Journal of Scientific Computing (2019) 79:1534–1571
https://doi.org/10.1007/s10915-018-00903-0

On Multilevel Picard Numerical Approximations for High-Dimensional Nonlinear Parabolic Partial Differential Equations and High-Dimensional Nonlinear Backward Stochastic Differential Equations

Weinan E[1] · Martin Hutzenthaler[2] · Arnulf Jentzen[3] · Thomas Kruse[2]

Received: 28 August 2017 / Revised: 11 December 2018 / Accepted: 29 December 2018 /
Published online: 7 March 2019

Abstract
Parabolic partial differential equations (PDEs) and backward stochastic differential equations (BSDEs) are key ingredients in a number of models in physics and financial engineering. In particular, parabolic PDEs and BSDEs are fundamental tools in pricing and hedging models for financial derivatives. The PDEs and BSDEs appearing in such applications are often high-dimensional and nonlinear. Since explicit solutions of such PDEs and BSDEs are typically not available, it is a very active topic of research to solve such PDEs and BSDEs approximately. In the recent article (E et al., Multilevel Picard iterations for solving smooth semilinear parabolic heat equations, arXiv:1607.03295) we proposed a family of approximation methods based on Picard approximations and multilevel Monte Carlo methods and showed under suitable regularity assumptions on the exact solution of a semilinear heat equation that the computational complexity is bounded by $O(d\,\varepsilon^{-(4+\delta)})$ for any $\delta \in (0,\infty)$ where d is the dimensionality of the problem and $\varepsilon \in (0,\infty)$ is the prescribed accuracy. In this paper, we test the applicability of this algorithm on a variety of 100-dimensional nonlinear PDEs that arise in physics and finance by means of numerical simulations presenting approximation accuracy against runtime. The simulation results for many of these 100-dimensional example PDEs are very satisfactory in terms of both accuracy and speed. Moreover, we also provide a review of other approximation methods for nonlinear PDEs and BSDEs from the scientific literature.

Keywords Curse of dimensionality · High-dimensional PDEs · High-dimensional nonlinear BSDEs · Multilevel Picard approximations · Multilevel Monte Carlo method

Mathematics Subject Classification 65M75

Contents

Extended author information available on the last page of the article

1 Introduction

Parabolic partial differential equations (PDEs) and backward stochastic differential equations (BSDEs) have a wide range of applications. To give specific examples we focus now on a number of applications in finance. There are several fundamental assumptions incorporated in the Black–Scholes model that are not met in the real-life trading of financial derivatives. A number of derivative pricing models have been developed in about the last four decades to relax these assumptions; see, e.g., [8,9,18,28,41,61,64] for models taking into account the fact that the "risk-free" bank account has higher interest rates for borrowing than for lending, particularly, due to the default risk of the trader, see, e.g., [18,53] for models incorporating the default risk of the issuer of the financial derivative, see, e.g., [5,6,85] for models for the pricing of financial derivatives on underlyings which are not tradeable such as financial derivatives on the temperature or mortality-dependent financial derivatives, see, e.g., [1] for models incorporating that the hedging strategy influences the price processes through demand and supply (so-called large investor effects), see, e.g., [32,47,63] for models taking the transaction costs in the hedging portfolio into account, and see, e.g., [2,47] for models incorporating uncertainties in the model parameters for the underlying. In each of the above references the value function u, describing the price of the financial derivative, solves a *nonlinear* parabolic PDE. Moreover, the PDEs for the value functions emerging from the above models are often high-dimensional as the financial derivative depends in several cases on a whole basket of underlyings and as a portfolio containing several financial derivatives must often be treated as a whole in the case where the above nonlinear effects are taken into account (cf., e.g., [8,18,32]). These high-dimensional nonlinear PDEs can typically not be solved explicitly and, in particular, there is a strong demand from the financial engineering industry to approximately compute the solutions of such high-dimensional nonlinear parabolic PDEs.

The numerical analysis literature contains a number of deterministic approximation methods for nonlinear parabolic PDEs such as finite element methods, finite difference methods, spectral Galerkin approximation methods, or sparse grid methods (cf., e.g., [83, Chapter 14], [82, Sect. 3], [81], and [76]). Some of these approximation methods achieve high convergence rates with respect to the computational effort and, in particular, provide efficient approximations in low or moderate dimensions. However, these approximation methods can not be used in high dimensions as the computational effort grows *exponentially* in the dimension $d \in \mathbb{N} = \{1, 2, \ldots\}$ of the considered nonlinear parabolic PDE and then the approximation method fails to terminate within years even for low accuracies.

In the case of linear parabolic PDEs the *Feynman–Kac formula* establishes an explicit representation of the solution of the PDE as the expectation of the solution of an appropriate stochastic differential equation (SDE). (Multilevel) Monte Carlo methods together with suitable discretizations of the SDE (see, e.g., [56,58,60,71]) then result in a numerical approximation method with a computational effort that grows under suitable hypotheses at most polynomially in the dimension $d \in \mathbb{N}$ of the PDE and that grows up to an arbitrarily small order quadratically in the reciprocal of the approximation precision (cf., e.g., [38,45,49,50]). These multilevel Monte Carlo approximations are, however, limited to *linear* PDEs as the classical Feynman–Kac formula provides only in the case of a linear PDE an explicit representation of the solution of the PDE. For lower error bounds in the literature on random and deterministic numerical approximation methods for high-dimensional linear PDEs the reader is, e.g., referred to Heinrich [51, Theorem 1].

In the seminal papers [73–75], Pardoux and Peng developed the theory of nonlinear backward stochastic differential equations and, in particular, established a considerably generalized nonlinear Feynman–Kac formula to obtain an explicit representation of the solution of a nonlinear parabolic PDE by means of the solution of an appropriate BSDE; see also Cheridito et al. [17] for second-order BSDEs. Discretizations of BSDEs, however, require suitable discretizations of nested conditional expectations (see, e.g., [10,17,31,46,86]). Discretization methods for these nested conditional expectations proposed in the literature include the 'straight forward' Monte Carlo method, the quantization tree method (see [4]), the regression method based on Malliavin calculus or based on kernel estimation (see [10]), the projection on function spaces method (see [41]), the cubature on Wiener space method (see [22]), and the Wiener chaos decomposition method (see [12]). None of these discretization methods has the property that the computational effort of the method grows at most polynomially both in the dimension and in the reciprocal of the prescribed accuracy (see Sects. 4.1–4.6 below for a detailed discussion). We note that solving high-dimensional semilinear parabolic PDEs at single space-time points and solving high-dimensional nonlinear BSDEs at single time points is essentially equivalent due to the generalized nonlinear Feynman–Kac formula established by Pardoux and Peng. In recent years the concept of fractional smoothness in the sense of function spaces has been used for studying variational properties of BSDEs. This concept of fractional smoothness quantifies the propagation of singularities in time and shows that certain non-uniform time grids are more suitable in the presence of singularities; see, e.g., Geiss and Geiss [35], Gobet and Makhlouf [42] or Geiss et al. [34] for details. Also these temporal discretization methods require suitable discretizations of nested conditional expectations resulting in the same problems as in the case of uniform time grids.

Another probabilistic representation for solutions of some nonlinear parabolic PDEs with polynomial nonlinearities has been established in Skorohod [80] by means of *branching diffusion processes*. Recently, this classical representation has been extended under suitable assumptions in Henry-Labordère [53] to more general analytic nonlinearities and in Henry-Labordère et al. [54] to polynomial nonlinearities in the pair $(u(t,x), (\nabla_x u)(t,x)) \in \mathbb{R}^{1+d}$, $t \in [0,T]$, $x \in \mathbb{R}^d$, where u is the solution of the PDE, $d \in \mathbb{N}$ is the dimension, and $T \in (0,\infty)$ is the time horizon. This probabilistic representation has been successfully used in Henry-Labordère [53] (see also Henry-Labordère et al. [55]) and in Henry-Labordère et al. [54] to obtain a Monte Carlo approximation method for semilinear parabolic PDEs with a computational complexity which is bounded by $O(d\,\varepsilon^{-2})$ where d is the dimensionality of the problem and $\varepsilon \in (0,\infty)$ is the prescribed accuracy. The major drawback of the branching diffusion method is its insufficient applicability, namely it requires the terminal/initial condition of the parabolic PDE to be quite small (see Sect. 4.7 below for a detailed discussion).

In the recent article [27] we proposed a family of approximation methods which we denote as multilevel Picard approximations (see (9) for their definitions and Sect. 2 for their derivations). Corollary 3.18 in [27] shows under suitable regularity assumptions (including smoothness and Lipschitz continuity) on the exact solution that the computational complexity of this algorithm is bounded by $O(d\,\varepsilon^{-(4+\delta)})$ for any $\delta \in (0,\infty)$, where d is the dimensionality of the problem and $\varepsilon \in (0,\infty)$ is the prescribed accuracy. In this paper we complement the theoretical complexity analysis of [27] with a simulation study. Our simulations in Sect. 3 indicate that the computational complexity grows at most linearly in the dimension and quartically in the reciprocal of the prescribed accuracy also for several 100-dimensional nonlinear PDEs from physics and finance with non-smooth and/or non-Lipschitz nonlinearities and terminal condition functions. The simulation results for many of these 100-dimensional example PDEs are very satisfactory in terms of accuracy and speed.

1.1 Notation

Throughout this article we frequently use the following notation. We denote by $\langle\cdot,\cdot\rangle\colon (\cup_{n\in\mathbb{N}}(\mathbb{R}^n\times\mathbb{R}^n))\to[0,\infty)$ the function that satisfies for all $n\in\mathbb{N}$, $v=(v_1,\ldots,v_n)$, $w=(w_1,\ldots,w_n)\in\mathbb{R}^n$ that $\langle v,w\rangle=\sum_{i=1}^n v_i w_i$. For every topological space $(E,\mathcal{E})$ we denote by $\mathcal{B}(E)$ the Borel-sigma-algebra on $(E,\mathcal{E})$. For all measurable spaces $(A,\mathcal{A})$ and $(B,\mathcal{B})$ we denote by $\mathcal{M}(\mathcal{A},\mathcal{B})$ the set of $\mathcal{A}/\mathcal{B}$-measurable functions from A to B. For every probability space $(\Omega,\mathcal{A},\mathbb{P})$ we denote by $\|\cdot\|_{L^2(\mathbb{P};\mathbb{R})}\colon\mathcal{M}(\mathcal{A},\mathcal{B}(\mathbb{R}))\to[0,\infty]$ the function that satisfies for all $X\in\mathcal{M}(\mathcal{A},\mathcal{B}(\mathbb{R}))$ that $\|X\|_{L^2(\mathbb{P};\mathbb{R})}=\sqrt{\mathbb{E}\big[|X|^2\big]}$. For all metric spaces (E,d_E) and (F,d_F) we denote by $\operatorname{Lip}(E,F)$ the set of all globally Lipschitz continuous functions from E to F. For every $d\in\mathbb{N}$ we denote by $\mathrm{I}_{\mathbb{R}^{d\times d}}$ the identity matrix in $\mathbb{R}^{d\times d}$ and we denote by $\mathbb{R}^{d\times d}_{\mathrm{Inv}}$ the set of invertible matrices in $\mathbb{R}^{d\times d}$. For every $d\in\mathbb{N}$ and every $A\in\mathbb{R}^{d\times d}$ we denote by $A^*\in\mathbb{R}^{d\times d}$ the transpose of A. For every $d\in\mathbb{N}$ and every $x=(x_1,\ldots,x_d)\in\mathbb{R}^d$ we denote by $\operatorname{diag}(x)\in\mathbb{R}^{d\times d}$ the diagonal matrix with diagonal entries $x_1,\ldots,x_d$. For every $T\in(0,\infty)$ we denote by $\mathcal{Q}_T$ the set given by $\mathcal{Q}_T=\{w\colon[0,T]\to\mathbb{R}\colon w^{-1}(\mathbb{R}\backslash\{0\})\text{ is a finite set}\}$. For every set A and every function $f\colon[0,T]\to[0,\infty]$ we write We denote by $\lfloor\cdot\rfloor\colon\mathbb{R}\to\mathbb{Z}$ and $[\cdot]^+\colon\mathbb{R}\to[0,\infty)$ the functions that satisfy for all $x\in\mathbb{R}$ that $\lfloor x\rfloor=\max(\mathbb{Z}\cap(-\infty,x])$ and $[x]^+=\max\{x,0\}$.

2 Multilevel Picard Approximations for High-Dimensional Semilinear PDEs

For this article to be self-contained, we recall the derivation of multilevel Picard approximations from [27]. In Sect. 2.3 below we define multilevel Picard approximations (see (9) below) in the case of semilinear PDEs (cf. (6) in Sect. 2.2 below). In Sect. 2.1 we explain the ideas behind these approximations in the special case of gradient-independent nonlinearities and in the case of Brownian motion as forward diffusion. The case of general semilinear PDEs will then be explained in Sect. 2.2.

2.1 Derivation of the Multilevel Picard Algorithm for Gradient-Independent Nonlinearities

Multilevel Picard approximations are based on the Picard approximation method of the solution of a fixed-point equation in order to avoid unfavorable error propagation in time of an explicit or implicit Euler-type method; cf. the discussion in Sect. 4 below. For this we first derive a fixed-point equation for the solution of a semilinear PDE. Then we discretize the fixed-point operator in a suitable way. To simplify the presentation, we only consider in this subsection the special case of gradient-independent nonlinearities and the case of the Brownian motion as the forward diffusion. We refer to Sect. 2.2 below for the more general case.

