Commun. Math. Stat. (2017) 5:349–380
https://doi.org/10.1007/s40304-017-0117-6

Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations

Weinan E[1,2,3] · **Jiequn Han**[2] · **Arnulf Jentzen**[4]

Received: 19 July 2017 / Accepted: 23 August 2017 / Published online: 10 November 2017

Abstract We study a new algorithm for solving parabolic partial differential equations (PDEs) and backward stochastic differential equations (BSDEs) in high dimension, which is based on an analogy between the BSDE and reinforcement learning with the gradient of the solution playing the role of the policy function, and the loss function given by the error between the prescribed terminal condition and the solution of the BSDE. The policy function is then approximated by a neural network, as is done in deep reinforcement learning. Numerical results using TensorFlow illustrate the efficiency and accuracy of the studied algorithm for several 100-dimensional nonlinear PDEs from physics and finance such as the Allen–Cahn equation, the Hamilton–Jacobi–Bellman equation, and a nonlinear pricing model for financial derivatives.

Keywords PDEs · High dimension · Backward stochastic differential equations · Deep learning · Control · Feynman-Kac

Mathematics Subject Classification 65M75 · 60H35 · 65C30

✉ Weinan E
weinan@math.princeton.edu

Jiequn Han
jiequnh@princeton.edu

Arnulf Jentzen
arnulf.jentzen@sam.math.ethz.ch

1 Beijing Institute of Big Data Research, Beijing, China

2 Princeton University, Princeton, NJ, USA

3 Peking University, Beijing, China

4 ETH Zurich, Zurich, Switzerland

1 Introduction

Developing efficient numerical algorithms for high-dimensional (say, hundreds of dimensions) partial differential equations (PDEs) has been one of the most challenging tasks in applied mathematics. As is well known, the difficulty lies in the "curse of dimensionality" [1], namely, as the dimensionality grows, the complexity of the algorithms grows exponentially. For this reason, there are only a limited number of cases where practical high-dimensional algorithms have been developed (cf., e.g., [9,12,13,20–22]). For linear parabolic PDEs, one can use the Feynman–Kac formula and Monte Carlo methods to develop efficient algorithms to evaluate solutions at any given space–time locations. For a class of inviscid Hamilton–Jacobi equations, Darbon and Osher have recently developed an algorithm which performs numerically well in the case of such high-dimensional inviscid Hamilton–Jacobi equations (see [9]). Darbon and Osher's algorithm is based on results from compressed sensing and on the Hopf formulas for the Hamilton–Jacobi equations. A general algorithm for (nonlinear) parabolic PDEs based on the Feynman–Kac and Bismut–Elworthy–Li formula and a multilevel decomposition of Picard iteration was developed in [12] and has been shown to be quite efficient on a number examples in finance and physics (cf. [13]). The complexity of the algorithm is essentially shown to be $O(d\varepsilon^{-4})$ for certain semilinear heat equations, where d is the dimensionality of the problem and ε is the required accuracy.

In recent years, a new class of techniques, called deep learning, have emerged in machine learning and have proven to be very effective in dealing with a large class of high-dimensional problems in computer vision (cf., e.g., [26]), natural language processing (cf., e.g., [23]), time series analysis, etc. (cf., e.g., [17,27]). This success fuels in speculations that deep learning might hold the key to solve the curse of dimensionality problem. It should be emphasized that, at the present time, there are no theoretical results that support such claims although the practical success of deep learning has been astonishing. However, this should not prevent us from trying to apply deep learning to other problems where the curse of dimensionality has been the issue.

In our recent work Han et al. [19] we proposed and sketched the derivation of a new algorithm which explores the use of deep learning for solving general high-dimensional PDEs. To this end, it is necessary to formulate the PDEs as a learning problem. Motivated by ideas in [18] where deep learning-based algorithms were developed for high-dimensional stochastic control problems, we explore a connection between (nonlinear) parabolic PDEs and backward stochastic differential equations (BSDEs) (see [28,29,31]) since BSDEs share a lot of common features with stochastic control problems. In this paper, we generalize and specify the new algorithm (see Sects. 3 and 5), we detail the derivation of the algorithm (see Sect. 2), and we also test the algorithm numerically in the case of several 100-dimensional example PDEs from the finance, physics, and economics literature (see Sect. 4). The numerical tests suggest that the algorithm works in the case of the considered example PDEs very satisfactory in terms of both accuracy and speed.

2 Main Ideas of the Algorithm

We will consider a fairly general class of nonlinear parabolic PDEs (see (4.1) in Subsect. 4.1). The proposed algorithm is based on the following set of ideas:

(i) Through the so-called nonlinear Feynman–Kac formula, we can formulate the PDEs equivalently as BSDEs.
(ii) One can view the BSDE as a stochastic control problem with the gradient of the solution being the policy function. These stochastic control problems can then be viewed as model-based reinforcement learning problems.
(iii) The (high-dimensional) policy function can then be approximated by a deep neural network, as has been done in deep reinforcement learning.

Instead of formulating initial value problems, as is commonly done in the PDE literature, we consider the setup with terminal conditions since this facilitates making connections with BSDEs. Terminal value problems can obviously be transformed to initial value problems and vice versa.

In the remainder of this section, we present a rough sketch of the derivation of the proposed algorithm, which we refer to as deep BSDE method. In this derivation, we restrict ourselves to a specific class of nonlinear PDEs; that is, we restrict ourselves to semilinear heat equations (see (PDE) below) and refer to Subsects. 3.2 and 4.1 for the general introduction of the deep BSDE method.

2.1 An Example: A Semilinear Heat Partial Differential Equation (PDE)

Let $T \in (0, \infty)$, $d \in \mathbb{N}$, $\xi \in \mathbb{R}^d$, let $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ be continuous functions, and let $u = (u(t, x))_{t\in[0,T],x\in\mathbb{R}^d} \in C^{1,2}([0, T] \times \mathbb{R}^d, \mathbb{R})$ satisfy for all $t \in [0, T]$, $x \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t, x) + \frac{1}{2}(\Delta_x u)(t, x) + f\big(u(t, x), (\nabla_x u)(t, x)\big) = 0. \tag{PDE}$$

A key idea of this work is to reformulate the PDE (PDE) as an appropriate stochastic control problem.

2.2 Formulation of the PDE as a Suitable Stochastic Control Problem

More specifically, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, let $W : [0, T] \times \Omega \to \mathbb{R}^d$ be a d-dimensional standard Brownian motion on $(\Omega, \mathcal{F}, \mathbb{P})$, let $\mathbb{F} = (\mathbb{F}_t)_{t\in[0,T]}$ be the normal filtration on $(\Omega, \mathcal{F}, \mathbb{P})$ generated by W, let $\mathcal{A}$ be the set of all $\mathbb{F}$-adapted $\mathbb{R}^d$-valued stochastic processes with continuous sample paths, and for every $y \in \mathbb{R}$ and every $Z \in \mathcal{A}$ let $Y^{y,Z} : [0, T] \times \Omega \to \mathbb{R}$ be a $\mathbb{F}$-adapted stochastic process with continuous sample paths which satisfies that for all $t \in [0, T]$ it holds $\mathbb{P}$-a.s. that

$$Y_t^{y,Z} = y - \int_0^t f\big(Y_s^{y,Z}, Z_s\big)\, ds + \int_0^t \langle Z_s, dW_s\rangle_{\mathbb{R}^d}. \tag{2.1}$$

We now view the solution $u \in C^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ of (PDE) and its spatial derivative as the solution of a stochastic control problem associated with (2.1). More formally, under suitable regularity hypotheses on the nonlinearity f, it holds that the pair consisting of $u(0,\xi)\in\mathbb{R}$ and $((\nabla_x u)(t,\xi+W_t))_{t\in[0,T]}\in\mathcal{A}$ is the (up to indistinguishability) unique global minimum of the function

$$\mathbb{R}\times\mathcal{A}\ni(y,Z)\mapsto\mathbb{E}\big[|Y_T^{y,Z}-g(\xi+W_T)|^2\big]\in[0,\infty]. \tag{2.2}$$

One can also view the stochastic control problem (2.1)–(2.2) (with Z being the control) as a model-based reinforcement learning problem. In that analogy, we view Z as the policy and we approximate $Z\in\mathcal{A}$ using feedforward neural networks (see (2.11) and Sect. 4 for further details). The process $u(t,\xi+W_t)$, $t\in[0,T]$, corresponds to the value function associated with the stochastic control problem and can be computed approximately by employing the policy Z (see (2.9) for details). The connection between the PDE (PDE) and the stochastic control problem (2.1)–(2.2) is based on the nonlinear Feynman–Kac formula which links PDEs and BSDEs (see (BSDE) and (2.3)).

2.3 The Nonlinear Feynman–Kac Formula

Let $Y\colon[0,T]\times\Omega\to\mathbb{R}$ and $Z\colon[0,T]\times\Omega\to\mathbb{R}^d$ be $\mathbb{F}$-adapted stochastic processes with continuous sample paths which satisfy that for all $t\in[0,T]$ it holds $\mathbb{P}$-a.s. that

$$Y_t=g(\xi+W_T)+\int_t^T f(Y_s,Z_s)\,ds-\int_t^T\langle Z_s,dW_s\rangle_{\mathbb{R}^d}. \tag{BSDE}$$

Under suitable additional regularity assumptions on the nonlinearity f, we have that the nonlinear parabolic PDE (PDE) is related to the BSDE (BSDE) in the sense that for all $t\in[0,T]$ it holds $\mathbb{P}$-a.s. that

$$Y_t=u(t,\xi+W_t)\in\mathbb{R}\qquad\text{and}\qquad Z_t=(\nabla_x u)(t,\xi+W_t)\in\mathbb{R}^d \tag{2.3}$$

(cf., e.g., [29, Section 3] and [30]). The first identity in (2.3) is sometimes referred to as nonlinear Feynman–Kac formula in the literature.

2.4 Forward Discretization of the Backward Stochastic Differential Equation (BSDE)

To derive the deep BSDE method, we first plug the second identity in (2.3) into (BSDE) to obtain that for all $t\in[0,T]$ it holds $\mathbb{P}$-a.s. that

$$\begin{aligned}Y_t&=g(\xi+W_T)+\int_t^T f\big(Y_s,(\nabla_x u)(s,\xi+W_s)\big)\,ds\\&\quad-\int_t^T\langle(\nabla_x u)(s,\xi+W_s),dW_s\rangle_{\mathbb{R}^d}.\end{aligned} \tag{2.4}$$

In particular, we obtain that for all $t_1, t_2 \in [0, T]$ with $t_1 \leq t_2$ it holds $\mathbb{P}$-a.s. that

$$\begin{aligned} Y_{t_2} = Y_{t_1} - \int_{t_1}^{t_2} f\big(Y_s, (\nabla_x u)(s, \xi + W_s)\big)\, ds \\ + \int_{t_1}^{t_2} \langle (\nabla_x u)(s, \xi + W_s), dW_s \rangle_{\mathbb{R}^d}. \end{aligned} \tag{2.5}$$

Next we apply a time discretization to (2.5). More specifically, let $N \in \mathbb{N}$ and let $t_0, t_1, \ldots, t_N \in [0, T]$ be real numbers which satisfy

$$0 = t_0 < t_1 < \ldots < t_N = T \tag{2.6}$$

and observe that (2.5) suggests for $N \in \mathbb{N}$ sufficiently large that

$$\begin{aligned} &Y_{t_{n+1}} \\ &\quad \approx Y_{t_n} - f\big(Y_{t_n}, (\nabla_x u)(t_n, \xi + W_{t_n})\big)\,(t_{n+1} - t_n) \\ &\quad + \langle (\nabla_x u)(t_n, \xi + W_{t_n}), W_{t_{n+1}} - W_{t_n} \rangle_{\mathbb{R}^d}. \end{aligned} \tag{2.7}$$

2.5 Deep Learning-Based Approximations

In the next step, we employ a deep learning approximation for

$$(\nabla_x u)(t_n, x) \in \mathbb{R}^d, \qquad x \in \mathbb{R}^d, \qquad n \in \{0, 1, \ldots, N\}, \tag{2.8}$$

but not for $u(t_n, x) \in \mathbb{R}$, $x \in \mathbb{R}^d$, $n \in \{0, 1, \ldots, N\}$. Approximations for $u(t_n, x) \in \mathbb{R}$, $x \in \mathbb{R}^d$, $n \in \{0, 1, \ldots, N\}$, in turn, can be computed recursively by using (2.7) together with deep learning approximations for (2.8). More specifically, let $\rho \in \mathbb{N}$, let $\mathcal{U}^\theta \in \mathbb{R}$, $\theta \in \mathbb{R}^\rho$, be real numbers, let $\mathcal{V}_n^\theta \colon \mathbb{R}^d \to \mathbb{R}^d$, $n \in \{0, 1, \ldots, N-1\}$, $\theta \in \mathbb{R}^\rho$, be continuous functions, and let $\mathcal{Y}^\theta \colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}$, $\theta \in \mathbb{R}^\rho$, be stochastic processes which satisfy for all $\theta \in \mathbb{R}^\rho$, $n \in \{0, 1, \ldots, N-1\}$ that $\mathcal{Y}_0^\theta = \mathcal{U}^\theta$ and

$$\begin{aligned} \mathcal{Y}_{n+1}^\theta = \mathcal{Y}_n^\theta - f\big(\mathcal{Y}_n^\theta, \mathcal{V}_n^\theta(\xi + W_{t_n})\big)\,(t_{n+1} - t_n) \\ + \langle \mathcal{V}_n^\theta(\xi + W_{t_n}), W_{t_{n+1}} - W_{t_n} \rangle_{\mathbb{R}^d}. \end{aligned} \tag{2.9}$$

We think of $\rho \in \mathbb{N}$ as the number of parameters in the neural network, for all appropriate $\theta \in \mathbb{R}^\rho$ we think of $\mathcal{U}^\theta \in \mathbb{R}$ as suitable approximations

$$\mathcal{U}^\theta \approx u(0, \xi) \tag{2.10}$$

of $u(0, \xi)$, and for all appropriate $\theta \in \mathbb{R}^\rho$, $x \in \mathbb{R}^d$, $n \in \{0, 1, \ldots, N-1\}$ we think of $\mathcal{V}_n^\theta(x) \in \mathbb{R}^d$ as suitable approximations

$$\mathcal{V}_n^\theta(x) \approx (\nabla_x u)(t_n, x) \tag{2.11}$$

of $(\nabla_x u)(t_n, x)$.

