ we choose the diffusion loss with $N = 20$ (i.e. $t = 0.02$) and the periodic boundary condition is encoded via the term (2.12). Here and in the following eigenvalue problem the positivity of the approximating function is enforced by adding a ReLU function after the last layer of the DenseNet.

In the left panel of Fig. 6.8 we display the approximated eigenfunction along the curve $\{(\kappa, \dots, \kappa)^\top : \kappa \in [0, 2\pi]\}$ and compare it to the reference solution. In the central panel we show the L^2 error with respect to the reference solution evaluated on uniformly sampled test data along the training iterations. The right panel displays the moving average of the absolute value of the eigenvalue taken over a moving window of 100 gradient steps (since the true value is $\lambda = 0$ it is not possible to compute a relative error here). We see that both the eigenfunction and the eigenvalue are approximated sufficiently well.

6.4.2   Nonlinear Schrödinger equation

Let us now consider a nonlinear eigenvalue problem. Again following [30], we consider the nonlinear Schrödinger operator including a cubic term that arises from the Gross-Pitaevskii equation for the single-particle wave function in a Bose-Einstein condensate [27, 69]. To be precise, we consider

$$\Delta V(x) - V^3(x) - \Psi(x) V(x) = -\lambda V(x), \qquad (6.15)$$

where

$$\Psi(x) = -\frac{1}{c^2} \exp\left(\frac{2}{d} \sum_{i=1}^{d} \cos x_i\right) + \sum_{i=1}^{d} \left(\frac{\sin^2(x_i)}{d^2} - \frac{\cos x_i}{d}\right) - 3. \qquad (6.16)$$

One can show that

$$V(x) = \frac{1}{c} \exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos x_i\right) \tag{6.17}$$

is the eigenfunction corresponding to the principal eigenvalue $\lambda = -3$, where c is determined from the normalization $\int_\Omega V^2(x)\, dx = |\Omega|$. We add the latter constraint into the loss function by replacing the term $\mathcal{L}_c(\varphi)$ in (4.7) with $\mathcal{L}_n(\varphi) = (\mathbb{E}[\varphi(X)^2] - 1)^2$, where the distribution of X is uniform on $\Omega = [0, 2\pi]^d$. We again rely on the diffusion loss with a trajectory length of $N = 20$ (i.e. $\mathsf{t} = 0.02$). Fig. 6.9 shows the approximate solution of the eigenfunction in $d = 5$ evaluated along the curve $\{(\kappa, \dots, \kappa)^\top : \kappa \in [0, 2\pi]\}$ as well as its L^2 error and the relative error of the approximated eigenvalue along the training iterations. We observe that both the eigenfunction and the eigenvalue are approximated quite well.

We repeat the experiment in dimension $d = 10$ and display the results in Fig. 6.10. The optimization task gets slightly more difficult, but the resulting eigenfunction and eigenvalue still fit the reference solution reasonably well. Note that in both experiments no explicit boundary conditions, but only the norm constraint and the periodicity are imposed.
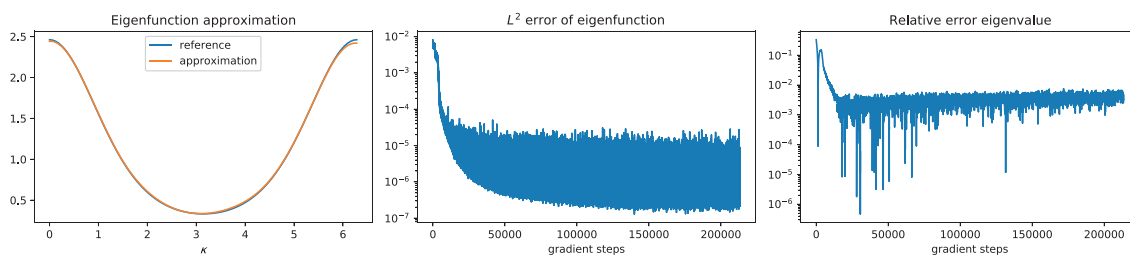