For the rest of this subsection let $T \in (0, \infty)$, $d \in \mathbb{N}$, let $g\colon \mathbb{R}^d \to \mathbb{R}$, $f\colon [0,T] \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, and $u\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}$ be sufficiently regular functions, assume for all $t \in [0,T)$, $x \in \mathbb{R}^d$ that $u(T,x) = g(x)$ and

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

let $(\Omega, \mathcal{F}, \mathbb{P}, (\mathbb{F}_t)_{t\in[0,T]})$ be a stochastic basis (cf., e.g., [77, Appendix E]), let $W\colon [0,T] \times \Omega \to \mathbb{R}^d$ be a standard $(\mathbb{F}_t)_{t\in[0,T]}$-Brownian motion, and let $\Phi\colon \mathrm{Lip}([0,T] \times \mathbb{R}^d, \mathbb{R}) \to \mathrm{Lip}([0,T] \times \mathbb{R}^d, \mathbb{R})$ satisfy for all $v \in \mathrm{Lip}([0,T] \times \mathbb{R}^d, \mathbb{R})$, $s \in [0,T)$, $x \in \mathbb{R}^d$ that

$$\begin{aligned}\big(\Phi(v)\big)(s,x) &= \mathbb{E}\big[g(x + W_T - W_s)\big] \\ &\quad + \int_s^T \mathbb{E}\Big[f\Big(t, x + W_t - W_s, v\big(t, x + W_t - W_s\big)\Big)\Big]\,dt.\end{aligned} \tag{2}$$

Our approximation scheme in (9) below is based on a suitable *fixed-point formulation* of the solution u of the PDE (1). More precisely, the *Feynman–Kac formula* implies that $u = \Phi(u)$.

Now the expectation and the time integral in (2) are non-discrete and, in general, cannot be simulated on a computer. For this reason we approximate the non-discrete quantities in (2) (expectation and time integral) by discrete quantities (Monte Carlo averages and quadrature formulas). For this let $(q_s^{n,\rho})_{n,\rho\in\mathbb{N}_0, s\in[0,T)} \subseteq \mathcal{Q}_T$ be quadrature rules which are just functions on $[0,T]$ which have non-zero values only on a finite subset of $[0,T]$. In addition, let $(m_{n,\rho})_{n,\rho\in\mathbb{N}_0} \subseteq \mathbb{N}_0$ be natural numbers which will be the numbers of Monte Carlo averages. For the rest of this paragraph we assume for all $s \in [0,T]$, $n, \rho \in \mathbb{N}$ that $m_{n,\rho} = \rho^{2n}$ and that $q_s^{n,\rho}$ is the left-rectangle rule on the interval $[s,T]$ with ρ^n rectangles so that for all $t \in [s,T]$ it holds that $q_s^{n,\rho}(t) = \frac{(T-s)}{\rho^n}\mathbb{1}_{\{s+i\frac{(T-s)}{\rho^n}\colon i\in\mathbb{N}_0\}}(t)$. Moreover, let $W^{i,n}\colon [0,T] \times \Omega \to \mathbb{R}^d$, $i, n \in \mathbb{N}_0$, be independent standard $(\mathbb{F}_t)_{t\in[0,T]}$-Brownian motions. Furthermore, for every $n, \rho \in \mathbb{N}_0$ and every function $V\colon [0,T] \times \mathbb{R}^d \times \Omega \to \mathbb{R}$ let $(V^i)_{i\in\mathbb{N}}$ be independent and identically distributed versions of V and let $\Psi_{n,\rho}(V)\colon [0,T] \times \mathbb{R}^d \times \Omega \to \mathbb{R}$ be the function which satisfies for all $s \in [0,T]$, $x \in \mathbb{R}^d$ that

$$\begin{aligned}(\Psi_{n,\rho}(V))(s,x) = \tfrac{1}{m_{n,\rho}} \sum_{i=1}^{m_{n,\rho}} \Bigg[& g(x + W_T^{i,n} - W_s^{i,n}) \\ & + \sum_{t\in[s,T]} q_s^{n,\rho}(t) f\Big(t, x + W_t^{i,n} - W_s^{i,n}, V^i(t, x + W_t^{i,n} - W_s^{i,n})\Big)\Bigg].\end{aligned} \tag{3}$$

With this we recursively define $(V_{n,\rho})_{n,\rho\in\mathbb{N}}$ such that for all $n, \rho \in \mathbb{N}$, $t \in [0,T]$, $x \in \mathbb{R}^d$ it holds that $V_{1,\rho}(t,x) = (\Psi_{1,\rho}(0))(t,x)$ and $V_{n+1,\rho}(t,x) = \big(\Psi_{n,\rho}(V_{n,\rho})\big)(t,x)$. We note that Bender and Denk [7] studied a non-implementable version hereof where expectations are not discretized. For every $n, \rho \in \mathbb{N}$ let $e_{n,\rho}$ be the number of function evaluations of g and f required to compute one realization of $V_{n,\rho}(0,0)$. Then this computational effort satisfies for all $n, \rho \in \mathbb{N}$ the recursive equation $e_{n+1,\rho} = \rho^{2n}(1 + \rho^n(1 + e_{n,\rho}))$. Hence, we obtain for all $n, \rho \in \mathbb{N}$ that $e_{n,\rho} \in [\rho^{3n^2}, \rho^{3n^2}3^n]$. Moreover, we observe in the special case where $f \equiv 0$ and g is bounded that the approximation error satisfies $\sup_{n,\rho\in\mathbb{N}} \|u^\infty(0,0) - V_{n,\rho}(0,0)\|_{L^2(\mathbb{P};\mathbb{R})}\rho^{-n} < \infty$. This suggests that the numerical approximations $(V_{n,\rho})_{n,\rho\in\mathbb{N}}$ do not converge with a strictly positive polynomial rate of convergence. For this reason we modify the above approximation method in a subtle way so that the computational effort is drastically reduced. We also mention that the approximations $(V_{n,\rho})_{n,\rho\in\mathbb{N}}$ can be improved by an approach with control variates which reduce the variance; see Gobet and Labart [39] for details.

The central new idea in our multilevel Picard approximations is to adapt in a suitable manner the multilevel Monte Carlo approach of Heinrich [49,50] and Giles [38] to the Picard approximation method. More precisely, let $(\mathfrak{u}_n)_{n\in\mathbb{N}_0} \subseteq \operatorname{Lip}([0,T]\times\mathbb{R}^d, \mathbb{R})$ be the Picard approximations satisfying for all $n \in \mathbb{N}_0$ that $\mathfrak{u}_{n+1} = \Phi(\mathfrak{u}_n)$. The Banach fixed-point theorem then ensures that the sequence $(\mathfrak{u}_n)_{n\in\mathbb{N}_0}$ converges at least *exponentially fast* to u. In the next step, we note that the fact that $\lim_{n\to\infty}\mathfrak{u}_n = u$, a telescope sum argument, and the fact that for all $n \in \mathbb{N}$ it holds that $\mathfrak{u}_n = \Phi(\mathfrak{u}_{n-1})$ ensure for all sufficiently large $n \in \mathbb{N}$ that

$$\begin{aligned} u \approx \mathfrak{u}_n = \mathfrak{u}_1 + \sum_{l=1}^{n-1}\big[\mathfrak{u}_{l+1} - \mathfrak{u}_l\big] &= \Phi(\mathfrak{u}_0) + \sum_{l=1}^{n-1}\Big[\Phi(\mathfrak{u}_l) - \Phi(\mathfrak{u}_{l-1})\Big] \\ &\approx \Psi_{n,\rho}(u_0) + \sum_{l=1}^{n-1}\Big[\Psi_{n-l,\rho}(\mathfrak{u}_l) - \Psi_{n-l,\rho}(\mathfrak{u}_{l-1})\Big]. \end{aligned} \tag{4}$$

Display (4) suggests to introduce numerical approximations as follows. Let $(U_{n,\rho})_{n,\rho\in\mathbb{N}_0} \subseteq \operatorname{Lip}([0,T]\times\mathbb{R}^d,\mathbb{R})$ satisfy for all $n, \rho \in \mathbb{N}$ that $U_{0,\rho} = u_0$ and that

$$U_{n,\rho} = \Psi_{n,\rho}(U_{0,\rho}) + \sum_{l=1}^{n-1}\Big[\Psi_{n-l,\rho}\big(U_{l,\rho}\big) - \Psi_{n-l,\rho}\big(U_{l-1,\rho}\big)\Big]. \tag{5}$$

A key feature of the approximations in (5) is that the approximations (5) keep the computational cost moderate compared to the desired approximation precision. More precisely, for every $n, \rho \in \mathbb{N}$ let $e_{n,\rho}$ be the number of function evaluations of f and g required to compute one realization of $U_{n,\rho}(0,0)$. Note that this computational effort satisfies for all $n, \rho \in \mathbb{N}$ the recursive equation $e_{n,\rho} = \rho^{2n}(1+\rho^n) + \sum_{l=1}^{n-1}\big(\rho^{2(n-l)}(1 + \rho^{n-l}(e_{l,\rho} + e_{l-1,\rho}))\big)$. Hence, we obtain for all $n, \rho \in \mathbb{N}$ that $e_{n,\rho} \le 2\cdot 3^n\rho^{3n}$. Let us mention that the numerical approximations specified in (9) in Sect. 2.3 below differ from (5) in the way that essentially all Brownian motions appearing in (9) are assumed to be independent and this difference considerably simplifies their implementation.

We would like to point out that the approximations in (5) are *full history recursive* in the sense that for every $k, \rho \in \mathbb{N}$ the “full history” $U_{0,\rho}, U_{1,\rho}, \ldots, U_{k-1,\rho}$ needs to be computed recursively in order to compute $U_{k,\rho}$. Moreover, we note that the approximations in (5) exploit multilevel/multigrid ideas (cf., e.g., [38,49,50,52]). Typically multilevel ideas appear where the different levels correspond to approximations with different time or space step sizes while here different levels correspond to different stages of the fixed-point iteration. This, in turn, results in numerical approximations which require the full history of the approximations.

2.2 A Fixed-Point Equation for General Semilinear PDEs

The derivation of the approximation method in (9) in the case of a gradient-dependent nonlinearity and a more general forward diffusion is analogous to the special case of a gradient-independent nonlinearity and of a Brownian motion as the forward diffusion as considered in Sect. 2.1. Only the derivation of an appropriate fixed-point equation is nontrivial. For the derivation of this fixed-point equation we impose for simplicity of presentation appropriate additional hypotheses that are not needed for the definition of the scheme in (9) (cf. (6)–(8) in this subsection with Sect. 2.3).

Let $T \in (0, \infty)$, $d \in \mathbb{N}$, let $g\colon \mathbb{R}^d \to \mathbb{R}$, $f\colon [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$, $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$, $\eta\colon \mathbb{R}^d \to \mathbb{R}^d$, $\mu\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$, and $\sigma = (\sigma_1, \ldots, \sigma_d)\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}_{\text{Inv}}$ be sufficiently regular functions, assume for all $t \in [0, T)$, $x \in \mathbb{R}^d$ that $u(T, x) = g(x)$ and

$$\begin{aligned}&(\tfrac{\partial}{\partial t}u)(t,x) + f\big(t, x, u(t,\eta(x)), [\sigma(t,\eta(x))]^*(\nabla_x u)(t,\eta(x))\big) + \langle \mu(t,x), (\nabla_x u)(t,x)\rangle \\ &\quad + \tfrac{1}{2}\,\mathrm{Trace}\big(\sigma(t,x)[\sigma(t,x)]^*(\mathrm{Hess}_x\, u)(t,x)\big) = 0,\end{aligned} \tag{6}$$

let $(\Omega, \mathcal{F}, \mathbb{P}, (\mathbb{F}_t)_{t\in[0,T]})$ be a stochastic basis, let $W = (W^1, \ldots, W^d)\colon [0, T]\times\Omega \to \mathbb{R}^d$ be a standard $(\mathbb{F}_t)_{t\in[0,T]}$-Brownian motion, and for every $s \in [0, T]$, $x \in \mathbb{R}^d$ let $X^{s,x}\colon [s, T] \times \Omega \to \mathbb{R}^d$ and $D^{s,x}\colon [s, T] \times \Omega \to \mathbb{R}^{d\times d}$ be $(\mathbb{F}_t)_{t\in[s,T]}$-adapted stochastic processes with continuous sample paths which satisfy that for all $t \in [s, T]$ it holds $\mathbb{P}$-a.s. that

$$\begin{aligned}X^{s,x}_t &= x + \int_s^t \mu(r, X^{s,x}_r)\, dr + \sum_{j=1}^d \int_s^t \sigma_j(r, X^{s,x}_r)\, dW^j_r, \\ D^{s,x}_t &= \mathrm{I}_{\mathbb{R}^{d\times d}} + \int_s^t (\tfrac{\partial}{\partial x}\mu)(r, X^{s,x}_r)\, D^{s,x}_r\, dr + \sum_{j=1}^d \int_s^t (\tfrac{\partial}{\partial x}\sigma_j)(r, X^{s,x}_r)\, D^{s,x}_r\, dW^j_r.\end{aligned} \tag{7}$$

Note that for every $s \in [0, T]$ we have that the processes $D^{s,x}$, $x \in \mathbb{R}^d$, are in a suitable sense the *derivative processes* of the processes $X^{s,x}$, $x \in \mathbb{R}^d$, with respect to $x \in \mathbb{R}^d$. The function η in (6) allows to include a possible space shift in the PDE. Typically we are interested in the case where η is the identity, that is, for all $x \in \mathbb{R}^d$ it holds that $\eta(x) = x$. Our approximation scheme in (9) below is based on a suitable *fixed-point formulation* of the solution of the PDE (6). To obtain such a fixed-point formulation, we apply the *Feynman–Kac formula* and the *Bismut–Elworthy–Li formula* (see, e.g., Elworthy and Li [29, Theorem 2.1] or Da Prato and Zabczyk [25, Theorem 2.1]). More precisely, let $\mathbf{u}^\infty \in \mathrm{Lip}([0, T] \times \mathbb{R}^d, \mathbb{R}^{1+d})$ satisfy for all $t \in [0, T)$, $x \in \mathbb{R}^d$ that $\mathbf{u}^\infty(t, x) = \big(u(t, x), [\sigma(t, x)]^*(\nabla_x u)(t, x)\big)$ and let $\Phi\colon \mathrm{Lip}([0, T] \times \mathbb{R}^d, \mathbb{R}^{1+d}) \to \mathrm{Lip}([0, T] \times \mathbb{R}^d, \mathbb{R}^{1+d})$ satisfy for all $\mathbf{v} \in \mathrm{Lip}([0, T] \times \mathbb{R}^d, \mathbb{R}^{1+d})$, $s \in [0, T)$, $x \in \mathbb{R}^d$ that

$$\begin{aligned}\big(\Phi(\mathbf{v})\big)(s,x) &= \mathbb{E}\Big[\big(g(X^{s,x}_T) - g(x)\big)\Big(1, \tfrac{[\sigma(s,x)]^*}{T-s}\textstyle\int_s^T \big[\sigma(r, X^{s,x}_r)^{-1} D^{s,x}_r\big]^* dW_r\Big)\Big] + (g(x), 0) \\ &\quad + \int_s^T \mathbb{E}\Big[f\Big(t, X^{s,x}_t, \mathbf{v}\big(t, \eta(X^{s,x}_t)\big)\Big)\Big(1, \tfrac{[\sigma(s,x)]^*}{t-s}\textstyle\int_s^t \big[\sigma(r, X^{s,x}_r)^{-1} D^{s,x}_r\big]^* dW_r\Big)\Big]\, dt\end{aligned} \tag{8}$$

Combining (8) with the *Feynman–Kac formula* and the *Bismut–Elworthy–Li formula* ensures that $\mathbf{u}^\infty = \Phi(\mathbf{u}^\infty)$. This fixed-point equation is well-known in the literature; cf., e.g., Theorem 4.2 in Ma and Zhang [70]. Note that we have incorporated a zero expectation term in (8). The purpose of this term is to slightly reduce the variance when approximating the right-hand side of (8) by Monte Carlo approximations. Now we approximate the non-discrete quantities in (8) (expectation and time integral) by discrete quantities (Monte Carlo averages and quadrature formulas) with different degrees of discretization on different levels (cf. the remarks in Sect. 2.4 below). This yields a family of approximations of Φ. With these approximations of Φ we finally define multilevel Picard approximations of $\mathbf{u}^\infty$ through (5) above. These multilevel Picard approximations are specified in (9) in Sect. 2.3 below.