2.6 Stochastic Optimization Algorithms

The "appropriate" $\theta \in \mathbb{R}^\rho$ can be obtained by minimizing the expected loss function through stochastic gradient descent-type algorithms. For the loss function, we pick the squared approximation error associated with the terminal condition of the BSDE (BSDE). More precisely, let $\Lambda \in \mathbb{R}^\rho$ be a suitable local minimum of the function

$$\mathbb{R}^\rho \ni \theta \mapsto \mathbb{E}\big[|\mathcal{Y}_N^\theta - g(\xi + W_T)|^2\big] \in [0, \infty]. \tag{2.12}$$

Minimizing the function (2.12) is inspired by the fact that

$$\mathbb{E}\left[|Y_T - g(\xi + W_T)|^2\right] = 0 \tag{2.13}$$

according to (BSDE) above (cf. (2.2) above). Under suitable regularity assumptions, we approximate the vector $\Lambda \in \mathbb{R}^\rho$ through stochastic gradient descent-type approximation methods, and thereby we obtain random approximations $\Theta_0, \Theta_1, \Theta_2, \ldots : \Omega \to \mathbb{R}^\rho$ of $\Lambda \in \mathbb{R}^\rho$. For sufficiently large $N, \rho, m \in \mathbb{N}$, we then employ the random variable $\mathcal{U}^{\Theta_m} : \Omega \to \mathbb{R}$ as a suitable implementable approximation

$$\mathcal{U}^{\Theta_m} \approx u(0, \xi) \tag{2.14}$$

of $u(0, \xi)$ (cf. (2.10) above), and for sufficiently large $N, \rho, m \in \mathbb{N}$ and all $x \in \mathbb{R}^d$, $n \in \{0, 1, \ldots, N-1\}$, we use the random variable $\mathcal{V}_n^{\Theta_m}(x) \colon \Omega \to \mathbb{R}^d$ as a suitable implementable approximation

$$\mathcal{V}_n^{\Theta_m}(x) \approx (\nabla_x u)(t_n, x) \tag{2.15}$$

of $(\nabla_x u)(t_n, x)$ (cf. (2.11) above). In the next section the proposed approximation method is described in more detail.

To simplify the presentation, we have restricted us in (PDE), (2.1), (2.2), (BSDE) above and Subsect. 3.1 to semilinear heat equations. We refer to Subsect. 3.2 and Sect. 4 for the general description of the deep BSDE method.

3 Details of the Algorithm

3.1 Formulation of the Proposed Algorithm in the Case of Semilinear Heat Equations

In this subsection, we describe the proposed algorithm in the specific situation where (PDE) is the PDE under consideration, where batch normalization (see Ioffe and Szegedy [24]) is not employed, and where the plain-vanilla stochastic gradient descent method with a constant learning rate $\gamma \in (0, \infty)$ and without mini-batches is the employed stochastic approximation algorithm. The general framework, which includes the setting in this subsection as a special case, is found in Subsect. 3.2.

Framework 3.1 (Specific case). *Let* $T, \gamma \in (0, \infty)$, $d, \rho, N \in \mathbb{N}$, $\xi \in \mathbb{R}^d$, *let* $f\colon \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ *and* $g\colon \mathbb{R}^d \to \mathbb{R}$ *be functions, let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space, let* $W^m\colon [0,T] \times \Omega \to \mathbb{R}^d$, $m \in \mathbb{N}_0$, *be independent* d*-dimensional standard Brownian motions on* $(\Omega, \mathcal{F}, \mathbb{P})$, *let* $t_0, t_1, \ldots, t_N \in [0, T]$ *be real numbers with*

$$0 = t_0 < t_1 < \ldots < t_N = T, \tag{3.1}$$

for every $\theta \in \mathbb{R}^\rho$ *let* $\mathcal{U}^\theta \in \mathbb{R}$, *for every* $\theta \in \mathbb{R}^\rho$, $n \in \{0, 1, \ldots, N-1\}$, *let* $\mathcal{V}^\theta_n\colon \mathbb{R}^d \to \mathbb{R}^d$ *be a function, for every* $m \in \mathbb{N}_0$, $\theta \in \mathbb{R}^\rho$ *let* $\mathcal{Y}^{\theta,m}\colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}$ *be a stochastic process which satisfies for all* $n \in \{0, 1, \ldots, N-1\}$ *that* $\mathcal{Y}^{\theta,m}_0 = \mathcal{U}^\theta$ *and*

$$\begin{aligned}\mathcal{Y}^{\theta,m}_{n+1} = \mathcal{Y}^{\theta,m}_n &- f\big(\mathcal{Y}^{\theta,m}_n, \mathcal{V}^\theta_n(\xi + W^m_{t_n})\big)\,(t_{n+1} - t_n)\\ &+ \langle \mathcal{V}^\theta_n(\xi + W^m_{t_n}), W^m_{t_{n+1}} - W^m_{t_n}\rangle_{\mathbb{R}^d},\end{aligned} \tag{3.2}$$

for every $m \in \mathbb{N}_0$ *let* $\phi^m\colon \mathbb{R}^\rho \times \Omega \to \mathbb{R}$ *be the function which satisfies for all* $\theta \in \mathbb{R}^\rho$, $\omega \in \Omega$ *that*

$$\phi^m(\theta, \omega) = \big|\mathcal{Y}^{\theta,m}_N(\omega) - g(\xi + W^m_T(\omega))\big|^2, \tag{3.3}$$

for every $m \in \mathbb{N}_0$ *let* $\Phi^m\colon \mathbb{R}^\rho \times \Omega \to \mathbb{R}^\rho$ *be a function which satisfies for all* $\omega \in \Omega$, $\theta \in \{v \in \mathbb{R}^\rho\colon (\mathbb{R}^\rho \ni w \mapsto \phi^m(w, \omega) \in \mathbb{R}$ *is differentiable at* $v \in \mathbb{R}^\rho)\}$ *that*

$$\Phi^m(\theta, \omega) = (\nabla_\theta \phi^m)(\theta, \omega), \tag{3.4}$$

and let $\Theta\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\rho$ *be a stochastic process which satisfies for all* $m \in \mathbb{N}$ *that*

$$\Theta_m = \Theta_{m-1} - \gamma \cdot \Phi^m(\Theta_{m-1}). \tag{3.5}$$

Under suitable further hypotheses (cf. Sects. 4 and 5), we think in the case of sufficiently large $\rho, N, m \in \mathbb{N}$ and sufficiently small $\gamma \in (0, \infty)$ of $\mathcal{U}^{\Theta_m} \in \mathbb{R}$ as an appropriate approximation

$$\mathcal{U}^{\Theta_m} \approx u(0, \xi) \tag{3.6}$$

of the solution $u(t, x) \in \mathbb{R}$, $(t, x) \in [0, T] \times \mathbb{R}^d$, of the PDE

$$\frac{\partial u}{\partial t}(t, x) + \frac{1}{2}(\Delta_x u)(t, x) + f\big(u(t, x), (\nabla_x u)(t, x)\big) = 0 \tag{3.7}$$

for $(t, x) \in [0, T] \times \mathbb{R}^d$.

3.2 Formulation of the Proposed Algorithm in the General Case

Framework 3.2 (General case). *Let* $T \in (0, \infty)$, $d, k, \rho, \varrho, N, \varsigma \in \mathbb{N}$, $\xi \in \mathbb{R}^d$, *let* $f\colon [0, T] \times \mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^{k \times d} \to \mathbb{R}^k$, $g\colon \mathbb{R}^d \to \mathbb{R}^k$, *and* $\Upsilon\colon [0, T]^2 \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ *be functions, let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space, let* $W^{m,j}\colon [0, T] \times \Omega \to \mathbb{R}^d$,

$m, j \in \mathbb{N}_0$, be independent d-dimensional standard Brownian motions on $(\Omega, \mathcal{F}, \mathbb{P})$, let $t_0, t_1, \ldots, t_N \in [0, T]$ be real numbers with

$$0 = t_0 < t_1 < \ldots < t_N = T, \tag{3.8}$$

for every $\theta \in \mathbb{R}^\rho$ let $\mathcal{U}^\theta \in \mathbb{R}^k$, for every $\theta \in \mathbb{R}^\rho$, $\mathbf{s} \in \mathbb{R}^\varsigma$, $n \in \{0, 1, \ldots, N-1\}$, $j \in \mathbb{N}_0$ let $\mathcal{V}^{\theta,\mathbf{s}}_{n,j} \colon (\mathbb{R}^d)^{\mathbb{N}} \to \mathbb{R}^{k \times d}$ be a function, for every $m, j \in \mathbb{N}_0$ let $\mathcal{X}^{m,j} \colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}^d$ and $\mathcal{Y}^{\theta,\mathbf{s},m,j} \colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}^k$, $\theta \in \mathbb{R}^\rho$, $\mathbf{s} \in \mathbb{R}^\varsigma$, be stochastic processes which satisfy for all $\theta \in \mathbb{R}^\rho$, $\mathbf{s} \in \mathbb{R}^\varsigma$, $n \in \{0, 1, \ldots, N-1\}$ that

$$\mathcal{X}^{m,j}_0 = \xi, \qquad \mathcal{Y}^{\theta,\mathbf{s},m,j}_0 = \mathcal{U}^\theta, \qquad \mathcal{X}^{m,j}_{n+1} = \Upsilon\big(t_n, t_{n+1}, \mathcal{X}^{m,j}_n, W^{m,j}_{t_{n+1}} - W^{m,j}_{t_n}\big), \tag{3.9}$$

and

$$\begin{aligned}\mathcal{Y}^{\theta,\mathbf{s},m,j}_{n+1} &= \mathcal{Y}^{\theta,\mathbf{s},m,j}_n - f\big(t_n, \mathcal{X}^{m,j}_n, \mathcal{Y}^{\theta,\mathbf{s},m,j}_n, \mathcal{V}^{\theta,\mathbf{s}}_{n,j}(\{\mathcal{X}^{m,i}_n\}_{i\in\mathbb{N}})\big)(t_{n+1} - t_n) \\ &\quad + \mathcal{V}^{\theta,\mathbf{s}}_{n,j}\left(\left\{\mathcal{X}^{m,i}_n\right\}_{i\in\mathbb{N}}\right)\left(W^{m,j}_{t_{n+1}} - W^{m,j}_{t_n}\right),\end{aligned} \tag{3.10}$$

for every $m, j \in \mathbb{N}_0$, $\mathbf{s} \in \mathbb{R}^\varsigma$ let $\phi^{m,j}_{\mathbf{s}} \colon \mathbb{R}^\rho \times \Omega \to \mathbb{R}$ be the function which satisfies for all $\theta \in \mathbb{R}^\rho$, $\omega \in \Omega$ that

$$\phi^{m,j}_{\mathbf{s}}(\theta, \omega) = \|\mathcal{Y}^{\theta,\mathbf{s},m,j}_N(\omega) - g(\mathcal{X}^{m,j}_N(\omega))\|^2_{\mathbb{R}^k}, \tag{3.11}$$

for every $m, j \in \mathbb{N}_0$, $\mathbf{s} \in \mathbb{R}^\varsigma$ let $\Phi^{m,j}_{\mathbf{s}} \colon \mathbb{R}^\rho \times \Omega \to \mathbb{R}^\rho$ be a function which satisfies for all $\omega \in \Omega$, $\theta \in \{v \in \mathbb{R}^\rho \colon (\mathbb{R}^\rho \ni w \mapsto \phi^{m,j}_{\mathbf{s}}(w, \omega) \in \mathbb{R}$ is differentiable at $v \in \mathbb{R}^\rho)\}$ that