Figure 6.9: Left: Approximation and reference of the eigenfunction corresponding to the principal eigenvalue of the nonlinear Schrödinger operator in $d = 5$. Middle: L^2 error with respect to test data along the training iterations. Right: Relative error of the approximated eigenvalue along the gradient steps.
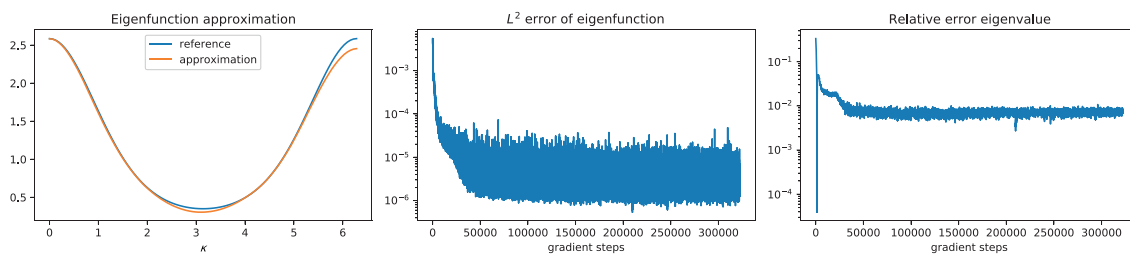

Figure 6.10: Same experiment as in Fig. 6.9 in dimension $d = 10$.

7   Conclusion and outlook

In this paper, we have investigated the relationship between BSDE and PINN based approximation schemes for high-dimensional PDEs through the lens of the novel diffusion loss. In particular, we have shown that the diffusion loss provides an interpolation between the aforementioned methods and demonstrated its promising numerical performance in a range of experiments, allowing us to trade-off strengths and weaknesses of BSDEs and PINNs. Although we believe that the diffusion loss may be a stepping stone towards a unified understanding of computational approaches for high-dimensional PDEs, many questions remain open: First, there is a need for principled insights into the mechanisms with which BSDE, PINN and diffusion losses can or cannot overcome the curse of dimensionality. Second, it is of great practical interest to optimize the algorithmic details, preferably according to well-understood theoretical foundations. In this regard, we mention sophisticated (possibly adaptive) choices of the weights α_{int}, α_{b}, α_{T} and the measures $\nu_{\Omega \times [0,T]}$, ν_Ω, $\nu_{\partial\Omega \times [0,T]}$ as well as variance reduction techniques for estimator versions of the losses (see, for instance, [62, Remark 4.7]). Last but not least, we expect that the concepts explored in this paper may be fruitfully extended to the setting of parameter-dependent PDEs.

Acknowledgments

References

[1] W. F. Ames, Numerical Methods for Partial Differential Equations, Academic Press, 2014.

[2] F. Bach, Breaking the curse of dimensionality with convex neural networks, *J. Mach. Learn. Res.*, **18**(1): 629–681, 2017.

[3] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld, Deep splitting method for parabolic PDEs, *SIAM J. Sci. Comput.*, **43**(5):A3135–A3154, 2021.

[4] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, Solving stochastic differential equations and Kolmogorov equations by means of deep learning, *arXiv:1806.00421*, 2018.

[5] C. Beck, W. E, and A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, *J. Nonlinear Sci.*, **29**(4):1563–1619, 2019.

[6] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck, An overview on deep learning-based approximation methods for partial differential equations, *arXiv:2012.12348*, 2020.

[7] S. Becker, R. Braunwarth, M. Hutzenthaler, A. Jentzen, and P. von Wurstemberger, Numerical simulations for full history recursive multilevel Picard approximations for systems of high-dimensional partial differential equations, *arXiv:2005.10206*, 2020.

[8] H. Berestycki, L. Nirenberg, and S. R. S. Varadhan, The principal eigenvalue and maximum principle for second-order elliptic operators in general domains, *Comm. Pure Appl. Math.*, **47**(1):47–92, 1994.