2.3 The Approximation Scheme

In this subsection we specify multilevel Picard approximations in the case of semilinear PDEs with gradient-dependent nonlinearities and general diffusion processes as forward diffusions (see (9) below). To this end we consider the following setting.

Let $T \in (0,\infty)$, $d \in \mathbb{N}$, $\Theta = \cup_{n\in\mathbb{N}}\mathbb{R}^n$, let $g\colon \mathbb{R}^d \to \mathbb{R}$, $f\colon [0,T]\times\mathbb{R}^d\times\mathbb{R}^{d+1}\to\mathbb{R}$, $\eta\colon \mathbb{R}^d\to\mathbb{R}^d$, $\mu\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}^d$, $\sigma\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}^{d\times d}_{\mathrm{Inv}}$ be measurable functions, let $(q_s^{k,l,\rho})_{k,l\in\mathbb{N}_0,\rho\in(0,\infty),s\in[0,T)} \subseteq \mathcal{Q}_T$, $(m^g_{k,l,\rho})_{k,l\in\mathbb{N}_0,\rho\in(0,\infty)}$, $(m^f_{k,l,\rho})_{k,l\in\mathbb{N}_0,\rho\in(0,\infty)} \subseteq \mathbb{N}$, let $(\Omega,\mathcal{F},\mathbb{P},(\mathbb{F}_t)_{t\in[0,T]})$ be a stochastic basis, let $W^\theta\colon [0,T]\times\Omega\to\mathbb{R}^d$, $\theta\in\Theta$, be independent standard $(\mathbb{F}_t)_{t\in[0,T]}$-Brownian motions with continuous sample paths, for every $l\in\mathbb{Z}$, $\rho\in(0,\infty)$, $\theta\in\Theta$, $x\in\mathbb{R}^d$, $s\in[0,T)$, $t\in[s,T]$ let $\mathcal{X}^{l,\rho,\theta}_{x,s,t}\colon\Omega\to\mathbb{R}^d$, $\mathcal{D}^{l,\rho,\theta}_{x,s,t}\colon\Omega\to\mathbb{R}^{d\times d}$, and $\mathcal{I}^{l,\rho,\theta}_{x,s,t}\colon\Omega\to\mathbb{R}^{1+d}$ be functions, and for every $\theta\in\Theta$, $\rho\in(0,\infty)$ let $\mathbf{U}^\theta_{k,\rho}\colon [0,T]\times\mathbb{R}^d\times\Omega\to\mathbb{R}^{d+1}$, $k\in\mathbb{N}_0$, be functions which satisfy for all $k\in\mathbb{N}$, $s\in[0,T)$, $x\in\mathbb{R}^d$ that

$$\begin{aligned}\mathbf{U}^\theta_{k,\rho}(s,x) &= \sum_{l=0}^{k-1}\sum_{i=1}^{m^g_{k,l,\rho}} \frac{1}{m^g_{k,l,\rho}}\Big[g(\mathcal{X}^{l,\rho,(\theta,l,-i)}_{x,s,T}) - \mathbb{1}_{\mathbb{N}}(l)\, g(\mathcal{X}^{l-1,\rho,(\theta,l,-i)}_{x,s,T})\\ &\quad - \mathbb{1}_{\{0\}}(l)\, g(x)\Big]\,\mathcal{I}^{l,\rho,(\theta,l,-i)}_{x,s,T}\\ &\quad + \big(g(x),0\big) + \sum_{l=0}^{k-1}\sum_{i=1}^{m^f_{k,l,\rho}}\sum_{t\in[s,T]}\frac{q_s^{k,l,\rho}(t)}{m^f_{k,l,\rho}}\\ &\quad \times\Big[f\Big(t,\mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t},\mathbf{U}^{(\theta,l,i,t)}_{l,\rho}\big(t,\eta(\mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t})\big)\Big)\\ &\quad - \mathbb{1}_{\mathbb{N}}(l)\, f\Big(t,\mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t},\mathbf{U}^{(\theta,-l,i,t)}_{[l-1]^+,\rho}\big(t,\eta(\mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t})\big)\Big)\Big]\,\mathcal{I}^{k-l,\rho,(\theta,l,i)}_{x,s,t}.\end{aligned} \tag{9}$$

2.4 Remarks on the Approximation Scheme

In this subsection we add a few comments on the numerical approximations in (9). For this we assume the setting in Sect. 2.3. The set Θ allows to index families of independent random variables which we need for Monte Carlo approximations. The natural numbers $(m^g_{k,l,\rho})_{k,l\in\mathbb{N},\rho\in(0,\infty)}$, $(m^f_{k,l,\rho})_{k,l\in\mathbb{N}_0,\rho\in(0,\infty)} \subseteq \mathbb{N}$ specify the number of Monte Carlo samples in the corresponding levels for approximating the expectations involving g and f appearing on the right-hand side of (8). The family $(q_s^{k,l,\rho})_{k,l\in\mathbb{N}_0,\rho\in(0,\infty),s\in[0,T)} \subseteq \mathcal{Q}_T$ provides the quadrature formulas that we employ to approximate the time integrals $\int_s^T \ldots dt$, $s\in[0,T]$, appearing on the right-hand side of (8). In Sects. 3.1, 3.2 these parameters satisfy that for every $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ it holds that $m^g_{k,l,\rho} = \rho^{k-l}$, $m^f_{k,l,\rho} = \operatorname{round}(\sqrt{\rho}^{k-l})$ and that for every $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ it holds that $q^{k,l,\rho}$ is a Gauß–Legendre quadrature rule with $\operatorname{round}(\Gamma^{-1}(\rho^{(k-l)/2}))$ quadrature nodes. In Sect. 3.3 these parameters satisfy that for every $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ it holds that $m^g_{k,l,\rho} = m^f_{k,l,\rho} = \rho^{k-l}$ and that for every $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ it holds that $q^{k,l,\rho}$ is a Gauß–Legendre quadrature rule with $\operatorname{round}(\Gamma^{-1}(\rho^{(k-l)/2}))$ quadrature nodes. For every $l \in \mathbb{N}$, $\rho \in (0, \infty)$, $\theta \in \Theta$, $s \in [0, T]$, $x \in \mathbb{R}^d$, $v \in (s, T]$ we think of the processes $(\mathcal{X}^{l,\rho,\theta}_{x,s,t})_{t\in[s,T]}$ and $(\mathcal{D}^{l,\rho,\theta}_{x,s,t})_{t\in[s,T]}$ as $(\mathbb{F}_t)_{t\in[s,T]}$-optional measurable computable approximations with $\mathbb{P}\big(\int_s^T \|\sigma(r, \mathcal{X}^{l,\rho,\theta}_{x,s,r})^{-1}\, \mathcal{D}^{l,\rho,\theta)}_{x,s,r}\|^2_{L(\mathbb{R}^d,\mathbb{R}^d)}\, dr < \infty\big) = 1$ (e.g., piecewise constant càdlàg Euler-Maruyama approximations) of the processes $(X^{s,x}_t)_{t\in[s,T]}$ and $(D^{s,x}_t)_{t\in[s,T]}$ in (7) and we think of $\mathcal{I}^{l,\rho,\theta}_{x,s,v}$ as a random variable that satisfies $\mathbb{P}$-a.s. that

$$\mathcal{I}^{l,\rho,\theta}_{x,s,v} = \Big(1, \tfrac{[\sigma(s,x)]^*}{v-s} \textstyle\int_s^v \big[\sigma(r, \mathcal{X}^{l,\rho,\theta}_{x,s,r})^{-1}\, \mathcal{D}^{l,\rho,\theta}_{x,s,r}\big]^* dW^\theta_r\Big). \tag{10}$$

Note that if $\mathcal{X}^{k,\rho,\theta}_{x,s,\cdot}$ and $\mathcal{D}^{k,\rho,\theta}_{x,s,\cdot}$ are piecewise constant then the stochastic integral on the right-hand side of (10) reduces to a stochastic Riemann-type sum which is not difficult to compute. Observe that our approximation scheme (9) employs Picard fixed-point approximations (cf., e.g., [7]), multilevel/multigrid techniques (see, e.g., [19,38,49,50]), discretizations of the SDE system (7), as well as quadrature approximations for the time integrals. Roughly speaking, the numerical approximations in (9) are full history recursive in the sense that for every $(k, \rho) \in \mathbb{N} \times (0, \infty)$ the full history $\mathbf{U}^{(\cdot)}_{0,\rho}, \mathbf{U}^{(\cdot)}_{1,\rho}, \ldots, \mathbf{U}^{(\cdot)}_{k-1,\rho}$ needs to be computed recursively in order to compute $\mathbf{U}^{(\cdot)}_{k,\rho}$. In this sense the numerical approximations in (9) are full history recursive multilevel Picard approximations.

2.5 Special Case: Semilinear Heat Equations

In this subsection we specialize the numerical scheme in (9) to the case of semilinear heat equations.

Proposition 2.1 *Assume the setting in Sect. 2.3, assume for all $k \in \mathbb{N}_0$, $\rho \in (0, \infty)$, $\theta \in \Theta$, $x \in \mathbb{R}^d$, $s \in [0, T)$, $t \in [s, T]$, $u \in (s, T]$ that $\eta(x) = x$, $\mathcal{X}^{k,\rho,\theta}_{x,s,t} = x + W^\theta_t - W^\theta_s$, $\mathcal{D}^{k,\rho,\theta}_{x,s,t} = \sigma(s, x) = \mathrm{I}_{\mathbb{R}^{d\times d}}$, $\mathcal{I}^{k,\rho,\theta}_{x,s,s} = 0$, $\mathcal{I}^{k,\rho,\theta}_{x,s,u} = (1, \frac{W^\theta_u - W^\theta_s}{u-s})$, and for every $\theta \in \Theta$, $a \in [0, T]$, $b \in [a, T]$ let $\Delta W^\theta_{a,b} \colon \Omega \to \mathbb{R}^d$ be the function given by $\Delta W^\theta_{a,b} = W^\theta_b - W^\theta_a$. Then it holds for all $\theta \in \Theta$, $k \in \mathbb{N}$, $\rho \in (0, \infty)$, $s \in [0, T)$, $x \in \mathbb{R}^d$ that*

$$\begin{aligned}
\mathbf{U}^\theta_{k,\rho}(s, x) = \big(g(x), 0\big) &+ \sum_{i=1}^{m^g_{k,0,\rho}} \frac{1}{m^g_{k,0,\rho}} \Big[g\big(x + \Delta W^{(\theta,0,-i)}_{s,T}\big) - g(x)\Big]\Big(1, \tfrac{1}{T-s} \Delta W^{(\theta,0,-i)}_{s,T}\Big) \\
&+ \sum_{l=0}^{k-1} \sum_{i=1}^{m^f_{k,l,\rho}} \sum_{t\in(s,T]} \frac{q^{k,l,\rho}_s(t)}{m^f_{k,l,\rho}} \Big[f\big(t, x + \Delta W^{(\theta,l,i)}_{s,t}, \mathbf{U}^{(\theta,l,i,t)}_{l,\rho}(t, x + \Delta W^{(\theta,l,i)}_{s,t})\big) \\
&- \mathbb{1}_{\mathbb{N}}(l) f\big(t, x + \Delta W^{(\theta,l,i)}_{s,t}, \mathbf{U}^{(\theta,-l,i,t)}_{[l-1]^+,\rho}(t, x + \Delta W^{(\theta,l,i)}_{s,t})\big)\Big]\big(1, \tfrac{1}{t-s} \Delta W^{(\theta,l,i)}_{s,t}\big).
\end{aligned} \tag{11}$$

The proof of Proposition 2.1 is clear and therefore omitted.

2.6 Special Case: Geometric Brownian Motion

In this subsection we specialize the numerical scheme in (9) to the special case where the forward diffusion is a geometric Brownian motion. Such PDEs often appear in the financial engineering literature.