$$\Phi^{m,j}_{\mathbf{s}}(\theta, \omega) = (\nabla_\theta \phi^{m,j}_{\mathbf{s}})(\theta, \omega), \tag{3.12}$$

let $\mathcal{S} \colon \mathbb{R}^\varsigma \times \mathbb{R}^\rho \times (\mathbb{R}^d)^{\{0,1,\ldots,N-1\}\times\mathbb{N}} \to \mathbb{R}^\varsigma$ be a function, for every $m \in \mathbb{N}$ let $\psi_m \colon \mathbb{R}^\varrho \to \mathbb{R}^\rho$ and $\Psi_m \colon \mathbb{R}^\varrho \times (\mathbb{R}^\rho)^{\mathbb{N}} \to \mathbb{R}^\varrho$ be functions, and let $\mathbb{S} \colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\varsigma$, $\Xi \colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\varrho$, and $\Theta \colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\rho$ be stochastic processes which satisfy for all $m \in \mathbb{N}$ that

$$\mathbb{S}_m = \mathcal{S}\big(\mathbb{S}_{m-1}, \Theta_{m-1}, \{\mathcal{X}^{m-1,i}_n\}_{(n,i)\in\{0,1,\ldots,N-1\}\times\mathbb{N}}\big), \tag{3.13}$$

$$\Xi_m = \Psi_m\big(\Xi_{m-1}, \{\Phi^{m-1,j}_{\mathbb{S}_m}(\Theta_{m-1})\}_{j\in\mathbb{N}}\big), \qquad \text{and}$$

$$\Theta_m = \Theta_{m-1} - \psi_m(\Xi_m). \tag{3.14}$$

In the next remark, we briefly describe the PDE that the approximation scheme in Framework 3.2 aims to approximate and we also sketch the role of $\mathcal{U}^\theta$, $\theta \in \mathbb{R}^\rho$, and $\mathcal{V}^{\theta,\mathbf{s}}_{n,j} \colon (\mathbb{R}^d)^{\mathbb{N}} \to \mathbb{R}^{k\times d}$, $\theta \in \mathbb{R}^\rho$, $s \in \mathbb{R}^\varsigma$, $n \in \{0, 1, \ldots, N-1\}$, $j \in \mathbb{N}_0$, appearing in Framework 3.2.

Remark 3.3 Consider the setting in Framework 3.2, let $\mu\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}^d$ and $\sigma\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}^{d\times d}$ be functions, assume that $k=1$, assume for all $s,t\in[0,T]$, $x,w\in\mathbb{R}^d$ that

$$\Gamma(s,t,x,w)=x+\mu(s,x)\,(t-s)+\sigma(t,x)w \tag{3.15}$$

(Euler-Maruyama scheme), and let $u\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}$ be a continuous function which satisfies for all $(t,x)\in[0,T)\times\mathbb{R}^d$ that $u|_{[0,T)\times\mathbb{R}^d}\in C^{1,2}([0,T)\times\mathbb{R}^d,\mathbb{R})$, $u(T,x)=g(x)$, and

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x)+\frac{1}{2}\operatorname{Trace}\big(\sigma(t,x)\,[\sigma(t,x)]^*\,(\operatorname{Hess}_x u)(t,x)\big)+\langle\mu(t,x),(\nabla_x u)(t,x)\rangle\\&\quad+f\big(t,x,u(t,x),[(\nabla_x u)(t,x)]^*\,\sigma(t,x)\big)=0.\end{aligned}\tag{3.16}$$

Under suitable further hypotheses (cf. Sects. 4 and 5), we then think in the case of sufficiently large $\rho,N,m\in\mathbb{N}$ of $\mathcal{U}^{\Theta_m}\in\mathbb{R}$ as an appropriate approximation

$$\mathcal{U}^{\Theta_m}\approx u(0,\xi) \tag{3.17}$$

of $u(0,\xi)$, and we then think in the case of sufficiently large $\rho,N\in\mathbb{N}$, appropriate $\theta\in\mathbb{R}^\rho$, $\mathbf{s}\in\mathbb{R}^\varsigma$, and all $n\in\{0,1,\dots,N-1\}$, $m,j\in\mathbb{N}_0$ of $\mathcal{V}^{\theta,\mathbf{s}}_{n,j}(\{\mathcal{X}^{m,i}_n\}_{i\in\mathbb{N}})$ as suitable approximations

$$\mathcal{V}^{\theta,\mathbf{s}}_{n,j}\Big(\big\{\mathcal{X}^{m,i}_n\big\}_{i\in\mathbb{N}}\Big)\approx\Big[(\nabla_x u)\Big(t_n,\mathcal{X}^{m,j}_n\Big)\Big]^*\,\sigma\Big(t_n,\mathcal{X}^{m,j}_n\Big)\in\mathbb{R}^{k\times d}=\mathbb{R}^{1\times d} \tag{3.18}$$

of $[(\nabla_x u)(t_n,\mathcal{X}^{m,j}_n)]^*\,\sigma(t_n,\mathcal{X}^{m,j}_n)$.

3.3 Comments on the Proposed Algorithm

The dynamics in (3.9) associated with the stochastic processes $(\mathcal{X}^{m,j}_n)_{n\in\{0,1,\dots,N\}}$ for $m,j\in\mathbb{N}_0$ allows us to incorporate different algorithms for the discretization of the considered forward stochastic differential equation (SDE) into the deep BSDE method in Subsect. 3.2. The dynamics in (3.14) associated with the stochastic processes Ξ_m, $m\in\mathbb{N}_0$, and Θ_m, $m\in\mathbb{N}_0$, allows us to incorporate different stochastic approximation algorithms such as

- stochastic gradient descent with or without mini-batches (see Subsect. 5.1) as well as
- adaptive moment estimation (Adam) with mini-batches (see Kingma and Jimmy [25] and Subsect. 5.2) into the deep BSDE method in Subsect. 3.2.

The dynamics in (3.13) associated with the stochastic process $\mathbb{S}_m$, $m\in\mathbb{N}_0$, allows us to incorporate the standardization procedure in batch normalization (see Ioffe and Szegedy [24] and also Sect. 4) into the deep BSDE method in Subsect. 3.2. In that case, we think of $\mathbb{S}_m$, $m\in\mathbb{N}_0$, as approximately calculated means and standard deviations.

4 Examples for Nonlinear Partial Differential Equations (PDEs)and Nonlinear Backward Stochastic Differential Equations (BSDEs)

In this section, we illustrate the algorithm proposed in Subsect. 3.2 using several concrete example PDEs. In the examples below, we will employ the general approximation method in Subsect. 3.2 in conjunction with the Adam optimizer (cf. Example 5.2 and Kingma and Ba [25]) with mini-batches with 64 samples in each iteration step (see Subsect. 4.1 for a detailed description).

In our implementation, we employ $N-1$ fully-connected feedforward neural networks to represent $\mathcal{V}^{\theta}_{n,j}$ for $n \in \{1, 2, \ldots, N-1\}$, $j \in \{1, 2, \ldots, 64\}$, $\theta \in \mathbb{R}^{\rho}$ (cf. also Fig. 1 for a rough sketch of the architecture of the deep BSDE method). Each of the neural networks consists of four layers (one input layer [d-dimensional], two hidden layers [both $d+10$-dimensional], and one output layer [d-dimensional]). The number of hidden units in each hidden layer is equal to $d+10$. We also adopt batch normalization (BN) (see Ioffe and Szegedy [24]) right after each matrix multiplication and before activation. We employ the rectifier function $\mathbb{R} \ni x \mapsto \max\{0, x\} \in [0, \infty)$ as our activation function for the hidden variables. All the weights in the network are initialized using a normal or a uniform distribution without any pre-training. Each of the numerical experiments presented below is performed in PYTHON using TENSORFLOW on a MACBOOK PRO with a 2.90 Gigahertz (GHz) INTEL CORE i5 microprocessor and 16 gigabytes (GB) of 1867 megahertz (MHz) double-data-rate-type three synchronous dynamic random-access memory (DDR3-SDRAM). We also refer to the PYTHON code 1 in Subsection 6.1 in [11] for an implementation of the deep BSDE method in the case of the 100-dimensional Allen–Cahn PDE (4.6). Further PYTHON source codes for each of the example PDEs below can be found at https://github.com/frankhan91/DeepBSDE.

4.1 Setting

Assume the setting in Framework 3.2, and assume for all $\theta = (\theta_1, \ldots, \theta_\rho) \in \mathbb{R}^{\rho}$, $n \in \{0, 1, \ldots, N\}$ that $t_n = \frac{nT}{N}$, $\rho = d+1+(N-1)\big(2d(d+10) + (d+10)^2 + 4(d+10)$

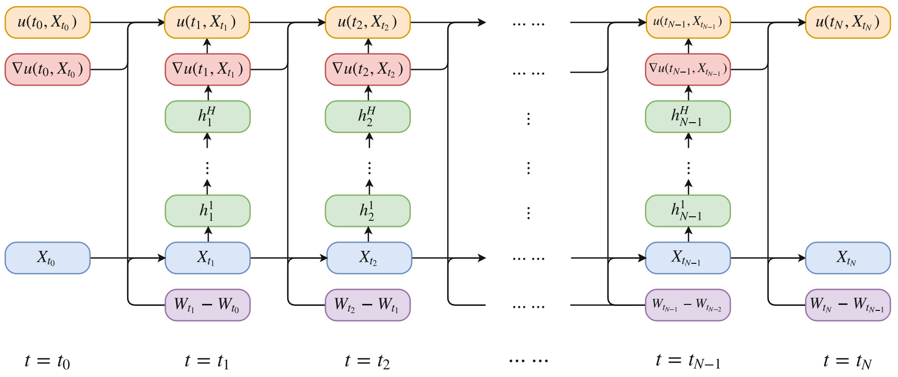

Fig. 1 Rough sketch of the architecture of the deep BSDE method

$+4d)$, $k = 1$, $\varrho = 2\rho$, $\mathcal{U}^\theta = \theta_1$, $\Xi_0 = 0$, let $\mu\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ be functions, let $u\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}$ be a continuous and at most polynomially growing function which satisfies for all $(t,x) \in [0,T) \times \mathbb{R}^d$ that $u|_{[0,T)\times\mathbb{R}^d} \in C^{1,2}([0,T) \times \mathbb{R}^d, \mathbb{R})$, $u(T,x) = g(x)$, and

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\operatorname{Trace}\big(\sigma(t,x)\,[\sigma(t,x)]^*\,(\operatorname{Hess}_x u)(t,x)\big) + \langle \mu(t,x), (\nabla_x u)(t,x)\rangle \\ &\quad + f\big(t,x,u(t,x),[(\nabla_x u)(t,x)]^*\,\sigma(t,x)\big) = 0,\end{aligned} \tag{4.1}$$

assume for all $s,t \in [0,T]$, $x,w \in \mathbb{R}^d$ that $\Upsilon(s,t,x,w) = x + \mu(s,x)\,(t-s) + \sigma(s,x)w$ (cf. Remark 3.3 above and Subsect. 5.3), let $\varepsilon = 10^{-8}$, $\mathbb{X} = \frac{9}{10}$, $\mathbb{Y} = \frac{999}{1000}$, $J = 64$, $(\gamma_m)_{m\in\mathbb{N}} \subseteq (0,\infty)$, let $\operatorname{Pow}_r\colon \mathbb{R}^\rho \to \mathbb{R}^\rho$, $r \in (0,\infty)$, be the functions which satisfy for all $r \in (0,\infty)$, $x = (x_1,\dots,x_\rho) \in \mathbb{R}^\rho$ that

$$\operatorname{Pow}_r(x) = \big(|x_1|^r, \dots, |x_\rho|^r\big), \tag{4.2}$$

and assume for all $m \in \mathbb{N}$, $x = (x_1,\dots,x_\rho)$, $y = (y_1,\dots,y_\rho) \in \mathbb{R}^\rho$, $(\varphi_j)_{j\in\mathbb{N}} \in (\mathbb{R}^\rho)^{\mathbb{N}}$ that

$$\begin{aligned}\Psi_m(x,y,(\varphi_j)_{j\in\mathbb{N}}) = \big(&\mathbb{X}x + (1-\mathbb{X})\big(\tfrac{1}{J}\textstyle\sum_{j=1}^J \varphi_j\big), \mathbb{Y}y \\ &+ (1-\mathbb{Y})\operatorname{Pow}_2\big(\tfrac{1}{J}\textstyle\sum_{j=1}^J \varphi_j\big)\big)\end{aligned} \tag{4.3}$$

and

$$\psi_m(x,y) = \left(\left[\varepsilon + \frac{\sqrt{|y_1|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m x_1}{(1-\mathbb{X}^m)}, \dots, \left[\varepsilon + \frac{\sqrt{|y_\rho|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m x_\rho}{(1-\mathbb{X}^m)}\right) \tag{4.4}$$

(cf. Example 5.2 and Kingma and Ba [25]).