[9] J. Berg and K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, *Neurocomputing*, **317**:28–41, 2018.

[10] J. Berner, P. Grohs, and A. Jentzen, Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, *SIAM J. Math. Data Sci.*, **2**(3):631–657, 2020.

[11] J.-M. Bismut, Conjugate convex functions in optimal stochastic control, *J. Math. Anal. Appl.*, **44**(2):384–404, 1973.

[12] J. Blechschmidt and O. G. Ernst, Three ways to solve partial differential equations with neural networks-A review, *GAMM-Mitteilungen*, e202100006, 2021.

[13] B. Bouchard and N. Touzi, Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations, *Stochastic Process. Appl.*, **111**(2):175–206, 2004.

[14] B. Bouchard et al., Strong approximations of BSDEs in a domain, *Bernoulli*, **15**(4):1117–1147, 2009.

[15] F. M. Buchmann and W. P. Petersen, Solving Dirichlet problems numerically using the Feynman-Kac representation, *BIT Numer. Math.*, **43**(3):519–540, 2003.

[16] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science*, **355**(6325):602–606, 2017.

[17] T. De Ryck and S. Mishra, Error analysis for physics informed neural networks (PINNs) approximating Kolmogorov PDEs, *arXiv:2106.14473*, 2021.

[18] T. Dockhorn, A discussion on solving partial differential equations using neural networks, *arXiv:1904.07200*, 2019.

[19] W. E, J. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Commun. Math. Stat.*, **5**(4): 349–380, 2017.

[20] W. E and E. Vanden-Eijnden, Towards a theory of transition paths, *J. Stat. Phys.*, **123**(3):503–523, 2006.

[21] W. E and B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, *Commun. Math. Stat.*, **6**(1):1–12, 2018.

[22] W. E et al., Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning, *arXiv:2008.13333*, 2020.

[23] E. Gobet et al., A regression-based Monte Carlo method to solve backward stochastic differential equations, *Ann. Appl. Probab.*, **15**(3):2172–2202, 2005.

[24] P. Grohs and L. Herrmann, Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions, *arXiv:2007.05384*, 2020.

[25] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, *arXiv:1809.02362*, 2018.

[26] P. Grohs, A. Jentzen, and D. Salimova, Deep neural network approximations for Monte Carlo algorithms, *arXiv:1908.10828*, 2019.

[27] E. P. Gross, Structure of a quantized vortex in boson systems, *Nuovo Cim*, **20**(3):454–477, 1961.

[28] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci. USA*, **115**(34):8505–8510, 2018.

[29] J. Han and J. Long, Convergence of the deep BSDE method for coupled FBSDEs, *Probab. Uncertain. Quant. Risk*, **5**(1):1–33, 2020.

[30] J. Han, J. Lu, and M. Zhou, Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach, *J. Comput. Phys.*, **423**:109792, 2020.

[31] J. Han, M. Nica, and A. R. Stinchcombe, A derivative-free method for solving elliptic partial differential equations with deep neural networks, *J. Comput. Phys.*, **419**:109672, 2020.

[32] C. Hartmann, O. Kebiri, L. Neureither, and L. Richter, Variational approach to rare event simulation using least-squares regression, *Chaos*, **29**(6):063107, 2019.

[33] E. Hausenblas, A numerical scheme using Itô excursions for simulating local time resp. stochastic differential equations with reflection, *Osaka J. Math.*, **36**(1):105–137, 1999.

[34] J. Hermann, Z. Schätzle, and F. Noé, Deep-neural-network solution of the electronic Schrödinger equation, *Nat. Chem.*, **12**(10):891–897, 2020.

[35] G. Huang, Z. Liu, L. van Der Maaten, and K. Q. Weinberger, Densely connected convolutional networks, In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708, 2017.