Proposition 2.2 *Assume the setting in Sect. 2.3, let $\bar{\mu} \in \mathbb{R}$, $\bar{\sigma} \in (0, \infty)$, for every $\theta \in \Theta$, $a \in [0, T]$, $b \in [a, T]$ let $\Delta W^{\theta}_{a,b} : \Omega \to \mathbb{R}^d$ be the function given by $\Delta W^{\theta}_{a,b} = W^{\theta}_b - W^{\theta}_a$, and assume for all $k \in \mathbb{N}_0$, $\rho \in (0, \infty)$, $\theta \in \Theta$, $x \in (0, \infty)^d$, $s \in [0, T)$, $t \in [s, T]$, $u \in (s, T]$ that $\eta(x) = x$, $\mathcal{D}^{k,\rho,\theta}_{x,s,t} = \exp((\bar{\mu} - \frac{\bar{\sigma}^2}{2})(t - s)) \exp(\bar{\sigma} \operatorname{diag}(\Delta W^{\theta}_{s,t}))$, $\mathcal{X}^{k,\rho,\theta}_{x,s,t} = \mathcal{D}^{k,\rho,\theta}_{x,s,t} x$, $\sigma(s, x) = \bar{\sigma} \operatorname{diag}(x)$, $\mathcal{I}^{k,\rho,\theta}_{x,s,s} = 0$, $\mathcal{I}^{k,\rho,\theta}_{x,s,u} = (1, \frac{1}{u-s} \Delta W^{\theta}_{s,u})$. Then it holds for all $\theta \in \Theta$, $k \in \mathbb{N}$, $\rho \in (0, \infty)$, $s \in [0, T)$, $x \in (0, \infty)^d$ that*

$$\begin{aligned}
\mathbf{U}^{\theta}_{k,\rho}(s, x) &= \big(g(x), 0\big) + \sum_{i=1}^{m^g_{k,0,\rho}} \frac{1}{m^g_{k,0,\rho}} \Big[g\big(\mathcal{X}^{0,\rho,(\theta,0,-i)}_{x,s,T}\big) - g(x)\Big]\Big(1, \tfrac{1}{T-s} \Delta W^{(\theta,0,-i)}_{s,T}\Big) \\
&\quad + \sum_{l=0}^{k-1} \sum_{i=1}^{m^f_{k,l,\rho}} \sum_{t \in (s,T]} \frac{q^{k,l,\rho}_s(t)}{m^f_{k,l,\rho}} \Big[f\big(t, \mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t}, \mathbf{U}^{(\theta,l,i,t)}_{l,\rho}(t, \mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t})\big) \\
&\quad - \mathbb{1}_{\mathbb{N}}(l) f\big(t, \mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t}, \mathbf{U}^{(\theta,-l,i,t)}_{[l-1]^+,\rho}(t, \mathcal{X}^{k-l,\rho,(\theta,l,i)}_{x,s,t})\big)\Big]\big(1, \tfrac{1}{t-s} \Delta W^{(\theta,l,i)}_{s,t}\big).
\end{aligned} \tag{12}$$

The proof of Proposition 2.2 is clear and therefore omitted. In the setting of Proposition 2.2 we note that for all $k \in \mathbb{N}$, $\rho \in (0, \infty)$, $\theta \in \Theta$, $x \in (0, \infty)^d$, $s \in [0, T)$, $t \in (s, T]$ it holds $\mathbb{P}$-a.s. that

$$\begin{aligned}
\mathcal{X}^{k,\rho,\theta}_{x,s,t} &= x + \int_s^t \bar{\mu} \mathcal{X}^{k,\rho,\theta}_{x,s,r} \, dr + \int_s^t \bar{\sigma} \operatorname{diag}(\mathcal{X}^{k,\rho,\theta}_{x,s,r}) \, dW^{\theta}_r, \\
\mathcal{D}^{k,\rho,\theta}_{x,s,t} &= \mathrm{I}_{\mathbb{R}^{d \times d}} + \int_s^t \bar{\mu} \mathcal{D}^{k,\rho,\theta}_{x,s,r} \, dr + \int_s^t \bar{\sigma} \operatorname{diag}(\mathcal{D}^{k,\rho,\theta}_{x,s,r}) \, dW^{\theta}_r, \\
\mathcal{I}^{k,\rho,\theta}_{x,s,t} &= \bigg(1, \tfrac{[\sigma(s,x)]^*}{t-s} \int_s^t \big[\sigma(r, \mathcal{X}^{k,\rho,\theta}_{x,s,r})^{-1} \mathcal{D}^{k,\rho,\theta}_{x,s,r}\big]^* dW^{\theta}_r\bigg).
\end{aligned} \tag{13}$$

3 Numerical Simulations of High-Dimensional Nonlinear PDEs

In this section we apply the algorithm in (9) to approximate the solutions of several nonlinear PDEs; see Sects. 3.1–3.3 below. The solutions of the PDEs in Sects. 3.1–3.3 are not known explicitly. In Sects. 3.1–3.3 the algorithm is tested for a one-dimensional and a one hundred-dimensional version of a PDE. In the one-dimensional cases in Sects. 3.1–3.3 we present the error of our algorithm relative to a high-precision approximation of the exact solution of the PDE provided by a finite difference approximation scheme (see the left-hand sides of Figs. 1, 2, 3, and 4 and Tables 2, 4, and 6 below). In the one hundred-dimensional cases in Sects. 3.1–3.3 we present the approximation increments of our scheme to analyze the performance of our scheme in the case of high-dimensional PDEs (see the right-hand sides of Figs. 1, 2, 3, and 4 and Tables 3, 5, and 7 below). Moreover, for each of the PDEs in Sects. 3.1–3.3 we illustrate the growth of the computational effort with respect to the dimension by running the algorithm for each PDE for every dimension $d \in \{5, 6, \ldots, 100\}$ and recording the associated runtimes (see Fig. 5). All simulations are performed with MATLAB on a 2.8 GHz Intel i7 processor with 16 GB RAM. All MATLAB codes are provided in the "Appendix".

Throughout this section assume the setting in Sect. 2.3, let $x_0 \in \mathbb{R}^d$, let $u \in C^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R})$ be a function which satisfies for all $t \in [0,T)$, $x \in \mathbb{R}^d$ that $u(T,x) = g(x)$ and

$$(\tfrac{\partial}{\partial t}u)(t,x) + f\big(t,x,u(t,\eta(x)),[\sigma(t,\eta(x))]^*(\nabla_x u)(t,\eta(x))\big) + \langle \mu(t,x), (\nabla_x u)(t,x)\rangle \\ + \tfrac{1}{2}\operatorname{Trace}\big(\sigma(t,x)[\sigma(t,x)]^*(\operatorname{Hess}_x u)(t,x)\big) = 0, \tag{14}$$

and assume for all $\theta \in \Theta$, $\rho \in (0,\infty)$, $s \in [0,T)$, $x \in \mathbb{R}^d$ that

$$\mathbf{U}^{\theta}_{0,\rho}(s,x) = \big(g(x),0\big) + \sum_{i=1}^{m^g_{0,0,\rho}} \frac{1}{m^g_{0,0,\rho}} \big[g(\mathcal{X}^{0,\rho,(\theta,0,-i)}_{x,s,T}) - g(x)\big]\mathcal{I}^{0,\rho,(\theta,0,-i)}_{x,s,T}. \tag{15}$$

To obtain smoother results we average over 10 independent simulation runs. More precisely, for the numerical results in Sects. 3.1–3.3, for every $d \in \{1,100\}$ we produce one realization of

$$\{1,2,\ldots,\rho_{\max}\} \times \{1,2,\ldots,10\} \ni (\rho,i) \mapsto \mathbf{U}^i_{\rho,\rho}(0,x_0) \\ = (\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0), \mathbf{U}^{i,[2]}_{\rho,\rho}(0,x_0), \ldots, \mathbf{U}^{i,[d+1]}_{\rho,\rho}(0,x_0)) \in \mathbb{R}^{d+1}, \tag{16}$$

where $\rho_{\max} = 7$ in Sects. 3.1, 3.2 and where $\rho_{\max} = 5$ in Sect. 3.3.

Figures 1–3 illustrate the empirical convergence of our scheme. In Figs. 1–3 the left-hand side depicts for the settings of Sects. 3.1–3.3 in the one-dimensional case the relative approximation errors

$$\frac{\frac{1}{10}\sum_{i=1}^{10} |\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0) - \mathrm{v}|}{|\mathrm{v}|} \tag{17}$$

against the average runtime needed to compute the realizations $(\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0))_{i\in\{1,2,\ldots,10\}}$ for $\rho \in \{1,2,\ldots,\rho_{\max}\}$, where $\mathrm{v} \in \mathbb{R}$ is the approximation obtained through the finite difference approximation scheme. The right-hand side of the Figs. 1–3 depicts for the settings of Sects. 3.1–3.3 in the one hundred-dimensional case the relative approximation increments

$$\frac{\frac{1}{10}\sum_{i=1}^{10} |\mathbf{U}^{i,[1]}_{\rho+1,\rho+1}(0,x_0) - \mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)|}{\frac{1}{10}|\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho_{\max},\rho_{\max}}(0,x_0)|} \tag{18}$$

against the average runtime needed to compute the realizations $(\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0))_{i\in\{1,2,\ldots,10\}}$ for $\rho \in \{1,2,\ldots,\rho_{\max}-1\}$.
Tables 2–7 present several statistics for the simulations.

Figure 5 shows the growth of the runtime of our algorithm with respect to the dimension for each of the example PDEs. More precisely, the panels on the left and in the middle of Fig. 5 show for the settings in Sects. 3.1, 3.2 the runtime needed to compute one realization of $\mathbf{U}^1_{6,6}(0,x_0)$ against the dimension $d \in \{5,6,\ldots,100\}$. The panel on the right of Fig. 5 shows for the setting in Sect. 3.3 the average runtime needed to compute 20 realizations of $\mathbf{U}^1_{4,4}(0,x_0)$ against the dimension $d \in \{5,6,\ldots,100\}$. We average over 20 runs here to obtain smoother results.

We remark that the theoretical results in [27] do not apply to the example PDEs of Sects. 3.1–3.3 since these PDEs, besides other constraints, do not have both a globally Lipschitz continuous nonlinearity and a terminal condition with a bounded derivative. Under these and further regularity assumptions, [27, Corollary 3.14] proves that there exists a constant $C \in (0,\infty)$ such that for all $\rho \in \mathbb{N}$ it holds that

$$\sup_{t\in[0,T],x\in\mathbb{R}^d} \|\mathbf{U}^{0,[1]}_{\rho,\rho}(t,x) - u(t,x)\|_{L^2(\mathbb{P};\mathbb{R})} \le C\left[\frac{(1+2L)e^T}{\sqrt{\rho}}\right]^{\rho}, \tag{19}$$

where $L \in [0,\infty)$ denotes the Lipschitz constant of the nonlinearity f.

3.1 Pricing with Counterparty Credit Risk

In this subsection we present a numerical simulation of a semilinear PDE that arises in the valuation of derivative contracts with counterparty credit risk. The PDE is a special case of the PDEs that are, e.g., derived in Henry-Labordère [53] and Burgard and Kjaer [13].

Throughout this subsection assume the setting in the beginning of Sect. 3, let $\bar{\sigma} = 0.2$, $\beta = 0.03$, $K_1 \in \mathbb{R}$, $K_2 \in (K_1,\infty)$, and assume for all $s \in [0,T]$, $t \in [s,T]$, $x = (x_1,\dots,x_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^d$, $k \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $\theta \in \Theta$ that $\eta(x) = x$, $\mu(s,x) = 0$, $\sigma(s,x) = \bar{\sigma}\operatorname{diag}(x)$, $x_0 = (100,100,\dots,100) \in \mathbb{R}^d$, $\mathcal{D}^{k,\rho,\theta}_{x,s,t} = \exp(-\frac{\bar{\sigma}^2}{2}(t-s))\exp\big(\bar{\sigma}\operatorname{diag}(\Delta W^{\theta}_{s,t})\big)$, $\mathcal{X}^{k,\rho,\theta}_{x,s,t} = \mathcal{D}^{k,\rho,\theta}_{x,s,t}\,x$, $f(s,x,y,z) = \beta([y]^+ - y)$, and

$$g(x) = \left[\min_{j\in\{1,2,\dots,d\}} x_j - K_1\right]^+ - \left[\min_{j\in\{1,2,\dots,d\}} x_j - K_2\right]^+ - \frac{K_2-K_1}{2}. \tag{20}$$

Note that the solution u of the PDE (14) satisfies for all $t \in [0,T)$, $x = (x_1,x_2,\dots,x_d) \in \mathbb{R}^d$ that $u(T,x) = \min\{\max\{\min_{j\in\{1,2,\dots,d\}} x_j, K_1\}, K_2\} - \frac{K_1+K_2}{2}$ and

$$(\tfrac{\partial}{\partial t}u)(t,x) - \beta\min\{u(t,x),0\} + \frac{\bar{\sigma}^2}{2}\sum_{i=1}^{d}|x_i|^2\big(\tfrac{\partial^2}{\partial x_i^2}u\big)(t,x) = 0. \tag{21}$$

We note that (21) is not the standard PDE associated with counterparty credit risk but a transformed version hereof; cf., e.g., (3) and (5) in [53]. In (21) the function u models the price of an European financial derivative with possibly negative payoff g at maturity T whose buyer may default. The real number $-e^{r(T-t)}u(t,x) \in \mathbb{R}$ describes the value of the financial derivative at time $t \in [0,T]$ in dependence on the prices $x = (x_1,\dots,x_d) \in \mathbb{R}^d$ of the $d \in \mathbb{N}$ underlyings of the model given that no default has occurred before time t where r is the interest rate. In the model the option payoff depends in the case of default on the price of the derivative itself. The choice of the parameters is based on the choice of the parameters in Henry-Labordère [53, Sect. 5.3]. As in [53, Sect. 5.3] we approximate the solution of (21) for different time horizons $T \in \{2,4,6,8,10\}$. We choose the parameters $K_1 \in \mathbb{R}$ and $K_2 \in (K_1,\infty)$ in dependence on T and d as follows. For $d = 1$ and $T = 2$ we choose $K_1 = 90$ and $K_2 = 110$ as in [53]. In this case it holds for all $k \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ that

$$\mathbb{E}\left[\min_{j\in\{1,\dots,d\}} \mathcal{X}^{k,\rho,0}_{x_0,0,T}(j)\right] = \frac{K_1+K_2}{2} \quad \text{and}$$
$$\mathbb{P}\left(\min_{j\in\{1,\dots,d\}} \mathcal{X}^{k,\rho,0}_{x_0,0,T}(j) \in [K_1,K_2]\right) \approx 0.27. \tag{22}$$

In particular, these properties ensure that for all $k \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ the random variable $g(\mathcal{X}^{k,\rho,0}_{x_0,0,T})$ is not a linear function of $\mathcal{X}^{k,\rho,0}_{x_0,0,T}$. In all other cases $d \in \{1,100\}$ and $T \in \{2,4,6,8,10\}$ we choose $K_1 \in \mathbb{R}$ and $K_2 \in (K_1,\infty)$ in a way so that (22) holds. The values of $K_1 \in \mathbb{R}$ and $K_2 \in (K_1,\infty)$ are presented in Table 1.