Remark 4.1 In this remark, we illustrate the specific choice of the dimension $\rho \in \mathbb{N}$ of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ in the framework in Subsect. 4.1 above.

(i) The first component of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ is employed for approximating the real number $u(0,\xi) \in \mathbb{R}$.

(ii) The next d-components of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ are employed for approximating the components of the $(1 \times d)$-matrix $(\frac{\partial}{\partial x}u)(0,\xi)\,\sigma(0,\xi) \in \mathbb{R}^{1\times d}$.

(iii) In each of the employed $N-1$ neural networks, we use $d(d+10)$ components of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ to describe the linear transformation from the d-dimensional first layer (input layer) to the $(d+10)$-dimensional second layer (first hidden layer) (to uniquely describe a real $(d+10) \times d$-matrix).

(iv) In each of the employed $N-1$ neural networks, we use $(d+10)^2$ components of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ to uniquely describe the linear transformation from the $(d+10)$-dimensional second layer (first hidden layer) to the $(d+10)$-dimensional third layer (second hidden layer) (to uniquely describe a real $(d+10) \times (d+10)$-matrix).

(v) In each of the employed $N-1$ neural networks, we use $d(d+10)$ components of $\theta = (\theta_1,\dots,\theta_\rho) \in \mathbb{R}^\rho$ to describe the linear transformation from the $(d+10)$-dimensional third layer (second hidden layer) to the d-dimensional fourth layer (output layer) (to uniquely describe a real $d \times (d+10)$-matrix).

(vi) Before each of the linear transformations in item (iii) above as well as after each of the linear transformations in items (iii)–(v) above we employ a componentwise affine linear transformation (multiplication with a diagonal matrix and addition of a vector) within the batch normalization procedure, i.e., in each of the employed $N-1$ neural networks, we use $2d$ components of $\theta = (\theta_1, \dots, \theta_\rho) \in \mathbb{R}^\rho$ for the componentwise affine linear transformation before the first linear transformation (see item (iii)), we use $2(d+10)$ components of $\theta = (\theta_1, \dots, \theta_\rho) \in \mathbb{R}^\rho$ for the componentwise affine linear transformation between the first linear transformation (see item (iii)) and the first application of the activation function, we use $2(d+10)$ components of $\theta = (\theta_1, \dots, \theta_\rho) \in \mathbb{R}^\rho$ for the componentwise affine linear transformation between the second linear transformation (see item (iv)) and the second application of the activation function, and we use $2d$ components of $\theta = (\theta_1, \dots, \theta_\rho) \in \mathbb{R}^\rho$ for the componentwise affine linear transformation after the third linear transformation (see item (v)).

Summing (i)–(vi) results in

$$\begin{aligned}\rho = {} & \underbrace{1+d}_{\text{items (i)–(ii)}} + \underbrace{(N-1)\Big(d(d+10) + (d+10)^2 + d(d+10)\Big)}_{\text{items (iii)–(v)}} \\ & + \underbrace{(N-1)\,(2d + 2(d+10) + 2(d+10) + 2d)}_{\text{item ((vi))}} \qquad (4.5) \\ = {} & d + 1 + (N-1)\Big(2d(d+10) + (d+10)^2 + 4(d+10) + 4d\Big).\end{aligned}$$

4.2 Allen–Cahn Equation

In this section, we test the deep BSDE method in the case of a 100-dimensional Allen–Cahn PDE with a cubic nonlinearity (see (4.6)).

More specifically, assume the setting in Subsect. 4.1 and assume for all $t \in [0, T]$, $x, w \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $m \in \mathbb{N}$ that $\gamma_m = 5 \cdot 10^{-4}$, $d = 100$, $T = \frac{3}{10}$, $N = 20$, $\mu(t,x) = 0$, $\sigma(t,x)w = \sqrt{2}\,w$, $\xi = (0, 0, \dots, 0) \in \mathbb{R}^d$, $f(t,x,y,z) = y - y^3$, and $g(x) = \big[2 + \frac{2}{5}\,\|x\|^2_{\mathbb{R}^d}\big]^{-1}$. Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0, T)$, $x \in \mathbb{R}^d$ that $u(T,x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t,x) + u(t,x) - [u(t,x)]^3 + (\Delta_x u)(t,x) = 0. \qquad (4.6)$$

In Table 1, we approximately calculate the mean of $\mathcal{U}^{\Theta_m}$, the standard deviation of $\mathcal{U}^{\Theta_m}$, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, the standard deviation of the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, and the runtime in seconds needed to calculate one realization of $\mathcal{U}^{\Theta_m}$ against $m \in \{0, 1000, 2000, 3000, 4000\}$ based on five independent realizations (five independent runs) (cf. also the PYTHON code 1 in Subsection 6.1 in [11]). Table 1 also depicts the mean of the loss function associated with Θ_m and the standard deviation of the loss function associated with

Table 1 Numerical simulations for the deep BSDE method in Subsect. 3.2 in the case of the PDE (4.6)

Number of iteration steps m	Mean of $\mathcal{U}^{\Theta_m}$	Standard deviation of $\mathcal{U}^{\Theta_m}$	Relative L^1-appr. error	Standard deviation of the relative L^1-appr. error	Mean of the loss function	Standard deviation of the loss function	Runtime in sec. for one realization of $\mathcal{U}^{\Theta_m}$
0	0.4740	0.0514	7.9775	0.9734	0.11630	0.02953	
1000	0.1446	0.0340	1.7384	0.6436	0.00550	0.00344	201
2000	0.0598	0.0058	0.1318	0.1103	0.00029	0.00006	348
3000	0.0530	0.0002	0.0050	0.0041	0.00023	0.00001	500
4000	0.0528	0.0002	0.0030	0.0022	0.00020	0.00001	647

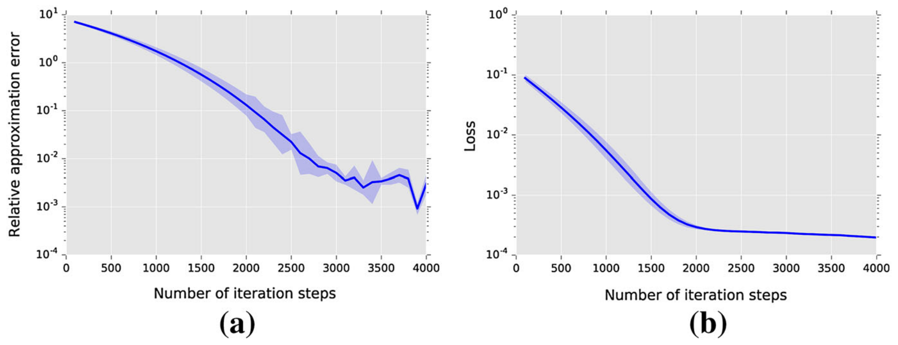

Fig. 2 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \ldots, 4000\}$ in the case of the PDE (4.6). The deep BSDE approximation $\mathcal{U}^{\Theta_{4000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0030 in a runtime of 595 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

Θ_m against $m \in \{0, 1000, 2000, 3000, 4000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In addition, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \ldots, 4000\}$ is pictured on the left hand side of Fig. 2 based on 5 independent realizations (5 independent runs) and the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \ldots, 4000\}$ is pictured on the right hand side of Fig. 2 based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In the approximative computations of the relative L^1-approximation errors in Table 1 and Fig. 2, the value $u(0, \xi) = u(0, 0, \ldots, 0)$ of the exact solution u of the PDE (4.6) is replaced by the value 0.052802 which, in turn, is calculated by means of the Branching diffusion method (cf. the MATLAB code 2 in Subsection 6.2 in [11] and cf., e.g., [20–22] for analytical and numerical results for the Branching diffusion method in the literature).

4.3 A Hamilton–Jacobi–Bellman (HJB) Equation

In this subsection, we apply the deep BSDE method in Subsect. 3.2 to a Hamilton–Jacobi–Bellman (HJB) equation which admits an explicit solution that can be obtained through the Cole-Hopf transformation (cf., e.g., Chassagneux and Richou [7, Subsect. 4.2] and Debnath [10, Subsect. 8.4]).

Assume the setting in Subsect. 4.1 and assume for all $t \in [0, T]$, $x, w \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $m \in \mathbb{N}$ that $d = 100$, $T = 1$, $N = 20$, $\gamma_m = \frac{1}{100}$, $\mu(t, x) = 0$, $\sigma(t, x)w = \sqrt{2}\, w$, $\xi = (0, 0, \ldots, 0) \in \mathbb{R}^d$, $f(t, x, y, z) = -\|z\|^2_{\mathbb{R}^{1\times d}}$, and $g(x) = \ln(\frac{1}{2}\,[1 + \|x\|^2_{\mathbb{R}^d}])$. Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0, T)$, $x \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t, x) + (\Delta_x u)(t, x) = \|(\nabla_x u)(t, x)\|^2_{\mathbb{R}^d}. \tag{4.7}$$

In Table 2, we approximately calculate the mean of $\mathcal{U}^{\Theta_m}$, the standard deviation of $\mathcal{U}^{\Theta_m}$, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, the standard deviation

Table 2 Numerical simulations for the deep BSDE method in Subsect. 3.2 in the case of the PDE (4.7)

Number of iteration steps m	Mean of $\mathcal{U}^{\Theta_m}$	Standard deviation of $\mathcal{U}^{\Theta_m}$	Relative L^1-appr. error	Standard deviation of the relative L^1-appr. error	Mean of the loss function	Standard deviation of the loss function	Runtime in sec. for one realization of $\mathcal{U}^{\Theta_m}$
0	0.3167	0.3059	0.9310	0.0666	18.4052	2.5090	
500	2.2785	0.3521	0.5036	0.0767	2.1789	0.3848	116
1000	3.9229	0.3183	0.1454	0.0693	0.5226	0.2859	182
1500	4.5921	0.0063	0.0013	0.006	0.0239	0.0024	248
2000	4.5977	0.0019	0.0017	0.0004	0.0231	0.0026	330

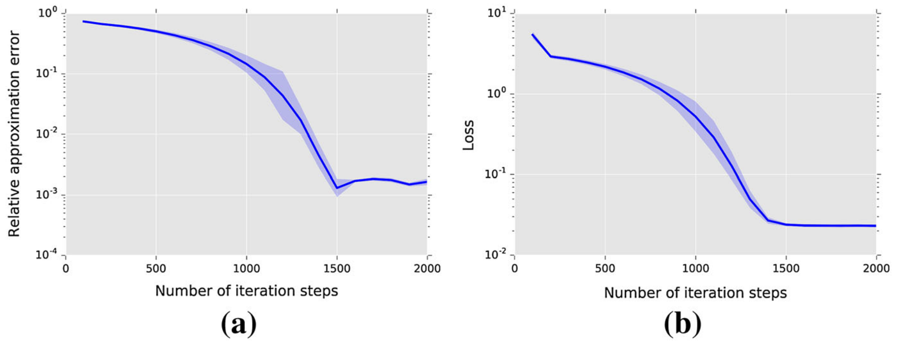

Fig. 3 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \dots, 2000\}$. The deep BSDE approximation $\mathcal{U}^{\Theta_{2000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0017 in a runtime of 283 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

of the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, and the runtime in seconds needed to calculate one realization of $\mathcal{U}^{\Theta_m}$ against $m \in \{0, 500, 1000, 1500, 2000\}$ based on 5 independent realizations (5 independent runs). Table 2 also depicts the mean of the loss function associated with Θ_m and the standard deviation of the loss function associated with Θ_m against $m \in \{0, 500, 1000, 1500, 2000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In addition, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \dots, 2000\}$ is pictured on the left hand side of Fig. 3 based on 5 independent realizations (5 independent runs) and the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \dots, 2000\}$ is pictured on the right hand side of Fig. 3 based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In the approximative computations of the relative L^1-approximation errors in Table 2 and Fig. 3, the value $u(0, \xi) = u(0, 0, \dots, 0)$ of the exact solution u of the PDE (4.6) is replaced by the value 4.5901 which, in turn, is calculated by means of Lemma 4.2 (with $d = 100$, $T = 1$, $\alpha = 1$, $\beta = -1$, $g = \mathbb{R}^d \ni x \mapsto \ln(\frac{1}{2}[1 + \|x\|^2_{\mathbb{R}^d}]) \in \mathbb{R}$ in the notation of Lemma 4.2) and a classical Monte Carlo method (cf. the MATLAB code 3 in Subsection 6.3 in [11]).