[36] C. Huré, H. Pham, and X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, *Math. Comp.*, **89**(324):1547–1579, 2020.

[37] M. Hutzenthaler et al., Multilevel Picard iterations for solving smooth semilinear parabolic heat equations, *arXiv:1607.03295*, 2016.

[38] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations, *SN Partial Differ. Equ. Appl.*, **1**(2):1–34, 2020.

[39] M. Hutzenthaler, A. Jentzen, T. Kruse, T. A. Nguyen, and P. von Wurstemberger, Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations, *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, **476**(2244):20190630, 2020.

[40] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *J. Comput. Phys.*, **404**:109136, 2020.

[41] A. Jentzen, D. Salimova, and T. Welti, Proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients, *Commun. Math. Sci.*, **19**(5):1167–1205, 2021.

[42] L. Jianyu, L. Siwei, Q. Yingjian, and H. Yaping, Numerical solution of elliptic partial differential equation using radial basis function neural networks, *Neural Netw.*, **16**(5-6):729–734, 2003.

[43] H. Jin, M. Mattheakis, and P. Protopapas, Unsupervised neural networks for quantum eigenvalue problems, *arXiv:2010.05075*, 2020.

[44] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.*, **3**(6):422–440, 2021.

[45] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun (Eds.), 2015.

[46] Y. Khoo, J. Lu, and L. Ying, Solving for high-dimensional committor functions using artificial neural networks, *Res. Math. Sci.*, **6**(1):1, 2019.

[47] P. E. Kloeden and E. Platen, Stochastic differential equations, In: *Numerical Solution of Stochastic Differential Equations*, Springer, 103–160, 1992.

[48] S. Kremsner, A. Steinicke, and M. Szölgyenyi, A deep neural network algorithm for semilinear elliptic PDEs with applications in insurance mathematics, *Risks*, **8**(4):136, 2020.

[49] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural network methods in quantum mechanics, *Comput. Phys. Commun.*, **104**(1-3):1–14, 1997.

[50] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw. Learn. Syst.*, **9**(5):987–1000, 1998.

[51] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, *IEEE Trans. Neural Netw.*, **11**(5):1041–1049, 2000.

[52] H. Lee and I. S. Kang, Neural algorithm for solving differential equations, *J. Comput. Phys.*, **91**(1):110–131, 1990.

[53] Q. Li, B. Lin, and W. Ren, Computing committor functions for the study of rare events using deep learning, *J. Chem. Phys.*, **151**(5):054112, 2019.

[54] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Fourier neural operator for parametric partial differential equations, *arXiv:2010.08895*, 2020.

[55] J. Lu and Y. Lu, A priori generalization error analysis of two-layer neural networks for solving high dimensional Schrödinger eigenvalue problems, *arXiv:2105.01228*, 2021.

[56] J. Lu and J. Nolen, Reactive trajectories and the transition path process, *Probab Theory Relat Field*, **161**(1): 195–244, 2015.

[57] M. Magill, F. Qureshi, and H. W. de Haan, Neural networks trained to solve differential equations learn general representations, *arXiv:1807.00042*, 2018.

[58] C. Martin, H. Zhang, J. Costacurta, M. Nica, and A. R. Stinchcombe, Solving elliptic equations with Brownian motion: bias reduction and temporal difference learning, *Methodol. Comput. Appl. Probab.*,

24(3):1603–1626, 2022.

[59] S. Mishra and R. Molinaro, Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs, *arXiv:2006.16144*, 2020.

[60] J. Müller and M. Zeinhofer, Deep Ritz revisited, *arXiv:1912.03937*, 2019.

[61] J. Müller and M. Zeinhofer, Notes on exact boundary values in residual minimisation, *arXiv:2105.02550*, 2021.

[62] N. Nüsken and L. Richter, Solving high-dimensional Hamilton-Jacobi-Bellman PDEs using neural networks: Perspectives from the theory of controlled diffusions and measures on path space, *SN Partial Differ. Equ. Appl.*, **2**(4):1–48, 2021.