Table 1 Choice of the parameters $K_1 \in \mathbb{R}$ and $K_2 \in (K_1, \infty)$ in dependence on T and d for the pricing with counterparty credit risk example in Sect. 3.1

	$d = 1$					$d = 100$				
T	2	4	6	8	10	2	4	6	8	10
K_1	90	86.0719	82.8696	80.1389	77.7074	45.6789	32.3809	24.6731	19.5154	15.8084
K_2	110	113.9280	117.1303	119.8610	122.2925	49.5468	36.3024	28.3657	22.9144	18.9086

The simulation results for the case $d = 1$ are presented in Table 2 and on the left-hand side of Fig. 1. The simulation results for the case $d = 100$ are presented in Table 3 and on the right-hand side of Fig. 1.

Figure 1 suggests for every $d \in \{1, 100\}$ and every $T \in \{2, 4, 6, 8, 10\}$ an empirical convergence rate close to $1/3$. Moreover, Fig. 1 indicates that the relative approximation errors (in the case $d = 1$) and the relative approximation increments (in the case $d = 100$) increase as the time horizon T increases. We presume that this effect can be explained by a higher variance of the random variables $\mathbf{U}^{\theta}_{k,\rho}(s, x)$ for $\theta \in \Theta, k \in \mathbb{N}, \rho \in (0, \infty), s \in [0, T)$, $x \in \mathbb{R}^d$ as the time horizon T increases.

3.2 Pricing with Different Interest Rates for Borrowing and Lending

We consider a pricing problem of an European option in a financial market with different interest rates for borrowing and lending. The model goes back to Bergman [9] and serves as a standard example in the literature on numerical methods for BSDEs (see, e.g., [7,8,12,22,41]).

Throughout this subsection assume the setting in the beginning of Sect. 3, let $\bar{\mu} = 0.06$, $\bar{\sigma} = 0.2$, $R^l = 0.04$, $R^b \in (R^l, \infty)$, and assume for all $s \in [0, T]$, $t \in [s, T]$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z = (z_1, \ldots, z_d) \in \mathbb{R}^d$, $k \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $\theta \in \Theta$ that $T = 0.5$, $\eta(x) = x$, $\mu(s, x) = \bar{\mu}x$, $\sigma(s, x) = \bar{\sigma}\,\mathrm{diag}(x)$, $x_0 = (100, 100, \ldots, 100) \in \mathbb{R}^d$, $\mathcal{D}^{k,\rho,\theta}_{x,s,t} = \exp((\bar{\mu} - \frac{\bar{\sigma}^2}{2})(t - s)) \exp\big(\bar{\sigma}\,\mathrm{diag}(\Delta W^{\theta}_{s,t})\big)$, $\mathcal{X}^{k,\rho,\theta}_{x,s,t} = \mathcal{D}^{k,\rho,\theta}_{x,s,t}\, x$, and

$$f(s, x, y, z) = -R^l y - \frac{(\bar{\mu} - R^l)}{\bar{\sigma}} \left(\sum_{i=1}^{d} z_i\right) + (R^b - R^l) \left[\frac{1}{\bar{\sigma}} \left(\sum_{i=1}^{d} z_i\right) - y\right]^+. \tag{23}$$

Note that the solution u of the PDE (14) 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

$$\begin{aligned}&(\tfrac{\partial}{\partial t}u)(t, x) + \tfrac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} |x_i|^2 \big(\tfrac{\partial^2}{\partial x_i^2}u\big)(t, x)\\ &\quad - \min\left\{R^b\left(u(t, x) - \sum_{i=1}^{d} x_i \big(\tfrac{\partial}{\partial x_i}u\big)(t, x)\right), R^l\left(u(t, x) - \sum_{i=1}^{d} x_i \big(\tfrac{\partial}{\partial x_i}u\big)(t, x)\right)\right\} = 0.\end{aligned} \tag{24}$$

In (24) the function u models the price of an European financial derivative with payoff g at maturity T in a financial market with a higher interest rate for borrowing than for lending. The number $u(t, x) \in \mathbb{R}$ describes the price of the financial derivative at time $t \in [0, T]$ in dependence on the prices $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ of the d underlyings of the model.

In the case $d = 1$ we assume that for all $x \in \mathbb{R}$ it holds that $g(x) = [x - 100]^+$. This setting agrees with the setting in Gobet et al. [41, Sect. 6.3.1], where it is also noted that the PDE (24) is actually linear. More precisely, in this case we have that u also satisfies (14) with $f\colon [0, T] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ being replaced by $\bar{f}\colon [0, T] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ satisfying for all $t \in [0, T]$, $x, y, z \in \mathbb{R}$ that $\bar{f}(t, x, y, z) = -R^b y - \frac{(\bar{\mu} - R^b)}{\bar{\sigma}} z$.

In the case $d = 100$ we assume that for all $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ it holds that $g(x) = [\max_{i \in \{1,\ldots,100\}} x_i - 120]^+ - 2[\max_{i \in \{1,\ldots,100\}} x_i - 150]^+$. This choice of g is based on the choice of g in Bender et al. [8, Sect. 4.2]. We note that with this choice of g the solution u of the PDE (24) does not solve the PDE (14) with $f\colon [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ replaced

Table 2 Average runtime, empirical mean, empirical standard deviation, and relative approximation error in the case $d = 1$ and $T = 2$ for the pricing with counterparty credit risk example in Sect. 3.1

ρ	1	2	3	4	5	6	7
Average runtime in seconds	0.002	0.008	0.093	0.968	11.430	155.607	6226.101
$\overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0) = \frac{1}{10}\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0)$	− 0.582	− 3.614	− 0.767	− 0.433	− 0.916	− 0.866	− 0.884
$\sqrt{\frac{1}{9}\sum_{i=1}^{10} \lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0) - \overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0)\rvert^2}$	9.346	3.964	1.279	0.782	0.105	0.067	0.015
$\frac{1}{10}\sum_{i=1}^{10} \frac{\lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)-\mathrm{v}\rvert}{\lvert\mathrm{v}\rvert}$	9.8923	4.1417	1.1794	0.7941	0.0943	0.0617	0.0135

The approximation by the finite difference approximation scheme in MATLAB code 7 yields `v=approximateUfinitediffgbm(0,x0,2^11)` ≈ -0.883

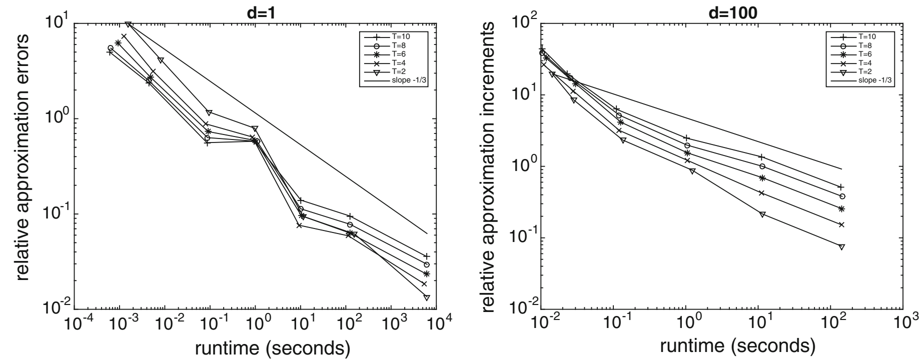


Fig. 1 Empirical convergence of the scheme in (12) for the pricing with counterparty credit risk example in Sect. 3.1. Left: Relative approximation errors $\frac{1}{10|\mathrm{v}|}\sum_{i=1}^{10}|\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)-\mathrm{v}|$ against the average runtime for $\rho \in \{1,2,\ldots,7\}$ and $T \in \{2,4,6,8,10\}$ in the case $d = 1$. Right: Relative approximation increments $\left(\frac{1}{10}\sum_{i=1}^{10}|\mathbf{U}_{\rho+1,\rho+1}^{i,[1]}(0,x_0)-\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)|\right)/\left(\frac{1}{10}|\sum_{i=1}^{10}\mathbf{U}_{7,7}^{i,[1]}(0,x_0)|\right)$ against the average runtime for $\rho \in \{1,2,\ldots,6\}$ and $T \in \{2,4,6,8,10\}$ in the case $d = 100$

by $\bar{f}\colon [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d \to \mathbb{R}$ satisfying for all $t \in [0,T]$, $y \in \mathbb{R}$, $x, z \in \mathbb{R}^d$ that $\bar{f}(t,x,y,z) = -R^b y - \frac{(\bar{\mu}-R^b)}{\bar{\sigma}}\sum_{i=1}^d z_i$.

For $d \in \{1, 100\}$ and $R^b \in \{0.06, 0.07, 0.09, 0.1, 0.12, 0.15\}$ we approximate the solution of (24). The case $d = 1$, $R^b = 0.06$ agrees with the choice in Gobet et al. [41, Sect. 6.3.1].

The simulation results for the case $d = 1$ are presented in Table 4 and on the left-hand side of Fig. 2. The left-hand side of Fig. 2 suggests in the case $d = 1$ for every $R^b \in \{0.06, 0.07, 0.09, 0.1, 0.12, 0.15\}$ an empirical convergence rate close to $1/4$. Moreover, the left-hand side of Fig. 2 indicates that as R^b increases the relative approximation errors increase. This observation is in accordance with the theoretical results. Indeed, note that as R^b increases the Lipschitz constant L of the nonlinearity f increases (see (23)). Moreover, the theoretical results from [27] - although not applicable here - indicate that the approximation error of our scheme grows as L increases (see (19)).

The simulation results for the case $d = 100$ are presented in Table 5 and on the right-hand side of Fig. 2. The right-hand side of Fig. 2 suggests in the case $d = 100$ for every $R^b \in \{0.06, 0.07\}$ an empirical convergence rate close to $1/4$. However, in the case $d = 100$ for every $R^b \in \{0.09, 0.1, 0.12, 0.15\}$ the right-hand side of Fig. 2 does not indicate whether the scheme converges. Although the theoretical results of [27] are not applicable in the case of the PDE (24), we suspect that an error estimate similar to (19) also holds in the case of the PDE (24). Larger values of R^b lead to a larger Lipschitz constant L of the nonlinearity f. For every $R^b \in \{0.09, 0.1, 0.12, 0.15\}$ we suspect that the Lipschitz constant L is so large so that convergence of the scheme only becomes apparent for values of ρ larger than 7 (observe, e.g., that $\left(\frac{0.5}{\sqrt{7}}\right)^7 < 10^{-5}$ whereas $\left(\frac{4}{\sqrt{7}}\right)^7 > 18$). We believe that this effect only shows up in the case $d = 100$ since the nonlinearity f is gradient-dependent. Indeed, (23) shows that the Lipschitz constant L of f depends on the dimension d. In particular, observe that for every $d \in \mathbb{N}$ it holds that $L = \sqrt{d}$ is the minimal Lipschitz constant for the function $\mathbb{R}^d \ni (z_1,\ldots,z_d) \mapsto (\sum_{i=1}^d z_i) \in \mathbb{R}$ with respect to the Euclidean norm on $\mathbb{R}^d$ and the absolute value on $\mathbb{R}$. In the case $d = 1$ the minimal Lipschitz constant L might thus even in the case $R^b \in \{0.09, 0.1, 0.12, 0.15\}$ be small enough so that convergence already becomes apparent for $\rho \in \{1,2,\ldots,7\}$, whereas the minimal Lipschitz constant L might be too large to see convergence in the case $d = 100$.