Lemma 4.2 (Cf., e.g., Subsect. 4.2 in [7] and Subsect. 8.4 in [10]) *Let $d \in \mathbb{N}$, $T, \alpha \in (0, \infty)$, $\beta \in \mathbb{R}\backslash\{0\}$, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, let $W\colon [0, T] \times \Omega \to \mathbb{R}^d$ be a d-dimensional standard Brownian motion, let $g \in C^2(\mathbb{R}^d, \mathbb{R})$ be a function which satisfies $\sup_{x\in\mathbb{R}^d} [\beta g(x)] < \infty$, let $f\colon [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ be the function which satisfies for all $t \in [0, T]$, $x = (x_1, \dots, x_d)$, $z = (z_1, \dots, z_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$ that*

$$f(t, x, y, z) = \beta\|z\|^2_{\mathbb{R}^d} = \beta \textstyle\sum_{i=1}^d |z_i|^2, \tag{4.8}$$

and let $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ be the function which satisfies for all $(t, x) \in [0, T] \times \mathbb{R}^d$ that

$$u(t, x) = \frac{\alpha}{\beta} \ln\left(\mathbb{E}\left[\exp\left(\frac{\beta g(x + W_{T-t}\sqrt{2\alpha})}{\alpha}\right)\right]\right). \tag{4.9}$$

Then

(i) *it holds that* $u\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}$ *is a continuous function*;
(ii) *it holds that* $u|_{[0,T)\times\mathbb{R}^d}\in C^{1,2}([0,T)\times\mathbb{R}^d,\mathbb{R})$; *and*
(iii) *it holds for all* $(t,x)\in[0,T)\times\mathbb{R}^d$ *that* $u(T,x)=g(x)$ *and*

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x)+\alpha(\Delta_x u)(t,x)+f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big)\\&=\frac{\partial u}{\partial t}(t,x)+\alpha(\Delta_x u)(t,x)+\beta\|(\nabla_x u)(t,x)\|_{\mathbb{R}^d}^2\\&=\frac{\partial u}{\partial t}(t,x)+\alpha(\Delta_x u)(t,x)+\beta\sum_{j=1}^{d}\left|\frac{\partial u}{\partial x_j}(t,x)\right|^2=0.\end{aligned}\tag{4.10}$$

Proof of Lemma 4.2 Throughout this proof, let $c=\frac{\alpha}{\beta}\in\mathbb{R}\backslash\{0\}$ and let $\mathcal{V}\colon\mathbb{R}^d\to(0,\infty)$ and $v\colon[0,T]\times\mathbb{R}^d\to(0,\infty)$ be the functions which satisfy for all $t\in[0,T]$, $x\in\mathbb{R}^d$ that

$$\mathcal{V}(x)=\exp\left(\frac{g(x)}{c}\right)=\exp\left(\frac{\beta g(x)}{\alpha}\right)\quad\text{and}\quad v(t,x)=\mathbb{E}\big[\mathcal{V}\big(x+W_{T-t}\sqrt{2\alpha}\big)\big].\tag{4.11}$$

Observe that the hypothesis that $\sup_{x\in\mathbb{R}^d}[\beta g(x)]<\infty$ ensures that for all $\omega\in\Omega$ it holds that

$$\sup_{t\in[0,T]}\sup_{x\in\mathbb{R}^d}\left|\mathcal{V}\big(x+W_{T-t}(\omega)\sqrt{2\alpha}\big)\right|\le\sup_{x\in\mathbb{R}^d}|\mathcal{V}(x)|=\sup_{x\in\mathbb{R}^d}\mathcal{V}(x)=\exp\left(\frac{\sup_{x\in\mathbb{R}^d}[\beta g(x)]}{\alpha}\right)<\infty.\tag{4.12}$$

Combining this with Lebesgue's theorem of dominated convergence ensures that $v\colon[0,T]\times\mathbb{R}^d\to(0,\infty)$ is a continuous function. This and the fact that

$$\forall\,(t,x)\in[0,T]\times\mathbb{R}^d\colon u(t,x)=c\ln(v(t,x))\tag{4.13}$$

establish Item (i). Next note that the Feynman–Kac formula ensures that for all $t\in[0,T)$, $x\in\mathbb{R}^d$ it holds that $v|_{[0,T)\times\mathbb{R}^d}\in C^{1,2}([0,T)\times\mathbb{R}^d,(0,\infty))$ and

$$\frac{\partial v}{\partial t}(t,x)+\alpha(\Delta_x v)(t,x)=0.\tag{4.14}$$

This and (4.13) demonstrate Item (ii). It thus remains to prove Item (iii). For this, note that the chain rule and (4.13) imply that for all $t\in[0,T)$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$, $i\in\{1,2,\dots,d\}$ it holds that

$$\frac{\partial u}{\partial t}(t,x)=\frac{c}{v(t,x)}\cdot\frac{\partial v}{\partial t}(t,x)\quad\text{and}\quad\frac{\partial u}{\partial x_i}(t,x)=\frac{c}{v(t,x)}\cdot\frac{\partial v}{\partial x_i}(t,x).\tag{4.15}$$

Again the chain rule and (4.13) hence ensure that for all $t \in [0, T)$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $i \in \{1, 2, \ldots, d\}$ it holds that

$$\frac{\partial u^2}{\partial x_i^2}(t, x) = \frac{c}{v(t, x)} \cdot \frac{\partial v^2}{\partial x_i^2}(t, x) - \frac{c}{[v(t, x)]^2} \cdot \left[\frac{\partial v}{\partial x_i}(t, x)\right]^2. \tag{4.16}$$

This assures that for all $t \in [0, T)$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $i \in \{1, 2, \ldots, d\}$ it holds that

$$\begin{aligned} \alpha(\Delta_x u)(t, x) &= \frac{\alpha c}{v(t, x)} \cdot (\Delta_x v)(t, x) - \frac{\alpha c}{[v(t, x)]^2} \cdot \sum_{i=1}^{d} \left[\frac{\partial v}{\partial x_i}(t, x)\right]^2 \\ &= \frac{\alpha c (\Delta_x v)(t, x)}{v(t, x)} - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2}. \end{aligned} \tag{4.17}$$

Combining this with (4.15) demonstrates that for all $t \in [0, T)$, $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned} &\frac{\partial u}{\partial t}(t, x) + \alpha(\Delta_x u)(t, x) + \beta \, \|(\nabla_x u)(t, x)\|_{\mathbb{R}^d}^2 \\ &= \frac{c}{v(t, x)} \cdot \frac{\partial v}{\partial t}(t, x) + \frac{\alpha c (\Delta_x v)(t, x)}{v(t, x)} - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2} \\ &+ \beta \, \|(\nabla_x u)(t, x)\|_{\mathbb{R}^d}^2 . \end{aligned} \tag{4.18}$$

Equation (4.14) hence shows that for all $t \in [0, T)$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned} &\frac{\partial u}{\partial t}(t, x) + \alpha(\Delta_x u)(t, x) + \beta \, \|(\nabla_x u)(t, x)\|_{\mathbb{R}^d}^2 \\ &= \beta \, \|(\nabla_x u)(t, x)\|_{\mathbb{R}^d}^2 - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2} \\ &= \beta \left[\sum_{i=1}^{d} \left|\frac{\partial u}{\partial x_i}(t, x)\right|^2\right] - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2}. \end{aligned} \tag{4.19}$$

This and (4.15) demonstrate that for all $t \in [0, T)$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned} &\frac{\partial u}{\partial t}(t, x) + \alpha(\Delta_x u)(t, x) + \beta \, \|(\nabla_x u)(t, x)\|_{\mathbb{R}^d}^2 \\ &= \beta \left[\sum_{i=1}^{d} \left|\frac{c}{v(t, x)} \cdot \frac{\partial v}{\partial x_i}(t, x)\right|^2\right] - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2} \\ &= \frac{c^2 \beta}{[v(t, x)]^2} \left[\sum_{i=1}^{d} \left|\frac{\partial v}{\partial x_i}(t, x)\right|^2\right] - \frac{\alpha c \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2} \\ &= \frac{[c^2 \beta - c\alpha] \, \|(\nabla_x v)(t, x)\|_{\mathbb{R}^d}^2}{[v(t, x)]^2} = 0. \end{aligned} \tag{4.20}$$

This and the fact that

$$\forall x \in \mathbb{R}^d\colon u(T,x) = c\ln(v(T,x)) = c\ln(\mathcal{V}(x)) \\ = c\ln\left(\exp\left(\frac{g(x)}{c}\right)\right) = g(x) \tag{4.21}$$

establish Item (iii). The proof of Lemma 4.2 is thus completed. □

4.4 Pricing of European Financial Derivatives with Different Interest Rates for Borrowing and Lending

In this subsection, we apply the deep BSDE method to a pricing problem of an European financial derivative in a financial market where the risk free bank account used for the hedging of the financial derivative has different interest rates for borrowing and lending (see Bergman [4] and, e.g., [2,3,5,8,13,14] where this example has been used as a test example for numerical methods for BSDEs).

Assume the setting in Subsect. 4.1, let $\bar{\mu} = \frac{6}{100}$, $\bar{\sigma} = \frac{2}{10}$, $R^l = \frac{4}{100}$, $R^b = \frac{6}{100}$, and assume for all $t \in [0,T]$, $x = (x_1,\ldots,x_d)$, $w = (w_1,\ldots,w_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $m \in \mathbb{N}$ that $d = 100$, $T = \frac{1}{2}$, $N = 20$, $\gamma_m = 5\cdot 10^{-3} = 0.005$, $\mu(t,x) = \bar{\mu}x$, $\sigma(t,x) = \bar{\sigma}\operatorname{diag}_{\mathbb{R}^{d\times d}}(x_1,\ldots,x_d)$, $\xi = (100,100,\ldots,100) \in \mathbb{R}^d$, and

$$g(x) = \max\left\{\left[\max_{1\le i\le 100} x_i\right] - 120, 0\right\} \\ -2\max\left\{\left[\max_{1\le i\le 100} x_i\right] - 150, 0\right\}, \tag{4.22}$$

$$f(t,x,y,z) = -R^l y - \frac{(\bar{\mu}-R^l)}{\bar{\sigma}}\sum_{i=1}^d z_i \\ +(R^b - R^l)\max\left\{0, \left[\frac{1}{\bar{\sigma}}\sum_{i=1}^d z_i\right] - y\right\}. \tag{4.23}$$

Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0,T)$, $x = (x_1,x_2,\ldots,x_d) \in \mathbb{R}^d$ that $u(T,x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t,x) + f\big(t,x,u(t,x),\bar{\sigma}\operatorname{diag}_{\mathbb{R}^{d\times d}}(x_1,\ldots,x_d)(\nabla_x u)(t,x)\big) \\ +\bar{\mu}\sum_{i=1}^d x_i\frac{\partial u}{\partial x_i}(t,x) + \frac{\bar{\sigma}^2}{2}\sum_{i=1}^d |x_i|^2\frac{\partial^2 u}{\partial x_i^2}(t,x) = 0. \tag{4.24}$$

Hence, we obtain for all $t \in [0, T)$, $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\begin{aligned} &\frac{\partial u}{\partial t}(t,x) + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \frac{\partial^2 u}{\partial x_i^2}(t,x) \\ &+ \max\left\{ R^b \left(\left[\textstyle\sum_{i=1}^{d} x_i \left(\frac{\partial u}{\partial x_i}\right)(t,x) \right] - u(t,x) \right), \right. \\ &\left. R^l \left(\left[\textstyle\sum_{i=1}^{d} x_i \left(\frac{\partial u}{\partial x_i}\right)(t,x) \right] - u(t,x) \right) \right\} = 0. \end{aligned} \tag{4.25}$$

This shows that for all $t \in [0, T)$, $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ it holds that $u(T, x) = g(x)$ and

$$\begin{aligned} &\frac{\partial u}{\partial t}(t,x) + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \frac{\partial^2 u}{\partial x_i^2}(t,x) \\ &- \min\left\{ R^b \left(u(t,x) - \sum_{i=1}^{d} x_i \frac{\partial u}{\partial x_i}(t,x) \right), \right. \\ &\left. R^l \left(u(t,x) - \sum_{i=1}^{d} x_i \frac{\partial u}{\partial x_i}(t,x) \right) \right\} = 0. \end{aligned} \tag{4.26}$$

In Table 3, we approximately calculate the mean of $\mathcal{U}^{\Theta_m}$, the standard deviation of $\mathcal{U}^{\Theta_m}$, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, the standard deviation of the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$, and the runtime in seconds needed to calculate one realization of $\mathcal{U}^{\Theta_m}$ against $m \in \{0, 1000, 2000, 3000, 4000\}$ based on 5 independent realizations (5 independent runs). Table 3 also depicts the mean of the loss function associated with Θ_m and the standard deviation of the loss function associated with Θ_m against $m \in \{0, 1000, 2000, 3000, 4000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In addition, the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \dots, 4000\}$ is pictured on the left hand side of Fig. 4 based on 5 independent realizations (5 independent runs), and the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \dots, 4000\}$ is pictured on the right hand side of Fig. 4 based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs). In the approximative computations of the relative L^1-approximation errors in Table 3 and Fig. 4, the value $u(0, \xi) = u(0, 100, \dots, 100)$ of the exact solution u of the PDE (4.26) is replaced by the value 21.299 which, in turn, is calculated by means of the multilevel Picard approximation method in E et al. [13] (see [13, $\rho = 7$ in Table 6 in Subsect. 3.3]).