[63] É. Pardoux, Backward stochastic differential equations and viscosity solutions of systems of semilinear parabolic and elliptic PDEs of second order, In: *Stochastic Analysis and Related Topics VI*, Springer, 79–127, 1998.

[64] É. Pardoux and S. Peng, Adapted solution of a backward stochastic differential equation, *Syst. Control. Lett.*, **14**(1):55–61, 1990.

[65] É. Pardoux and S. Zhang, Generalized BSDEs and nonlinear Neumann boundary value problems, *Probab. Theory Related Fields*, **110**(4):535–558, 1998.

[66] G. A. Pavliotis, Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations, Texts in Applied Mathematics, Vol. 60, Springer, 2014.

[67] D. Pfau, J. S. Spencer, A.G. D. G. Matthews, and W. M. C. Foulkes, Ab initio solution of the many-electron Schrödinger equation with deep neural networks, *Phys. Rev. Res.*, **2**(3):033429, 2020.

[68] H. Pham, Continuous-time stochastic control and optimization with financial applications, Springer Science & Business Media, Vol. 61, 2009.

[69] L. P. Pitaevskii, Vortex lines in an imperfect Bose gas, *Sov. Phys. JETP*, **13**(2):451–454, 1961.

[70] M. Raissi, Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations, *arXiv:1804.07010*, 2018.

[71] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations, *arXiv:1711.10561*, 2017.

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

[73] L. Richter, *Solving High-Dimensional PDEs, Approximation of Path Space Measures and Importance Sampling of Diffusions*, PhD Thesis, BTU Cottbus-Senftenberg, 2021.

[74] L. Richter and J. Berner, Robust SDE-based variational formulations for solving linear PDEs via deep learning, In: International Conference on Machine Learning, *Proc. Mach. Learn. Res. (PMLR)*, 18649–18666, 2022.

[75] L. Richter, L. Sallandt, and N. Nüsken, Solving high-dimensional parabolic PDEs using the tensor train format, In: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, M. Meila and T. Zhang (Eds.), *Proc. Mach. Learn. Res. (PMLR)*, **139**:8998–9009, 2021.

[76] G. M. Rotskoff and E. Vanden-Eijnden, Learning with rare data: Using active importance sampling to optimize objectives dominated by rare events, *arXiv:2008.06334*, 2020.

[77] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.*, **375**:1339–1364, 2018.

[78] C. Szepesvári, Algorithms for reinforcement learning, *Synth. Lect. Artif. Intell. Mach. Learn.*, **4**(1):1–103, 2010.

[79] T. Uchiyama and N. Sonehara, Solving inverse problems in nonlinear PDEs by recurrent neural networks, In: IEEE International Conference on Neural Networks, *IEEE*, 99–102, 1993.

[80] R. van der Meer, C. Oosterlee, and A. Borovykh, Optimally weighted loss functions for solving PDEs with neural networks, *arXiv:2002.06269*, 2020.

[81] S. Wang, Y. Teng, and P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.*, **43**(5):A3055–A3081, 2021.

[82] S. Wang, H. Wang, and P. Perdikaris, On the eigenvector bias of Fourier feature networks: From regres-

sion to solving multi-scale PDEs with physics-informed neural networks, *Comput. Methods Appl. Mech. Engrg.*, **384**:113938, 2021.

[83] S. Wang, X. Yu, and P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, *J. Comput. Phys.*, **449**:110768, 2022.

[84] Y. Zang, G. Bao, X. Ye, and H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, *J. Comput. Phys.*, **411**:109409, 2020.

[85] J. Zhang, Backward Stochastic Differential Equations, Springer, 2017.

[86] W. Zhang, T. Li, and C. Schütte, Solving eigenvalue PDEs of metastable diffusion processes using artificial neural networks, *arXiv:2110.14523*, 2021.

[87] 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 J. Sci. Comput.*, **43**(6):A4043–A4066, 2021.