Table 3 Average runtime, empirical mean, empirical standard deviation, and relative approximation increments in the case $d = 100$ and $T = 2$ for the pricing with counterparty credit risk example in Sect. 3.1

ρ	1	2	3	4	5	6	7
Average runtime in seconds	0.014	0.029	0.135	1.228	11.600	141.612	6946.315
$\overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0,x_0) = \frac{1}{10}\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)$	0.726	0.093	− 0.019	0.041	0.072	0.080	0.084
$\sqrt{\frac{1}{9}\sum_{i=1}^{10} \lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0) - \overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0,x_0)\rvert^2}$	1.808	0.910	0.327	0.091	0.021	0.008	0.004
$\frac{1}{10}\sum_{i=1}^{10} \frac{\lvert\mathbf{U}^{i,[1]}_{\rho+1,\rho+1}(0,x_0) - \mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)\rvert}{\lvert\overline{\mathbf{U}}^{[1]}_{7,7}(0,x_0)\rvert}$	19.6514	8.4584	2.3320	0.8836	0.2142	0.0766	

Table 4 Average runtime, empirical mean, empirical standard deviation, and relative approximation error in the case $d = 1$ and $R^b = 0.06$ for the pricing with different interest rates example in Sect. 3.2

ρ	1	2	3	4	5	6	7
Average runtime in seconds	0.001	0.011	0.114	0.990	11.347	132.484	6264.475
$\overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0) = \frac{1}{10}\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0)$	5.695	5.947	7.085	7.631	7.156	7.162	7.166
$\sqrt{\frac{1}{9}\sum_{i=1}^{10} \lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0) - \overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0)\rvert^2}$	7.780	4.080	1.612	0.811	0.151	0.071	0.016
$\frac{1}{10}\sum_{i=1}^{10} \frac{\lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)-\mathrm{v}\rvert}{\lvert\mathrm{v}\rvert}$	0.8285	0.4417	0.1777	0.1047	0.0170	0.0086	0.0019

The approximation by the finite difference approximation scheme in MATLAB code 7 yields `v=approximateUfinitediffgbm(0,x0,2^11)` ≈ 7.156

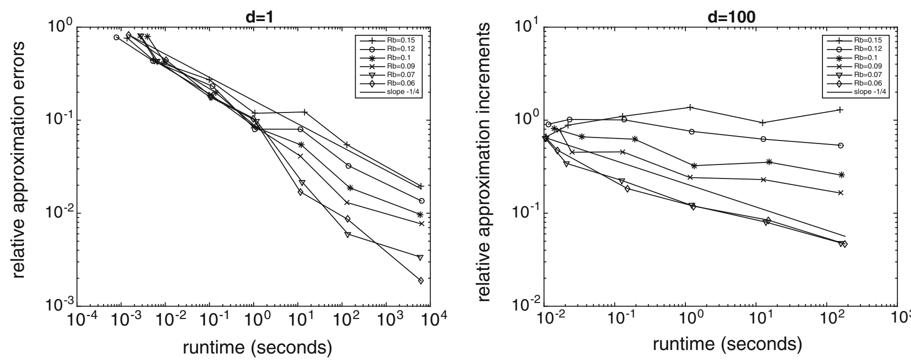


Fig. 2 Empirical convergence of the scheme (12) for the pricing with different interest rates example in Sect. 3.2. Left: Relative approximation errors $\frac{1}{10|\mathrm{v}|}\sum_{i=1}^{10}|\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)-\mathrm{v}|$ against the average runtime for $\rho \in \{1,2,\ldots,7\}$ and $R^b \in \{0.06, 0.07, 0.09, 0.1, 0.12, 0.15\}$ in the case $d=1$. Right: Relative approximation increments $\left(\frac{1}{10}\sum_{i=1}^{10}|\mathbf{U}^{i,[1]}_{\rho+1,\rho+1}(0,x_0)-\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)|\right)\big/\left(\frac{1}{10}|\sum_{i=1}^{10}\mathbf{U}^{i,[1]}_{7,7}(0,x_0)|\right)$ against the average runtime for $\rho \in \{1,2,\ldots,6\}$ and $R^b \in \{0.06, 0.07, 0.09, 0.1, 0.12, 0.15\}$ in the case $d=100$

3.3 Allen–Cahn Equation

In this subsection we consider the Allen–Cahn equation with a double well potential.

Throughout this subsection assume the setting in the beginning of Sect. 3 and assume for all $s \in [0,T]$, $t \in [s,T]$, $x=(x_1,\ldots,x_d) \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^d$, $k \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $\theta \in \Theta$ that $T=1$, $\eta(x)=x$, $\mu(s,x)=0$, $\sigma(s,x)=\mathrm{I}_{\mathbb{R}^{d\times d}}$, $x_0=(0,0,\ldots,0)\in\mathbb{R}^d$, $\mathcal{X}^{k,\rho,\theta}_{x,s,t}=x+W^\theta_t-W^\theta_s$, $\mathcal{D}^{k,\rho,\theta}_{x,s,t}=\mathrm{I}_{\mathbb{R}^{d\times d}}$, $f(s,x,y,z)=y-y^3$, $C\in\mathbb{R}$, and $g(x)=\frac{C}{1+\max\{|x_1|^2,\ldots,|x_d|^2\}}$. Note that the solution u of the PDE (14) satisfies for all $t\in[0,T)$, $x\in\mathbb{R}^d$ that $u(T,x)=\frac{C}{1+\max\{|x_1|^2,\ldots,|x_d|^2\}}$ and

$$(\tfrac{\partial}{\partial t}u)(t,x)+u(t,x)-\big[u(t,x)\big]^3+\tfrac{1}{2}\big(\Delta_x u\big)(t,x)=0. \tag{25}$$

We approximate the solution of (25) for different values of the constant C in the terminal condition. In the case $d=1$ we choose $C\in\{0.018, 0.18, 1, 1.8, 2.7\}$ and in the case $d=100$ we choose $C\in\{0.1, 1, 5.5, 10, 15\}$. Note that for every 100-dimensional standard Brownian motion $\mathcal{W}=(\mathcal{W}^1,\ldots,\mathcal{W}^{100})\colon[0,T]\times\Omega\to\mathbb{R}^{100}$ it holds that

$$\mathbb{E}\left[\frac{1}{1+\max_{j\in\{1,\ldots,100\}}(\mathcal{W}^j_T)^2}\right]\approx 0.18\cdot\mathbb{E}\left[\frac{1}{1+(\mathcal{W}^1_T)^2}\right]. \tag{26}$$

To ensure that the expected terminal values $\mathbb{E}[g(\mathcal{W}_T)]$ are approximately of the same size in the cases $d=1$ and $d=100$, we choose $C\in\{0.018, 0.18, 1, 1.8, 2.7\}$ in the case $d=1$ and $C\in\{0.1, 1, 5.5, 10, 15\}$ in the case $d=100$.

The simulation results in the case $d=1$ are presented in Table 6 and on the left-hand sides of Figs. 3 and 4. The simulation results for the case $d=100$ are presented in Table 7 and on the right-hand sides of Figs. 3 and 4. Figure 3 suggests in the case $d=1$, $C\in\{0.018, 0.18, 1\}$ and in the case $d=100$, $C\in\{0.1, 1, 5.5, 10\}$ an empirical convergence rate of $1/4$. In the case $d=1$, $C=1.8$ there seems to be convergence but an empirical convergence rate remains unclear. Figure 4 suggests that the algorithm diverges in the case $d=1$, $C=2.7$ and in the case $d=100$, $C=15$. Our explanation for this is a numerical instability due to the time discretization and due to the superlinearly growing nonlinearity. More precisely,

Table 5 Average runtime, empirical mean, empirical standard deviation, and relative approximation increments in the case $d = 100$ and $R^b = 0.06$ for the pricing with different interest rates example in Sect. 3.2

ρ	1	2	3	4	5	6	7
Average runtime in seconds	0.011	0.015	0.151	1.317	14.813	181.647	8825.390
$\overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0) = \frac{1}{10}\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0)$	28.902	22.854	23.356	21.771	21.374	21.274	21.299
$\sqrt{\frac{1}{9}\sum_{i=1}^{10} \lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0, x_0) - \overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0, x_0)\rvert^2}$	8.798	11.317	4.492	2.953	1.449	1.376	0.467
$\frac{1}{10}\sum_{i=1}^{10} \frac{\left\lvert\mathbf{U}^{i,[1]}_{\rho+1,\rho+1}(0,x_0)-\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)\right\rvert}{\left\lvert\overline{\mathbf{U}}^{[1]}_{7,7}(0,x_0)\right\rvert}$	0.6446	0.4770	0.1840	0.1190	0.0844	0.0467	

Table 6 Average runtime, empirical mean, empirical standard deviation, and relative approximation error in the case $d = 1$ and $C = 1$ for the Allen–Cahn equation in Sect. 3.3

ρ	1	2	3	4	5
Average runtime in seconds	0.005	0.035	0.237	7.402	345.124
$\overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0,x_0) = \frac{1}{10}\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)$	1.027	0.866	0.918	0.894	0.897
$\sqrt{\frac{1}{9}\sum_{i=1}^{10} \lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0) - \overline{\mathbf{U}}^{[1]}_{\rho,\rho}(0,x_0)\rvert^2}$	0.219	0.131	0.078	0.037	0.013
$\frac{1}{10}\sum_{i=1}^{10} \frac{\lvert\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)-\mathrm{v}\rvert}{\lvert\mathrm{v}\rvert}$	0.2462	0.1197	0.0691	0.0302	0.0124

The approximation by the finite difference approximation scheme in MATLAB code 8 yields `v=approximateUfinitediffgbm(0,x0,2^11)` ≈ 0.905

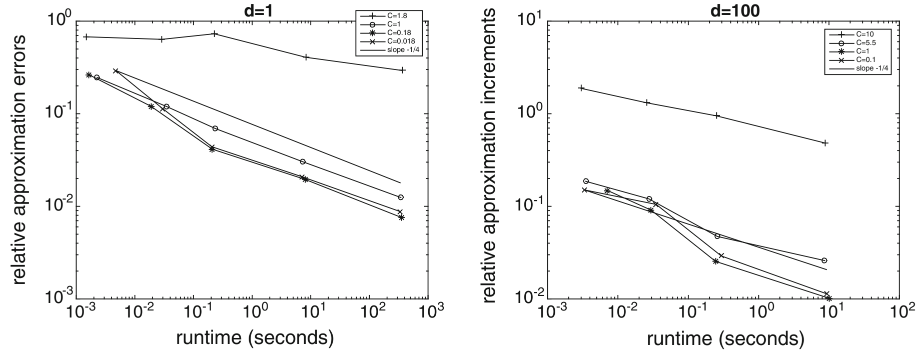
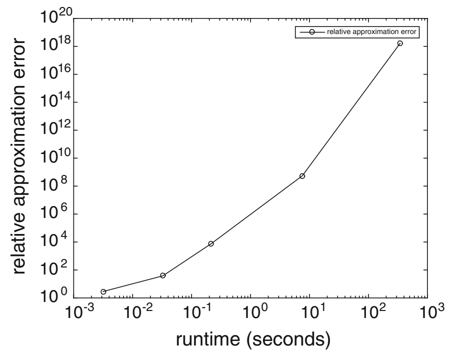


Fig. 3 Empirical convergence of the scheme (11) for the Allen–Cahn equation in Sect. 3.3. Left: Relative approximation errors $\frac{1}{10|\mathrm{v}|}\sum_{i=1}^{10} |\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)-\mathrm{v}|$ against the average runtime for $\rho \in \{1,2,\ldots,5\}$ and $C \in \{0.018, 0.18, 1, 1.8\}$ in the case $d = 1$. Right: Relative approximation increments $\left(\frac{1}{10}\sum_{i=1}^{10} |\mathbf{U}^{i,[1]}_{\rho+1,\rho+1}(0,x_0)-\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0)|\right)\Big/\left(\frac{1}{10}|\sum_{i=1}^{10} \mathbf{U}^{i,[1]}_{5,5}(0,x_0)|\right)$ against the average runtime for $\rho \in \{1,2,3,4\}$ and $C \in \{0.1, 1, 5.5, 10\}$ in the case $d = 100$

Theorem 2.1 in Hutzenthaler et al. [57] shows that absolute moments of the stochastic Euler approximations of SDEs with superlinearly growing coefficients at a fixed time point diverge to infinity. In addition, it has been conjectured in Conjecture 5.1 in Hutzenthaler et al. [59] that the absolute value of non-adaptive multilevel Monte Carlo Euler approximations of SDEs with superlinearly growing coefficients at a fixed time point diverge to infinity almost surely. We cannot exclude that an analogous almost sure divergence holds for multilevel Picard approximations of the Allen–Cahn equation (25) which has a super-linearly growing nonlinearity. An indication that a super-linearly growing nonlinearity in combination with time discretization might cause almost sure divergence is Lemma 1.1 in Lionnet et al. [65] which shows that the L^2-norms of explicit backward Euler discretizations of an Allen–Cahn-type PDE at time point 1 diverge.

4 Discussion of Approximation Methods from the Literature

In this section we aim to provide a rough overview on approximation methods for second-order parabolic PDEs from the scientific literature. Deterministic approximation methods for second-order parabolic PDEs are known to have exponentially growing computational



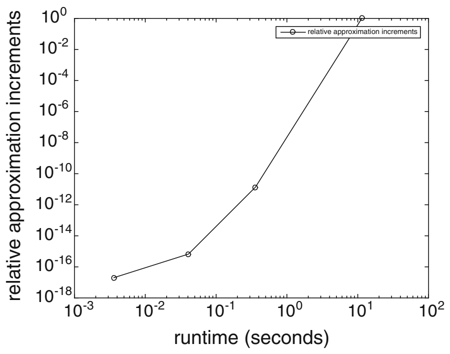


Fig. 4 Empirical divergence of the scheme (11) for the Allen–Cahn equation in Sect. 3.3. Left: Relative approximation errors $\frac{1}{10|\mathrm{v}|}\sum_{i=1}^{10}|\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)-\mathrm{v}|$ against the average runtime for $\rho \in \{1,2,\ldots,5\}$ and $C = 2.7$ in the case $d = 1$. Right: Relative approximation increments $\left(\frac{1}{10}\sum_{i=1}^{10}|\mathbf{U}_{\rho+1,\rho+1}^{i,[1]}(0,x_0)-\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)|\right)/\left(\frac{1}{10}|\sum_{i=1}^{10}\mathbf{U}_{5,5}^{i,[1]}(0,x_0)|\right)$ against the average runtime for $\rho \in \{1,2,3,4\}$ and $C = 15$ in the case $d = 100$

Table 7 Average runtime, empirical mean, empirical standard deviation, and relative approximation increments in the case $d = 100$ and $C = 5.5$ for the Allen–Cahn equation in Sect. 3.3