4.5 Multidimensional Burgers-Type PDEs with Explicit Solutions

In this subsection, we consider a high-dimensional version of the example analyzed numerically in Chassagneux [6, Example 4.6 in Subsection 4.2].

Table 3 Numerical simulations for the deep BSDE method in Subsect. 3.2 in the case of the PDE (4.26)

Number of iteration steps m	Mean of $\mathcal{U}^{\Theta_m}$	Standard deviation of $\mathcal{U}^{\Theta_m}$	Relative L^1-appr. error	Standard deviation of the relative L^1-appr. error	Mean of the loss function	Standard deviation of the loss function	Runtime in sec. for one realization of $\mathcal{U}^{\Theta_m}$
0	16.964	0.882	0.204	0.041	53.666	8.957	
1000	20.309	0.524	0.046	0.025	30.886	3.076	194
2000	21.150	0.098	0.007	0.005	29.197	3.160	331
3000	21.229	0.034	0.003	0.002	29.070	3.246	470
4000	21.217	0.043	0.004	0.002	29.029	3.236	617

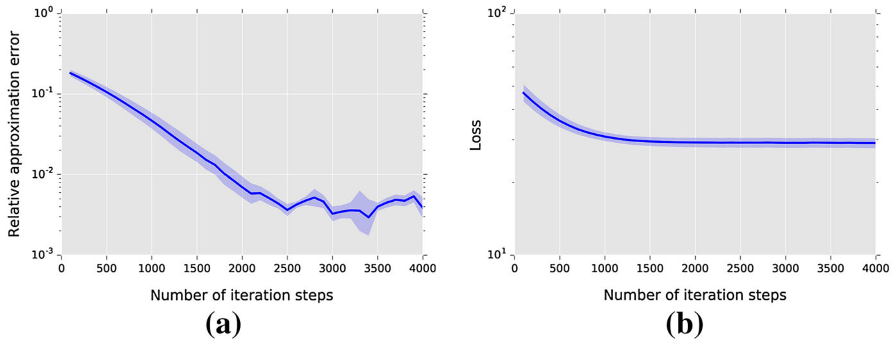

Fig. 4 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \dots, 4000\}$ in the case of the PDE (4.26). The deep BSDE approximation $\mathcal{U}^{\Theta_{4000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0039 in a runtime of 566 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

More specifically, assume the setting in Subsect. 4.1, and assume for all $t \in [0, T]$, $x = (x_1, \dots, x_d)$, $w = (w_1, \dots, w_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z = (z_i)_{i \in \{1,2,\dots,d\}} \in \mathbb{R}^{1 \times d}$ that $\mu(t, x) = 0$, $\sigma(t, x)w = \frac{d}{\sqrt{2}} w$, $\xi = (0, 0, \dots, 0) \in \mathbb{R}^d$, and

$$g(x) = \frac{\exp(T + \frac{1}{d}\sum_{i=1}^d x_i)}{\left(1 + \exp(T + \frac{1}{d}\sum_{i=1}^d x_i)\right)}, \qquad f(t, x, y, z) = \left(y - \frac{2+d}{2d}\right)\left(\sum_{i=1}^d z_i\right). \tag{4.27}$$

Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0, T)$, $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t, x) + \frac{d^2}{2}(\Delta_x u)(t, x) + \left(u(t, x) - \frac{2+d}{2d}\right)\left(d \sum_{i=1}^d \frac{\partial u}{\partial x_i}(t, x)\right) = 0 \tag{4.28}$$

(cf. Lemma 4.3 [with $\alpha = d^2$, $\kappa = \frac{1}{d}$ in the notation of Lemma 4.3]). On the left hand side of Fig. 5 we present approximately the relative L^1-approximation error associated to $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \dots, 60{,}000\}$ based on 5 independent realizations (5 independent runs) in the case

$$T = 1, \qquad d = 20, \qquad N = 80, \qquad \forall m \in \mathbb{N}: \\ \gamma_m = 10^{(\mathbb{1}_{[1,30000]}(m) + \mathbb{1}_{[1,50000]}(m) - 4)}. \tag{4.29}$$

On the right hand side of Fig. 5, we present approximately the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \dots, 60{,}000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs) in the case (4.29). On the left hand side of Fig. 6, we present approximately the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \dots, 30{,}000\}$ based on 5 independent realizations (5 independent runs) in the case

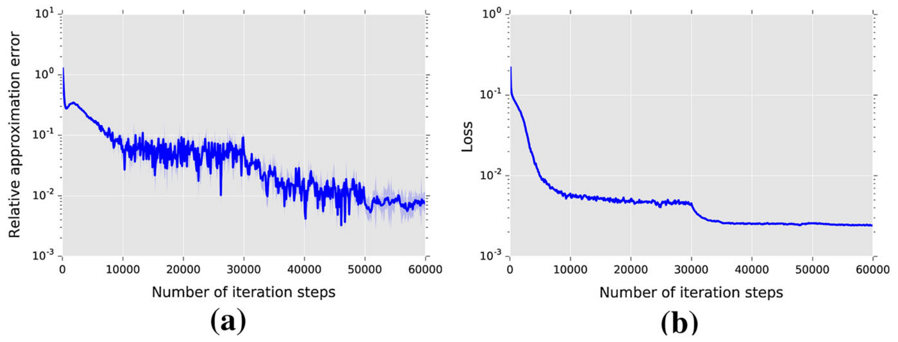

Fig. 5 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \dots, 60{,}000\}$ in the case of the PDE (4.28) with (4.29). The deep BSDE approximation $\mathcal{U}^{\Theta_{60{,}000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0073 in a runtime of 20 389 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

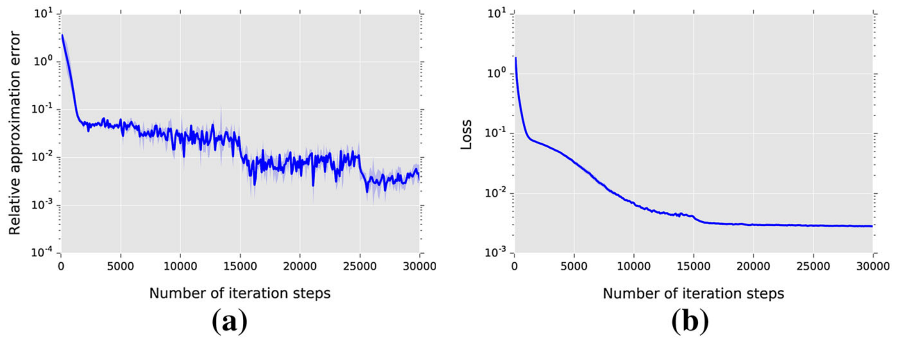

Fig. 6 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \dots, 30{,}000\}$ in the case of the PDE (4.28) with (4.30). The deep BSDE approximation $\mathcal{U}^{\Theta_{30{,}000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0035 in a runtime of 4281 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

$$T = \frac{2}{10}, \qquad d = 50, \qquad N = 30,$$
$$\forall\, m \in \mathbb{N} \colon \gamma_m = 10^{(\mathbb{1}_{[1,15000]}(m) + \mathbb{1}_{[1,25000]}(m) - 4)}. \tag{4.30}$$

On the right hand side of Fig. 6, we present approximately the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \dots, 30{,}000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs) in the case (4.30).

Lemma 4.3 (Cf. Example 4.6 in Subsection 4.2 in [6]) *Let* $\alpha, \kappa, T \in (0, \infty)$, $d \in \mathbb{N}$, *let* $u \colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ *be the function which satisfies for all* $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ *that*

$$u(t, x) = 1 - \frac{1}{\left(1 + \exp\left(t + \kappa \sum_{i=1}^{d} x_i\right)\right)} = \frac{\exp\left(t + \kappa \sum_{i=1}^{d} x_i\right)}{\left(1 + \exp\left(t + \kappa \sum_{i=1}^{d} x_i\right)\right)}, \tag{4.31}$$

and let $f\colon [0,T]\times\mathbb{R}^d\times\mathbb{R}^{1+d}\to\mathbb{R}$ *be the function which satisfies for all* $t\in[0,T]$, $x\in\mathbb{R}^d$, $y\in\mathbb{R}$, $z=(z_1,\dots,z_d)\in\mathbb{R}^d$ *that*

$$f(t,x,y,z)=\left(\alpha\kappa y-\frac{1}{d\kappa}-\frac{\alpha\kappa}{2}\right)\left(\sum_{i=1}^{d} z_i\right). \tag{4.32}$$

Then it holds for all $t\in[0,T]$, $x\in\mathbb{R}^d$ *that*

$$\frac{\partial u}{\partial t}(t,x)+\frac{\alpha}{2}(\Delta_x u)(t,x)+f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big)=0. \tag{4.33}$$

Proof of Lemma 4.3 Throughout this proof, let $\beta,\gamma\in(0,\infty)$ be the real numbers given by

$$\beta=\alpha\kappa \quad\text{and}\quad \gamma=\frac{1}{d\kappa}+\frac{\alpha\kappa}{2}, \tag{4.34}$$

and let $w\colon[0,T]\times\mathbb{R}^d\to(0,\infty)$ be the function which satisfies for all $t\in[0,T]$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ that

$$w(t,x)=\exp\left(t+\kappa\sum_{i=1}^{d}x_i\right). \tag{4.35}$$

Observe that for all $t\in[0,T]$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$, $i\in\{1,2,\dots,d\}$ it holds that

$$\begin{aligned} u(t,x)&=1-[1+w(t,x)]^{-1}=\frac{[1+w(t,x)]}{[1+w(t,x)]}-\frac{1}{[1+w(t,x)]}\\ &=\frac{w(t,x)}{1+w(t,x)}, \end{aligned} \tag{4.36}$$

$$\frac{\partial u}{\partial t}(t,x)=[1+w(t,x)]^{-2}\cdot\frac{\partial w}{\partial t}(t,x)=\frac{w(t,x)}{[1+w(t,x)]^2} \tag{4.37}$$

and

$$\frac{\partial u}{\partial x_i}(t,x)=[1+w(t,x)]^{-2}\cdot\frac{\partial w}{\partial x_i}(t,x)=\kappa\, w(t,x)\,[1+w(t,x)]^{-2}. \tag{4.38}$$

Note that (4.36), (4.37), and (4.38) ensure that for all $t\in[0,T]$, $x\in\mathbb{R}^d$ it holds that

$$\begin{aligned} &\frac{\partial u}{\partial t}(t,x)+\frac{\alpha}{2}(\Delta_x u)(t,x)+f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big)\\ &=\frac{\partial u}{\partial t}(t,x)+\frac{\alpha}{2}(\Delta_x u)(t,x)+(\beta u(t,x)-\gamma)\left(\sum_{i=1}^{d}\frac{\partial u}{\partial x_i}(t,x)\right)\\ &=\frac{\partial u}{\partial t}(t,x)+\frac{\alpha}{2}(\Delta_x u)(t,x)+d\,\frac{\partial u}{\partial x_1}(t,x)\big(\beta u(t,x)-\gamma\big)\\ &=\frac{w(t,x)}{[1+w(t,x)]^2}+\frac{\alpha}{2}(\Delta_x u)(t,x)+\frac{d\kappa w(t,x)}{[1+w(t,x)]^2}\left(\frac{\beta w(t,x)}{[1+w(t,x)]}-\gamma\right). \end{aligned} \tag{4.39}$$

Moreover, observe that (4.38) demonstrates that for all $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned}\frac{\partial^2 u}{\partial x_i^2}(t,x) &= \kappa \frac{\partial w}{\partial x_i}(t,x)\,[1+w(t,x)]^{-2} - 2\kappa w(t,x)\,[1+w(t,x)]^{-3}\,\frac{\partial w}{\partial x_i}(t,x) \\ &= \frac{\kappa^2 w(t,x)}{[1+w(t,x)]^2} - \frac{2\kappa^2 |w(t,x)|^2}{[1+w(t,x)]^3} = \frac{\kappa^2 w(t,x)}{[1+w(t,x)]^2} \\ &\left[1 - \frac{2w(t,x)}{[1+w(t,x)]}\right].\end{aligned} \tag{4.40}$$

Hence, we obtain that for all $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that

$$\frac{\alpha}{2}(\Delta_x u)(t,x) = \frac{d\alpha}{2}\frac{\partial^2 u}{\partial x_1^2}(t,x) = \frac{d\alpha\kappa^2 w(t,x)}{2\,[1+w(t,x)]^2}\left[1 - \frac{2w(t,x)}{[1+w(t,x)]}\right]. \tag{4.41}$$

Combining this with (4.39) implies that for all $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x) + \frac{\alpha}{2}(\Delta_x u)(t,x) + f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big) \\ &= \frac{w(t,x)\,[1-d\kappa\gamma]}{[1+w(t,x)]^2} + \frac{\alpha}{2}(\Delta_x u)(t,x) + \frac{d\beta\kappa |w(t,x)|^2}{[1+w(t,x)]^3} \\ &= \frac{w(t,x)\,[1-d\kappa\gamma]}{[1+w(t,x)]^2} + \frac{d\alpha\kappa^2 w(t,x)}{2\,[1+w(t,x)]^2}\left[1 - \frac{2w(t,x)}{[1+w(t,x)]}\right] \\ &+ \frac{d\beta\kappa |w(t,x)|^2}{[1+w(t,x)]^3} \\ &= \frac{w(t,x)\big[1-d\kappa\gamma + \frac{d\alpha\kappa^2}{2}\big]}{[1+w(t,x)]^2} - \frac{d\alpha\kappa^2 |w(t,x)|^2}{[1+w(t,x)]^3} + \frac{d\beta\kappa |w(t,x)|^2}{[1+w(t,x)]^3}.\end{aligned} \tag{4.42}$$

The fact that $\alpha\kappa^2 = \beta\kappa$ hence demonstrates that for all $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x) + \frac{\alpha}{2}(\Delta_x u)(t,x) + f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big) \\ &= \frac{w(t,x)\big[1-d\kappa\gamma + \frac{d\alpha\kappa^2}{2}\big]}{[1+w(t,x)]^2}.\end{aligned} \tag{4.43}$$

This and the fact that $1 + \frac{d\alpha\kappa^2}{2} = d\kappa\gamma$ show that for all $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that

$$\frac{\partial u}{\partial t}(t,x) + \frac{\alpha}{2}(\Delta_x u)(t,x) + f\big(t,x,u(t,x),(\nabla_x u)(t,x)\big) = 0. \tag{4.44}$$

The proof of Lemma 4.3 is thus completed. □

4.6 An Example PDE with Quadratically Growing Derivatives and an Explicit Solution

In this subsection, we consider a modified version of the example analyzed numerically in Gobet and Turkedjiev [15, Section 5].