ρ	1	2	3	4	5
Average runtime in seconds	0.003	0.028	0.259	8.686	407.143
$\overline{\mathbf{U}}_{\rho,\rho}^{[1]}(0,x_0) = \frac{1}{10}\sum_{i=1}^{10}\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)$	0.978	0.916	0.901	0.920	0.924
$\sqrt{\frac{1}{9}\sum_{i=1}^{10}\lvert\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)-\overline{\mathbf{U}}_{\rho,\rho}^{[1]}(0,x_0)\rvert^2}$	0.225	0.111	0.063	0.027	0.008
$\frac{1}{10}\sum_{i=1}^{10}\frac{\left\lvert\mathbf{U}_{\rho+1,\rho+1}^{i,[1]}(0,x_0)-\mathbf{U}_{\rho,\rho}^{i,[1]}(0,x_0)\right\rvert}{\left\lvert\overline{\mathbf{U}}_{5,5}^{[1]}(0,x_0)\right\rvert}$	0.1862	0.1203	0.0473	0.0260	

effort in the PDE dimension. Since a program with 10^{80}, say, floating point operations will never terminate (on a non-quantum computer), deterministic approximation methods such as finite elements methods, finite difference methods, spectral Galerkin approximation methods, or sparse grid methods are not suitable for solving high-dimensional nonlinear second-order parabolic PDEs no matter what the convergence rate of the approximation method is. For this reason we discuss only stochastic approximation methods for nonlinear second-order parabolic PDEs. In the literature there exist many articles (see, e.g., [3,4,7,10–12,14–16,21–24,26,36,39–41,43,44,53–55,62,64,65,69,78,79,84,86]), which propose (possibly non-implementable) stochastic approximation methods for nonlinear second-order parabolic PDEs. Some of these approximation methods (see, e.g., [4,10,69]) aim at approximating the solution at a single space-time point and some approximation methods (see, e.g., [26,41,64]) approximate the solution at all space-time points which is, even in the linear case, computationally expensive; cf., e.g., Theorem 3.2 in Györfi et al. [48]. Except for [14,53–55,62] all of these approximation methods exploit a stochastic representation with BSDEs due to Pardoux and Peng [73]. Moreover, except for [12,14,36,39,53–55,62] all of these approximation methods can be described in two steps. In the first step, time in the corresponding BSDE is discretized backwards in time via an explicit or implicit Euler-type method which was investigated in detail, e.g., in Bouchard and Touzi [10] and Zhang [86]. The resulting approximations involve nested conditional expectations and are, therefore, not implementable. In the second step, these conditional expectations are approximated by 'straight-forward' Monte Carlo simulations, by the quantization tree method (proposed in [4]), by a regression method based on kernel-estimation or Malliavin calculus (proposed in [10]), by projections on function spaces (proposed in [41]), or by the cubature method on Wiener space (developed in [68] and proposed in [22]). The first step does not cause problems in high dimensions in the sense that the backward (explicit or implicit) Euler-type approximations converge under suitable assumptions with rate at least 1/2 (see Theorem 5.3 in Zhang [86] and Theorem 3.1 in Bouchard and Touzi [10] for the backward implicit Euler-type method) and the computational effort (assuming the conditional expectations are known exactly) grows at most linearly in the dimension for fixed accuracy. For this reason, we discuss below in detail only the different approximation methods for discretizing conditional expectations. In addition, we discuss the Wiener chaos decomposition method proposed in [12,36], the branching diffusion method proposed in [53–55], and approximation methods based on density estimation proposed in [14,62].

A difficulty in our discussion below is that the discussed algorithms (except for the branching diffusion method) depend on different parameters and the optimal choice of these parameters is typically unknown since no lower estimates for the approximation errors are known. For this reason we will choose parameters which are optimal with respect to the best known upper error bound. For these parameter choices we will show below for the discussed algorithms (except for the branching diffusion method) that the computational effort fails to grow at most polynomially both in the dimension and in the reciprocal of the best known upper error bound.

Throughout this section assume the setting in Sect. 2.3, let $u^\infty \in C^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ be a function which satisfies (14) and let $Y\colon [0,T]\times\Omega\to\mathbb{R}$ be the stochastic process which satisfies for all $t\in[0,T]$ that $Y_t = u^\infty(t, W_t^0)$.

4.1 The 'Straight-Forward' Monte Carlo Method

The 'straight-forward' Monte Carlo method approximates the conditional expectations involved in backward Euler-type approximations by Monte Carlo simulations. The resulting nesting of Monte Carlo averages is computational expensive in the following sense. If for a finite and non-empty set $\pi\subseteq[0,T]$ with $|\pi|\in\mathbb{N}$ many elements and for $M\in\mathbb{N}$ the random variable $\mathcal{Y}^{\pi,M}\colon \pi\times\Omega\to\mathbb{R}$ is the 'straight-forward' Monte Carlo approximation of $\mathcal{Y}$ with time grid π and M Monte Carlo averages for each involved conditional expectation, then the number of realizations of scalar standard normal random variables required to compute one realization of $\mathcal{Y}^{\pi,M}$ is $(Md)^{|\pi|}$ and the $L^2(\mathbb{P};\mathbb{R})$-error satisfies for a suitable constant $c\in\mathbb{R}$ independent of π and N that

$$\max_{t\in\pi}\|Y_t-\mathcal{Y}_t^{\pi,M}\|_{L^2(\mathbb{P};\mathbb{R})}\le c\left(|\pi|^{-\frac{1}{2}}+|\pi|M^{-1/2}\right) \tag{27}$$

(see, e.g., Crisan and Manolarakis [21, Theorem 4.3 and (4.14)]). Thus the computational effort $(Md)^{(\frac{1}{|\pi|^{-1/2}})^2}\ge(Md)^{(\frac{c}{c|\pi|^{-1/2}+c|\pi|M^{-1/2}})^2}$ grows at least exponentially in the reciprocal of the right-hand side of (27). This suggests an at most logarithmic convergence rate of the 'straight-forward' Monte Carlo method. We are not aware of a statement in the literature claiming that the 'straight-forward' Monte Carlo method has a polynomial convergence rate.

4.2 The Quantization Tree Method

The quantization tree method has been introduced in Bally and Pagès [3,4]. In the proposed algorithm, time is discretized by the explicit backward Euler-type method. Moreover, one chooses a space-time grid and computes the transition probabilities for the underlying forward diffusion projected to this grid by Monte Carlo simulation. With these discrete-space transition probabilities one can then approximate all involved conditional expectations. If for a finite and non-empty set $\pi \subseteq [0, T]$ with $|\pi| \in \mathbb{N}$ many elements and for $N \in \mathbb{N}$ the random variable $\mathcal{Y}^{\pi,N} : \pi \times \Omega \to \mathbb{R}$ is the quantization tree approximation of Y with time grid π, a specific space grid with a total number of N nodes and explicitly known transition probabilities of the forward diffusion and if the coefficients are sufficiently regular, then the number of realizations of scalar standard normal random variables required to compute one realization of $\mathcal{Y}^{\pi,N}$ is at least $Nd|\pi|$ and (6) in Bally and Pagès [3] shows for optimal grids and a constant $c \in \mathbb{R}$ independent of π and N that

$$\max_{t\in\pi} \|Y_t - \mathcal{Y}_t^{\pi,N}\|_{L^2(\mathbb{P};\mathbb{R})} \leq c \left(\frac{1}{|\pi|} + \frac{|\pi|^{1+1/d}}{N^{1/d}} \right). \tag{28}$$

To ensure that this upper bound does not explode as $|\pi| \to \infty$ it is thus necessary to choose a space-time grid with at least $N = |\pi|^{d+1}$ many nodes when there are $|\pi| \in \mathbb{N}$ many time steps. With this choice the computational effort of this algorithm grows exponentially fast in the dimension. We have not found a statement in the literature on the quantization tree method claiming that there exists a choice of parameters such that the computational effort grows at most polynomially both in the dimension and in the reciprocal of the prescribed accuracy.

4.3 The Malliavin Calculus Based Regression Method

The Malliavin calculus based regression method has been introduced in Sect. 6 in Bouchard and Touzi [10] and is based on the implicit backward Euler-type method. The algorithm involves iterated Skorohod integrals which by (3.2) in Crisan et al. [24] can be numerically computed with 2^d many independent standard normally distributed random variables. In that case the computational effort grows exponentially fast in the dimension. We are not aware of an approximation method of the involved iterated Skorohod integrals whose computational effort does not grow exponentially fast in the dimension. Example 4.1 in Bouchard and Touzi [10] also mentions an approximation method for approximating all involved conditional expectations using kernel estimation. For this approximation method we have not found an upper error estimate in the literature so that we do not known how to choose the bandwidth matrix of the kernel estimation given the number of time grid points.

4.4 The Projection on Function Spaces Method

The projection on function spaces method has been proposed in Gobet et al. [41]. The algorithm is based on estimating the involved conditional expectations by considering the projections of the random variables on a finite-dimensional function space and then estimating these projections by Monte Carlo simulation. In general the projection error and the computational effort depend on the choice of the basis functions. In the literature we have found the following three choices of basis functions. In Gobet et al. [41] (see also Gobet and Lemor [40] and Lemor et al. [64]) indicator functions of hypercubes are employed as basis functions. In this case there exists $c \in \mathbb{R}$ such that a projection error $\varepsilon \in (0, \infty)$ can be achieved by simulating $\lfloor c\varepsilon^{-(3+2d)}|\log(\varepsilon)|\rfloor$ paths of the forward diffusion. With this choice, the computational effort of the algorithm grows exponentially fast in the dimension for fixed accuracy $\varepsilon \in (0, 1)$. Gobet and Turkedjiev [44] use local polynomials on disjoint hypercubes as basis functions in order to exploit regularity of solution. With this choice, there exists $c \in (0, \infty)$ such that for fixed accuracy $\varepsilon \in (0, 1)$ the computational effort of the algorithm is at least $c\varepsilon^{-1-\frac{d}{2\kappa+2\eta}}(\log(1+\frac{1}{\varepsilon}))^d$ where $\kappa \in \mathbb{N}$ and $\eta \in (0, 1]$ are such that the true solution u^∞ is uniformly bounded and $\kappa + 1$-continuously space-differentiable with bounded derivatives and the $\kappa + 1$-th derivatives are uniformly η-Hölder continuous in space; see Sect. 4.4 in [43] for details. Also Gobet and Turkedjiev [43] use local polynomials on disjoint hypercubes and the same lower bound holds for the computational complexity. Ruijter and Oosterlee [78] use certain cosine functions as basis functions and motivate this with a Fourier cosine series expansion. The resulting approximation method has only been specified in a one-dimensional setting so that the computational effort in a high-dimensional setting remained unclear. We have not found a statement in the literature on the projection on function spaces method claiming that there exists a choice of function spaces and other algorithm parameters such that the computational effort of the approximation method grows at most polynomially both in the dimension of the PDE and in the reciprocal of the prescribed accuracy.

4.5 The Cubature on Wiener Space Method

The cubature on Wiener space method for approximating solutions of PDEs has been introduced in Crisan and Manolarakis [22]. This approximation method combines the implicit backward Euler-type scheme with the cubature method developed in Lyons and Victoir [68] for constructing finitely supported measures that approximate the distribution of the solution of a stochastic differential equation. This approximation method has a parameter $m \in \mathbb{N}$ and constructs for every finite time grid $\pi \subseteq [0, T]$ (with $|\pi| \in \mathbb{N}$ points) a sequence $w_1, \ldots, w_{(N_{m,d})^{|\pi|}} \in C^0([0, 1], \mathbb{R}^d)$ of paths with bounded variation where $N_{m,d} \in \mathbb{N}$ is the number of nodes needed for a cubature formula of degree m with respect to the d-dimensional Gaussian measure. We note that this construction is independent of f and g and can be computed once and then tabularized. Using these paths, Corollary 4.2 in Crisan and Manolarakis [22] shows in the case $m \geq 3$ that there exists a constant $c \in [0, \infty)$, a sequence $\pi_n \subseteq [0, T]$, $n \in \mathbb{N}$, of finite time grids and there exist implementable approximations $\mathcal{Y}^{\pi_n} : \pi_n \times \Omega \to \mathbb{R}$, $n \in \mathbb{N}$, of the exact solution Y such that for all $n \in \mathbb{N}$ it holds that $0 \in \pi_n$, π_n has $n + 1$ elements and

$$\|Y_0 - \mathcal{Y}_0^{\pi_n}\|_{L^2(\mathbb{P};\mathbb{R})} \leq \tfrac{c}{n}. \tag{29}$$

In this form of the algorithm, the computational effort for calculating $\mathcal{Y}^{\pi_n}$, which is at least the number $(N_{m,d})^n$ of paths to be used, grows exponentially in the reciprocal of the right-hand side of (29). To avoid this exponential growth of the computational effort in the number of cubature paths, Crisan and Manolarakis [22] specify two methods (a tree based branching algorithm of Crisan and Lyons [20] and the recombination method of Litterer and Lyons [66]) which reduce the number of nodes and which result in approximations which converge with polynomial rate; cf. Theorem 5.4 in [22]. The constant in the upper error estimate in Theorem 5.4 in [22] may depend on the dimension (cf. also (5.16) and the proof of Lemma 3.1 in [22]). Simulations in the literature on the cubature method were performed in dimension 1 (see Figs. 1–4 in [22] and Fig. 1 in [23]) or dimension 5 (see Figs. 5–6 in

[22]). To the best of our knowledge, there exist no statement in the literature on the cubature method which asserts that the computational effort of the cubature method together with a suitable complexity reduction method grows at most polynomially both in the dimension of the PDE and in the reciprocal of the prescribed accuracy.

4.6 The Wiener Chaos Decomposition Method

The Wiener chaos decomposition method has been introduced in Briand and Labart [12] and has been extended to the case of BSDEs with jumps in Geiss and Labart [36]. The algorithm is based on Picard iterations of the associated BSDE and evaluates the involved nested conditional expectations using Wiener chaos decomposition formulas. This algorithm does not need to discretize time since Wiener integrals over integrands with explicitly known antiderivative can be simulated exactly. The computational complexity of approximating the solution of a BSDE of dimension d using a Wiener chaos decomposition of order $p \in \mathbb{N}$, $K \in \mathbb{N}$ Picard iterations, $M \in \mathbb{N}$ Monte Carlo samples for each conditional expectation, and $N \in \mathbb{N}$ many time steps can be estimated by $O(K \times M \times p \times (N \times d)^{p+1})$; see Sect. 3.2.2 in Briand and Labart [12]. To ensure that the approximation error converges to 0, the order p of the chaos decomposition has to increase to ∞. This implies that the computational effort fails to grow at most polynomially both in the dimension of the PDE and in the reciprocal of the prescribed accuracy. We are not aware of a result in the literature that establishes a polynomial rate of convergence for the Wiener chaos decomposition method (see, e.g., Remark 4.8 in Briand and Labart [12]).