Assume the setting in Subsect. 4.1, let $\alpha = \frac{4}{10}$, let $\psi \colon [0,T] \times \mathbb{R}^d \to \mathbb{R}$ be the function which satisfies for all $(t,x) \in [0,T] \times \mathbb{R}^d$ that $\psi(t,x) = \sin([T - t + \frac{1}{d}\|x\|^2_{\mathbb{R}^d}]^\alpha)$, and assume for all $t \in [0,T)$, $x, w \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $m \in \mathbb{N}$ that $T = 1$, $d = 100$, $N = 30$, $\gamma_m = 5 \cdot 10^{-3} = \frac{5}{1000} = 0.005$, $\mu(t,x) = 0$, $\sigma(t,x)w = w$, $\xi = (0,0,\ldots,0) \in \mathbb{R}^d$, $g(x) = \sin(\frac{1}{d^\alpha}\|x\|^{2\alpha}_{\mathbb{R}^d})$, and

$$f(t,x,y,z) = \|z\|^2_{\mathbb{R}^{1\times d}} - \|(\nabla_x\psi)(t,x)\|^2_{\mathbb{R}^d} - \frac{\partial\psi}{\partial t}(t,x) - \frac{1}{2}(\Delta_x\psi)(t,x). \tag{4.45}$$

Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0,T)$, $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ that $u(T,x) = g(x)$ and

$$\begin{aligned}&\frac{\partial u}{\partial t}(t,x) + \|(\nabla_x u)(t,x)\|^2_{\mathbb{R}^d} + \frac{1}{2}(\Delta_x u)(t,x)\\ &= \frac{\partial\psi}{\partial t}(t,x) + \|(\nabla_x\psi)(t,x)\|^2_{\mathbb{R}^d} + \frac{1}{2}(\Delta_x\psi)(t,x).\end{aligned} \tag{4.46}$$

On the left hand side of Fig. 7, we present approximately the relative L^1-approximation error associated to $\mathcal{U}^{\Theta_m}$ against $m \in \{1,2,3,\ldots,4000\}$ based on 5 independent realizations (5 independent runs). On the right hand side of Fig. 7, we present approximately the mean of the loss function associated with Θ_m against $m \in \{1,2,3,\ldots,4000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs).

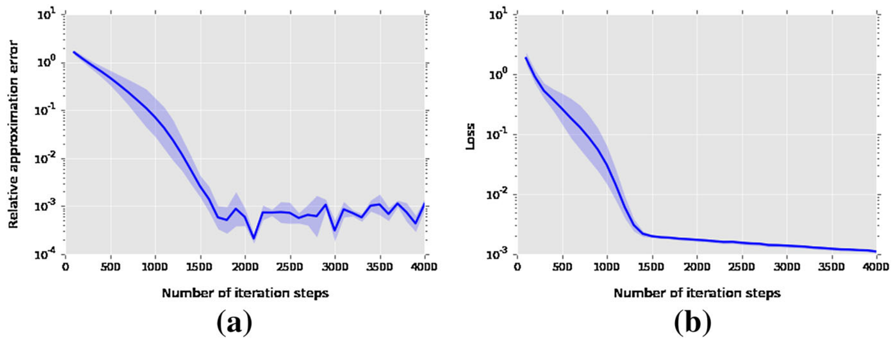

Fig. 7 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1,2,3,\ldots,4000\}$ in the case of the PDE (4.46). The deep BSDE approximation $\mathcal{U}^{\Theta_{4000}} \approx u(0,\xi)$ achieves a relative L^1-approximation error of size 0.0009 in a runtime of 957 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

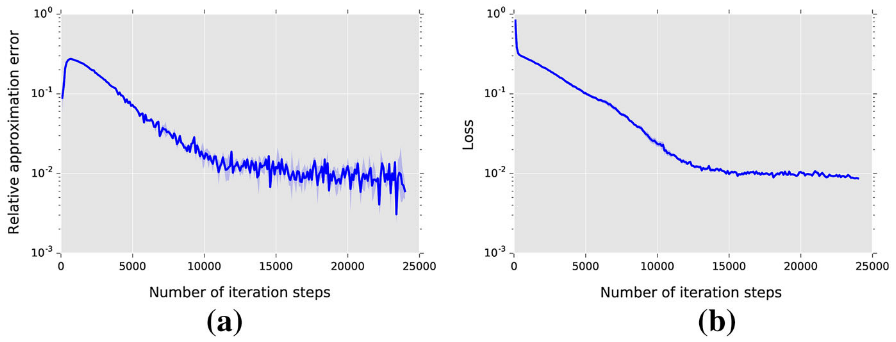

Fig. 8 Relative L^1-approximation error of $\mathcal{U}^{\Theta_m}$ and mean of the loss function against $m \in \{1, 2, 3, \ldots, 24{,}000\}$ in the case of the PDE (4.48). The deep BSDE approximation $\mathcal{U}^{\Theta_{24,000}} \approx u(0, \xi)$ achieves a relative L^1-approximation error of size 0.0060 in a runtime of 4872 sec. **a** Relative L^1-approximation error. **b** Mean of the loss function

4.7 A Time-Dependent Reaction–Diffusion-Type Example PDE with an Oscillating Explicit Solution

In this subsection, we consider a high-dimensional version of the example PDE analyzed numerically in Gobet and Turkedjiev [16, Subsection 6.1]. More specifically, Gobet and Turkedjiev [16, Subsection 6.1] employ the PDE in (4.48) below as a numerical test example but in two space-dimensions ($d = 2$) instead of in hundred space-dimensions ($d = 100$) as in this article (Fig. 8).

Assume the setting in Subsect. 4.1, let $\kappa = \frac{6}{10}$, $\lambda = \frac{1}{\sqrt{d}}$, assume for all $t \in [0, T]$, $x = (x_1, \ldots, x_d)$, $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^{1\times d}$, $m \in \mathbb{N}$ that $\gamma_m = \frac{1}{100} = 0.01$, $T = 1$, $d = 100$, $N = 30$, $\mu(t, x) = 0$, $\sigma(t, x)w = w$, $\xi = (0, 0, \ldots, 0) \in \mathbb{R}^d$, $g(x) = 1 + \kappa + \sin(\lambda \sum_{i=1}^d x_i)$, and

$$f(t, x, y, z) = \min\Big\{1, \big[y - \kappa - 1 - \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big)\big]^2\Big\}. \tag{4.47}$$

Note that the solution u of the PDE (4.1) then satisfies for all $t \in [0, T)$, $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\frac{\partial u}{\partial t}(t, x) + \min\Big\{1, \big[u(t, x) - \kappa - 1 - \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big)\big]^2\Big\} + \frac{1}{2}\,(\Delta_x u)(t, x) = 0 \tag{4.48}$$

(cf. Lemma 4.4). On the left hand side of Fig. 7, we present approximately the relative L^1-approximation error associated with $\mathcal{U}^{\Theta_m}$ against $m \in \{1, 2, 3, \ldots, 24{,}000\}$ based on 5 independent realizations (5 independent runs). On the right hand side of Fig. 7, we present approximately the mean of the loss function associated with Θ_m against $m \in \{1, 2, 3, \ldots, 24{,}000\}$ based on 256 Monte Carlo samples and 5 independent realizations (5 independent runs).

Lemma 4.4 (Cf. Subsection 6.1 in [16]) *Let $T, \kappa, \lambda \in (0, \infty)$, $d \in \mathbb{N}$ and let $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ be the function which satisfies for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ that*

$$u(t, x) = 1 + \kappa + \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big). \tag{4.49}$$

Then it holds for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ that $u \in C^{1,2}([0, T] \times \mathbb{R}^d, \mathbb{R})$, $u(T, x) = 1 + \kappa + \sin(\lambda \sum_{i=1}^d x_i)$, and

$$\frac{\partial u}{\partial t}(t, x) + \min\Big\{1, \big[u(t, x) - \kappa - 1 - \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big)\big]^2\Big\} + \frac{1}{2}\,(\Delta_x u)(t, x) = 0. \tag{4.50}$$

Proof of Lemma 4.4 Note that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\frac{\partial u}{\partial t}(t, x) = \frac{\lambda^2 d}{2} \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big). \tag{4.51}$$

In addition, observe that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, $k \in \{1, 2, \dots, d\}$ it holds that

$$\frac{\partial u}{\partial x_k}(t, x) = \lambda \cos\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big). \tag{4.52}$$

Hence, we obtain that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, $k \in \{1, \dots, d\}$ it holds that

$$\frac{\partial^2 u}{\partial x_k^2}(t, x) = -\lambda^2 \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big). \tag{4.53}$$

This ensures that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$(\Delta_x u)(t, x) = -\,d\,\lambda^2 \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big). \tag{4.54}$$

Combining this with (4.51) proves that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\frac{\partial u}{\partial t}(t, x) + \frac{1}{2}\,(\Delta_x u)(t, x) = 0. \tag{4.55}$$

This demonstrates that for all $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\frac{\partial u}{\partial t}(t, x) + \min\Big\{1, \big[u(t, x) - \kappa - 1 - \sin\big(\lambda \textstyle\sum_{i=1}^d x_i\big) \exp\big(\frac{\lambda^2 d(t-T)}{2}\big)\big]^2\Big\}$$

$$
\begin{aligned}
&+\frac{1}{2}(\Delta_x u)(t,x) \\
&= \frac{\partial u}{\partial t}(t,x) + \frac{1}{2}(\Delta_x u)(t,x) = 0. \qquad (4.56)
\end{aligned}
$$

The proof of Lemma 4.4 is thus completed. □

Acknowledgements Christian Beck and Sebastian Becker are gratefully acknowledged for useful suggestions regarding the implementation of the deep BSDE method. This project has been partially supported through the Major Program of NNSFC under grant 91130005, the research grant ONR N00014-13-1-0338, and the research grant DOE DE-SC0009248.

5 Appendix A: Special Cases of the Proposed Algorithm

In this section, we illustrate the general algorithm in Subsect. 3.2 in several special cases. More specifically, in Subsects. 5.1 and 5.2, we provide special choices for the functions ψ_m, $m \in \mathbb{N}$, and Ψ_m, $m \in \mathbb{N}$, employed in (3.14), and in Subsects. 5.3 and 5.4, we provide special choices for the function Υ in (3.9).

5.1 Stochastic Gradient Descent (SGD)

Example 5.1 Assume the setting in Subsect. 3.2, let $(\gamma_m)_{m\in\mathbb{N}} \subseteq (0,\infty)$, and assume for all $m \in \mathbb{N}$, $x \in \mathbb{R}^\varrho$, $(\varphi_j)_{j\in\mathbb{N}} \in (\mathbb{R}^\rho)^{\mathbb{N}}$ that

$$\varrho = \rho, \qquad \Psi_m(x, (\varphi_j)_{j\in\mathbb{N}}) = \varphi_1, \qquad \text{and} \qquad \psi_m(x) = \gamma_m x. \tag{5.1}$$

Then it holds for all $m \in \mathbb{N}$ that

$$\Theta_m = \Theta_{m-1} - \gamma_m \Phi^{m-1,1}_{\mathbb{S}_m}(\Theta_{m-1}). \tag{5.2}$$

5.2 Adaptive Moment Estimation (Adam) with Mini-Batches

In this subsection, we illustrate how the so-called Adam optimizer (see [25]) can be employed in conjunction with the deep BSDE method in Subsect. 3.2 (cf. also Subsect. 4.1 above).

Example 5.2 Assume the setting in Subsect. 3.2, assume that $\varrho = 2\rho$, let $\mathrm{Pow}_r\colon \mathbb{R}^\rho \to \mathbb{R}^\rho$, $r \in (0,\infty)$, be the functions which satisfy for all $r \in (0,\infty)$, $x = (x_1,\ldots,x_\rho) \in \mathbb{R}^\rho$ that

$$\mathrm{Pow}_r(x) = (|x_1|^r, \ldots, |x_\rho|^r), \tag{5.3}$$

let $\varepsilon \in (0,\infty)$, $(\gamma_m)_{m\in\mathbb{N}} \subseteq (0,\infty)$, $(J_m)_{m\in\mathbb{N}_0} \subseteq \mathbb{N}$, $\mathbb{X}, \mathbb{Y} \in (0,1)$, let $\mathbf{m} = (\mathbf{m}^{(1)},\ldots,\mathbf{m}^{(\rho)})\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\rho$ and $\mathbb{M} = (\mathbb{M}^{(1)},\ldots\mathbb{M}^{(\rho)})\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^\rho$ be the stochastic processes which satisfy for all $m \in \mathbb{N}_0$ that $\Xi_m =$

$(\mathbf{m}_m^{(1)}, \dots, \mathbf{m}_m^{(\rho)}, \mathbb{M}_m^{(1)}, \dots, \mathbb{M}_m^{(\rho)})$, and assume for all $m \in \mathbb{N}$, $x = (x_1, \dots, x_\rho)$, $y = (y_1, \dots, y_\rho) \in \mathbb{R}^\rho$, $(\varphi_j)_{j\in\mathbb{N}} \in (\mathbb{R}^\rho)^{\mathbb{N}}$ that

$$\Psi_m(x, y, (\varphi_j)_{j\in\mathbb{N}}) = \big(\mathbb{X}x + (1-\mathbb{X})\big(\tfrac{1}{J_m}\textstyle\sum_{j=1}^{J_m}\varphi_j\big), \mathbb{Y}y + (1-\mathbb{Y}) \\ \operatorname{Pow}_2\big(\frac{1}{J_m}\textstyle\sum_{j=1}^{J_m}\varphi_j\big)\big) \tag{5.4}$$

and

$$\psi_m(x, y) = \left(\left[\varepsilon + \frac{\sqrt{|y_1|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m x_1}{(1-\mathbb{X}^m)}, \dots, \left[\varepsilon + \frac{\sqrt{|y_\rho|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m x_\rho}{(1-\mathbb{X}^m)}\right). \tag{5.5}$$

Then it holds for all $m \in \mathbb{N}$ that

$$\begin{aligned} \Theta_m &= \Theta_{m-1} - \left(\left[\varepsilon + \frac{\sqrt{|\mathbb{M}_m^{(1)}|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m \mathbf{m}_m^{(1)}}{(1-\mathbb{X}^m)}, \dots, \left[\varepsilon + \frac{\sqrt{|\mathbb{M}_m^{(\rho)}|}}{\sqrt{1-\mathbb{Y}^m}}\right]^{-1} \frac{\gamma_m \mathbf{m}_m^{(\rho)}}{(1-\mathbb{X}^m)}\right), \\ \mathbf{m}_m &= \mathbb{X}\,\mathbf{m}_{m-1} + \frac{(1-\mathbb{X})}{J_m}\left(\sum_{j=1}^{J_m} \Phi_{\mathbb{S}_m}^{m-1,j}(\Theta_{m-1})\right), \\ \mathbb{M}_m &= \mathbb{Y}\,\mathbb{M}_{m-1} + (1-\mathbb{Y})\operatorname{Pow}_2\left(\frac{1}{J_m}\sum_{j=1}^{J_m} \Phi_{\mathbb{S}_m}^{m-1,j}(\Theta_{m-1})\right). \end{aligned} \tag{5.6}$$

5.3 Euler–Maruyama Scheme

Example 5.3 Assume the setting in Subsect. 3.2, let $\mu\colon [0,T]\times\mathbb{R}^d \to \mathbb{R}^d$ and $\sigma\colon [0,T]\times\mathbb{R}^d \to \mathbb{R}^d$ be functions, and assume for all $s, t \in [0,T]$, $x, w \in \mathbb{R}^d$ that

$$\Upsilon(s, t, x, w) = x + \mu(s, x)\,(t-s) + \sigma(s, x)\,w. \tag{5.7}$$

Then it holds for all $m, j \in \mathbb{N}_0$, $n \in \{0, 1, \dots, N-1\}$ that

$$\mathcal{X}_n^{m,j} = \mathcal{X}_n^{m,j} + \mu\left(t_n, \mathcal{X}_n^{m,j}\right)(t_{n+1}-t_n) + \sigma\left(t_n, \mathcal{X}_n^{m,j}\right)\left(W_{t_{n+1}} - W_{t_n}\right). \tag{5.8}$$

In the setting of Example 5.3, we consider under suitable further hypotheses for every sufficiently large $m \in \mathbb{N}_0$ the random variable $\mathcal{U}^{\Theta_m}$ as an approximation of $u(0,\xi)$ where $u\colon [0,T]\times\mathbb{R}^d \to \mathbb{R}^k$ is a suitable solution of the PDE

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\sum_{j=1}^{d}\left(\frac{\partial^2 u}{\partial x^2}\right)(t,x)\left[\sigma(t,x)\,e_j^{(d)}, \sigma(t,x)\,e_j^{(d)}\right] + \left(\frac{\partial u}{\partial x}\right)(t,x)\,\mu(t,x) \\ + f\left(t, x, u(t,x), \left(\frac{\partial u}{\partial x}\right)(t,x)\,\sigma(t,x)\right) = 0 \tag{5.9}$$

with $u(T,x) = g(x)$, $e_1^{(d)} = (1,0,\dots,0)$, $\dots$, $e_d^{(d)} = (0,\dots,0,1) \in \mathbb{R}^d$ for $t \in [0,T]$, $x = (x_1,\dots,x_d) \in \mathbb{R}^d$ (cf. (PDE) in Sect. 2 above).

5.4 Geometric Brownian Motion

Example 5.4 Assume the setting in Subsect. 3.2, let $\bar{\mu}, \bar{\sigma} \in \mathbb{R}$, and assume for all $s,t \in [0,T]$, $x = (x_1,\dots,x_d)$, $w = (w_1,\dots,w_d) \in \mathbb{R}^d$ that

$$\Upsilon(s,t,x,w) = \exp\left(\left(\bar{\mu} - \frac{\bar{\sigma}^2}{2}\right)(t-s)\right) \exp\left(\bar{\sigma}\, \operatorname{diag}_{\mathbb{R}^{d\times d}}(w_1,\dots,w_d)\right) x. \quad (5.10)$$

Then it holds for all $m, j \in \mathbb{N}_0$, $n \in \{0,1,\dots,N\}$ that

$$\mathcal{X}_n^{\theta,m,j} = \exp\left(\left(\bar{\mu} - \frac{\bar{\sigma}^2}{2}\right) t_n \operatorname{Id}_{\mathbb{R}^d} + \bar{\sigma}\, \operatorname{diag}_{\mathbb{R}^{d\times d}}\left(W_{t_n}^{m,j}\right)\right) \xi. \quad (5.11)$$

In the setting of Example 5.4 we view under suitable further hypotheses (cf. Subsect. 4.4 above) for every sufficiently large $m \in \mathbb{N}_0$ the random variable $\mathcal{U}^{\Theta_m}$ as an approximation of $u(0,\xi)$ where $u\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^k$ is a suitable solution of the PDE

$$\begin{aligned} &\tfrac{\partial u}{\partial t}(t,x) + \tfrac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \left(\tfrac{\partial^2 u}{\partial x_i^2}\right)(t,x) + \bar{\mu} \sum_{i=1}^{d} x_i \left(\tfrac{\partial u}{\partial x_i}\right)(t,x) \\ &\quad + f\left(t, x, u(t,x), \bar{\sigma} \left(\tfrac{\partial u}{\partial x}\right)(t,x) \operatorname{diag}_{\mathbb{R}^{d\times d}}(x_1,\dots,x_d)\right) = 0 \end{aligned} \quad (5.12)$$

with $u(T,x) = g(x)$ for $t \in [0,T]$, $x = (x_1,\dots,x_d) \in \mathbb{R}^d$.

References

1. Bellman, R.: Dynamic programming. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ. Reprint of the 1957 edition, with a new introduction by Stuart Dreyfus (2010)
2. Bender, C., Denk, R.: A forward scheme for backward SDEs. Stoch. Process. Appl. **117**(12), 1793–1812 (2007)
3. Bender, C., Schweizer, N., Zhuo, J.: A primal-dual algorithm for BSDEs. arXiv:1310.3694 (2014)
4. Bergman, Y.Z.: Option pricing with differential interest rates. Rev. Financ. Stud. **8**(2), 475–500 (1995)
5. Briand, P., Labart, C.: Simulation of BSDEs by Wiener chaos expansion. Ann. Appl. Probab. **24**(3), 1129–1171 (2014)
6. Chassagneux, J.-F.: Linear multistep schemes for BSDEs. SIAM J. Numer. Anal. **52**(6), 2815–2836 (2014)
7. Chassagneux, J.-F., Richou, A.: Numerical simulation of quadratic BSDEs. Ann. Appl. Probab. **26**(1), 262–304 (2016)
8. Crisan, D., Manolarakis, K.: Solving backward stochastic differential equations using the cubature method: application to nonlinear pricing. SIAM J. Financ. Math. **3**(1), 534–571 (2012)
9. Darbon, J., Osher, S.: Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. Res. Math. Sci. **3**(19), 26 (2016)
10. Debnath, L.: Nonlinear Partial Differential Equations for Scientists and Engineers, 3rd edn. Birkhäuser/Springer, New York (2012)

11. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. arXiv:1706.04702 (2017)
12. E, W., Hutzenthaler, M., Jentzen, A., Kruse, T.: Linear scaling algorithms for solving high-dimensional nonlinear parabolic differential equations. arXiv:1607.03295 (2017)
13. E, W., Hutzenthaler, M., Jentzen, A., Kruse, T.: On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. arXiv:1708.03223 (2017)
14. Gobet, E., Lemor, J.-P., Warin, X.: A regression-based Monte Carlo method to solve backward stochastic differential equations. Ann. Appl. Probab. **15**(3), 2172–2202 (2005)
15. Gobet, E., Turkedjiev, P.: Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions. Math. Comput. **85**(299), 1359–1391 (2016)
16. Gobet, E., Turkedjiev, P.: Adaptive importance sampling in least-squares Monte Carlo algorithms for backward stochastic differential equations. Stoch. Process. Appl. **127**(4), 1171–1203 (2017)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press 2016. http://www.deeplearningbook.org
18. Han, J., E, W.: Deep learning approximation for stochastic control problems. arXiv:1611.07422 (2016)
19. Han, J., Jentzen, A., E, W.: Overcoming the curse of dimensionality: solving high-dimensional partial differential equations using deep learning. arXiv:1707.02568 (2017)
20. Henry-Labordère, P.: Counterparty risk valuation: a marked branching diffusion approach. arXiv:1203.2369 (2012)
21. Henry-Labordère, P., Oudjane, N., Tan, X., Touzi, N., Warin, X.: Branching diffusion representation of semilinear PDEs and Monte Carlo approximation. arXiv:1603.01727 (2016)
22. Henry-Labordère, P., Tan, X., Touzi, N.: A numerical algorithm for a class of BSDEs via the branching process. Stoch. Process. Appl. **124**(2), 1112–1140 (2014)
23. Hinton, G.E., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition. Sig. Process. Mag. **29**, 82–97 (2012)
24. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning (ICML) (2015)
25. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (ICLR) (2015)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
27. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**, 436–444 (2015)
28. Pardoux, É., Peng, S.: Adapted solution of a backward stochastic differential equation. Syst. Control Lett. **14**(1), 55–61 (1990)
29. Pardoux, É., Peng, S.: Backward stochastic differential equations and quasilinear parabolic partial differential equations. In: Stochastic Partial Differential Equations and Their Applications (Charlotte, NC, 1991), vol. 176 of Lecture Notes in Control and Inform. Sci. Springer, Berlin, pp. 200–217 (1992)
30. Pardoux, É., Tang, S.: Forward-backward stochastic differential equations and quasilinear parabolic PDEs. Probab. Theory Relat. Fields **114**(2), 123–150 (1999)
31. Peng, S.: Probabilistic interpretation for systems of quasilinear parabolic partial differential equations. Stoch. Stoch. Rep. **37**(1–2), 61–74 (1991)