4.7 The Branching Diffusion Method

The branching diffusion method has been proposed in Henry-Labordère [53]; see also the extensions to the non-Markovian case in Henry-Labordère et al. [55] and to nonlinearities depending on derivatives in Henry-Labordère et al. [54]. This method approximates the nonlinearity f by polynomials and then exploits that the solution of a semilinear PDE with polynomial nonlinearity (KPP-type equations) can be represented as an expectation of a functional of a branching diffusion process due to Skorohod [80]. This expectation can then be numerically approximated with the standard Monte Carlo method and pathwise approximations of the branching diffusion process. The branching diffusion method does not suffer from the 'curse of dimensionality by construction' and works in all dimensions. Its convergence rate is 0.5 if the forward diffusion can be simulated exactly and, in general, its rate is $0.5-$ using a pathwise approximation of the forward diffusion and the multilevel Monte Carlo method proposed in Giles [37].

The major drawback of the branching diffusion method is its insufficient applicability. This approximation method replaces potentially 'nice' nonlinearities by potentially 'non-nice' polynomials. Semilinear PDEs with certain polynomial nonlinearities, however, can 'blow up' in finite time; see, e.g., Fujita [33], Escobedo and Herrero [30] for analytical proofs and see, e.g., Nagasawa and Sirao [72] and Lopez-Mimbela and Wakolbinger [67] for probabilistic proofs. If the approximating polynomial nonlinearity, the time horizon, and the terminal condition satisfy a certain condition, then the PDE does not 'blow up' until time T and the branching diffusion method is known to work well. More specifically, if there exist $\beta \in (0, \infty)$ and functions $a_k \colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$, $k \in \mathbb{N}_0$, such that for all $t \in [0, T]$, $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^d$ it holds that $f(t, x, y, z) = \beta[\sum_{k=0}^{\infty} a_k(t, x) y^k] - \beta y$, if the functions μ and σ are bounded, continuous and Lipschitz in the second argument, and if

$\forall\, x \in \mathbb{R}^d : \eta(x) = x$, then Theorem 2.13 in Henry-Labordère et al. [55] (see also Proposition 4.2 in Henry-Labordère [53] or Theorem 3.12 in Henry-Labordère et al. [54]) shows that a sufficient condition for a stochastic representation with a branching diffusion to hold is that

$$\int_{\sup_{x\in\mathbb{R}^d} |g(x)|}^{\infty} \frac{1}{\beta \max\left\{0, (\sum_{k=0}^{\infty} [\sup_{t\in[0,T], x\in\mathbb{R}^d} |a_k(t,x)|] y^k) - y\right\}}\, dy > T. \tag{30}$$

For the branching diffusion method to converge with rate 0.5 the random variables in the stochastic representation need to have finite second moments which leads to a more restrictive condition than (30); see Remark 2.14 in [55]. However, condition (30) is also necessary for the stochastic representation in [55] to hold if the functions g and a_k, $k \in \mathbb{N}_0$, are constant and strictly positive and if μ and σ are constant (then the PDE (14) reduces to an ODE for which the 'blow-up'-behavior is well-known); see, e.g., Lemma 2.5 in [55].

The branching diffusion method also seems to have difficulties with polynomial nonlinearities where the exact solution does not 'blow up' in finite time. Since no theoretical results are available in this direction, we illustrate this with simulations for an Allen–Cahn equation (a simplified version of the Ginzburg-Landau equation). More precisely, for the rest of this subsection assume that T, μ, σ, f, and g satisfy for all $t \in [0, T]$, $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z \in \mathbb{R}^d$ that $T = 1$, $\mu(x) = \mu(0)$, $\sigma(x) = \sigma(0)$, $f(t, x, y, z) = y - y^3$, and $g(x) = g(0) \geq 0$. Then (14) is an ODE and the solution u^∞ satisfies for all $t \in [0, T]$, $x \in \mathbb{R}^d$ that

$$u^\infty(t, x) = \frac{1}{\sqrt{1 - (1 - (g(0))^{-2})e^{2t-2}}}. \tag{31}$$

In this situation the sufficient condition (30) (choose for all $t \in [0, T]$, $x \in \mathbb{R}^d$, $k \in \mathbb{N}_0 \backslash \{1, 3\}$ that $\beta = 1$, $a_1(t, x) = 2$, $a_3(t, x) = -1$, and $a_k(t, x) = 0$) from Theorem 2.13 in [55] is equivalent to

$$1 < \int_{|g(0)|}^{\infty} \tfrac{1}{y+y^3}\, dy = \left[\ln(y) - \tfrac{1}{2}\ln(1 + y^2)\right]_{y=|g(0)|}^{y=\infty} = \tfrac{1}{2}\left[\ln\left(1 + \tfrac{1}{|g(0)|^2}\right)\right] \tag{32}$$

which is equivalent to $|g(0)| < (e^2 - 1)^{-\frac{1}{2}} = 0.395623\ldots$ We simulated the branching diffusion approximations of $u^\infty(0, 0)$ for different values of $g(0)$. Each approximation is an average over $M = 10^5$ independent copies $(\Psi^i_{0,0})_{i\in\{1,2,\ldots,10^5\}}$ of the random variable $\Psi_{0,0}$ defined in (2.12) in Henry-Labordère et al. [55] where we choose for all $k \in \mathbb{N}_0$ that $p_k = 0.5\, \mathbb{1}_{\{1,3\}}(k)$. We also report the estimated standard deviation $\big(10^{-5} \sum_{i=1}^{10^5} (\Psi^i_{0,0})^2 - \big[10^{-5} \sum_{i=1}^{10^5} \Psi^i_{0,0}\big]^2\big)/\sqrt{10^5}$ of the branching diffusion approximation $10^{-5} \sum_{i=1}^{10^5} \Psi^i_{0,0}$. Table 8 shows that the branching diffusion approximations of $u^\infty(0, 0)$ become poor as $g(0)$ increases from 0.1 to 0.7. Thus the branching diffusion method fails to produce good approximations for $u^\infty(0, 0)$ in our example as soon as condition (30) is not satisfied.

A further minor drawback of the branching diffusion method is that it requires a suitable approximation of the nonlinearity with polynomials and this might not be available. In particular, certain functions (e.g., the function $\mathbb{R} \ni x \mapsto \max\{0, x\} \in \mathbb{R}$) can only be well approximated by polynomials on finite intervals so that choosing suitable approximating polynomials might require appropriate a priori bounds on the exact solution of the PDE.

Table 8 Approximation of the PDE $\frac{\partial}{\partial t}u + \frac{1}{2}\Delta_x u + u - u^3 = 0$ with terminal condition $u(1, \cdot) = g(0)$ for $t \in [0, 1]$, $x \in \mathbb{R}$ provided by the branching diffusion method from Theorem 2.13 in Henry-Labordère et al. [55]

$g(0)$	Exact value $u^\infty(0, 0)$	Approximation	Estimated standard deviation
0.1	0.263540	0.271007	0.0169791
0.2	0.485183	0.499103	0.0361975
0.3	0.649791	0.848879	0.2211004
0.4	0.764605	3.495457	2.8179089
0.5	0.843347	21.68436	20.978325
0.6	0.897811	136.6667	110.02696
0.7	0.936233	7321.326	5404.5849

4.8 Approximations Based on Density Representations

Recently two approximation methods were proposed in Chang et al. [14] and Le Cavil et al. [62]. Both approximation methods are based on a stochastic representation with a McKean-Vlasov-type SDE where u^∞ is the density of a certain measure. As a consequence both approximation methods proposed in [14,62] encounter the difficulty of density estimation in high dimensions. More precisely, if $\bar{u}^{\varepsilon,N,n}$ is the approximation of u^∞ defined in (5.33) in [62] with a uniform time grid with $n \in \mathbb{N}$ time points, $N \in \mathbb{N}$ Monte Carlo averages, and bandwidth matrix $\varepsilon\, \mathrm{I}_{\mathbb{R}^{d\times d}}$ where $\varepsilon \in (0, 1]$, then arguments in the proof of [62, Theorem 5.6] and arguments in the proof of [62, Corollary 5.4] imply under suitable assumptions the existence of a function $c\colon (0, 1] \to (0, \infty)$ and real numbers $C, \bar{C} \in (0, \infty)$ (which are independent of n, N, and ε) such that

$$\sup_{t\in[0,T]} \mathbb{E}\bigg[\int_{\mathbb{R}^d} \Big|\bar{u}^{\varepsilon,N,n}(t,x) - u^\infty(t,x)\Big|\, dx + \int_{\mathbb{R}^d} \Big|(\nabla_x(\bar{u}^{\varepsilon,N,n} - u^\infty))(t,x)\Big|\, dx\bigg] \\ \le c_\varepsilon + \left[\tfrac{\bar{C}}{\varepsilon^{d+3}\sqrt{n}} + \tfrac{C}{\sqrt{\varepsilon^{d+4}N}}\right]\exp\!\Big(\tfrac{C}{\varepsilon^{d+1}}\Big). \tag{33}$$

This upper bound becomes only small if we choose the bandwidth ε small and if n and N grow exponentially in the dimension. The upper bounds established in [14] are less explicit in the dimension. However, the estimates in the proofs in [14] suggest that the number of initial particles in branching particle system approximations defined on pages 30 and 18 in [14] need to grow exponentially in the dimension.

4.9 Summary

Finally, we summarize the discussion in this section. It seems that all approximation methods from the scientific literature which discretize space or which approximate the full solution of the considered PDE suffer from the curse of dimensionality in the case of a general PDE. The straight-forward Monte-Carlo method is also no solution to the approximation problem since its computational effort grows exponentially in the reciprocal of the prescribed accuracy. An approximation method which does not suffer from the curse of dimensionality is the branching diffusion method in the case that the terminal condition, the nonlinearity, or the time horizon is sufficiently small. In addition, the simulations in Sect. 3 indicate that the multilevel Picard approximations do not suffer from the curse of dimensionality if LT is not large where L is the Lipschitz constant of the nonlinearity and where T is the time horizon.

Acknowledgements This project has been partially supported through the research Grants ONR N00014-13-1-0338 and DOE DE-SC0009248 and through the German Research Foundation via RTG 2131 *High-dimensional Phenomena in Probability—Fluctuations and Discontinuity* and via research Grant HU 1889/6-1.

Appendix

Here we provide the MATLAB codes needed to approximate the solutions of the example PDEs from Sect. 3. Throughout this section assume the setting in Sect. 2.3.

MATLAB code 1 below produces one realization of

$$\begin{aligned}&\{1,2,\ldots,7\}\times\{1,2,\ldots,5\}\ni(\rho,i)\mapsto \mathbf{U}^{i}_{\rho,\rho}(0,x_0)\\&=(\mathbf{U}^{i,[1]}_{\rho,\rho}(0,x_0),\mathbf{U}^{i,[2]}_{\rho,\rho}(0,x_0),\ldots,\mathbf{U}^{i,[d+1]}_{\rho,\rho}(0,x_0))\in\mathbb{R}^{d+1}.\end{aligned}\tag{34}$$

For the numerical results in Sects. 3.1, 3.2, for every $d\in\{1,100\}$ we run MATLAB code 1 twice where, in the second run, line 2 of MATLAB code 1 is replaced by `rng(2017)` to initiate the pseudorandom number generator with a different seed. This way we obtain in total 10 independent simulation runs. Moreover, for the numerical results in Sects. 3.3, we run MATLAB code 1 once, where lines 4, 5, and 14 are replaced by `average=10;`, `rhomax=5;`, and `[a,b]=approximateUZabm(n(rho),rho,zeros(dim,1),0);`, respectively.

```
 1  global Mf Mg Q c w T dim f g mu sigma eta;
 2  rng(2016)
 3  format long
 4  average=5;
 5  rhomax=7;
 6  rhomin=1;
 7  [T,dim,f,g,eta,mu,sigma]=modelparameters();
 8  [Mf,Mg,Q,c,w,n] = approxparameters(rhomax);
 9  value=zeros(average,rhomax);
10  time=value;
11  for rho=rhomin:rhomax
12    for k=1:average
13      tic
14      [a, b]=approximateUZgbm(n(rho),rho,100*ones(dim,1),0);
15      value(k,rho)=a;
16      time(k,rho)=toc;
17    end
18  end
19  name = [datestr(now, 'yymmddTHHMMSS') '.mat'];
20  save(name,'n','Q','Mf','Mg','value','time');
```

MATLAB **code 1** A MATLAB code to perform a testrun.

MATLAB code 1 calls the MATLAB functions `approximateUZgbm` (respectively `approximateUZabm`), `modelparameters`, and `approxparameters`. The MATLAB functions `approximateUZgbm` and `approximateUZabm` are presented in MATLAB codes 2 and 3 and implement the schemes (12) and (11), respectively. More precisely, up to rounding errors and the fact that random numbers are replaced by pseudo random numbers, it holds for all $\theta\in\Theta$, $n\in\mathbb{N}_0$, $\rho\in\mathbb{N}$, $x\in\mathbb{R}^d$, $s\in[0,T)$ that `approximateUZgbm`(n,ρ,x,s) returns one realization of $\mathbf{U}^{\theta}_{n,\rho}(s,x)$ satisfying (12). Moreover, up to rounding errors and the fact that random numbers are replaced by pseudo random numbers, it holds for all $\theta \in \Theta$, $n \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $x \in \mathbb{R}^d$, $s \in [0, T)$ that `approximateUZabm`(n, ρ, x, s) returns one realization of $\mathbf{U}^{\theta}_{n,\rho}(s, x)$ satisfying (11).

```
function [u, z] = approximateUZgbm(n,rho,x,s)
 global Mf Mg Q c w T dim f g mu sigma;
 cloc=(T−s)∗c/T+s;
 wloc=(T−s)∗w/T;
 MC=Mg(rho,n+1);
 W=sqrt(T−s)∗randn(dim,MC);
 X=repmat(x,1,MC).∗exp((mu−sigma^2/2)∗(T−s)+sigma∗W);
 xi=g(X);
 u=sum(xi,2)/MC;
 z=sum(repmat(xi−g(x)∗ones(1,MC),dim,1).∗W,2)./(MC∗(T−s));
 for l=0:(n−1)
  q=Q(rho,n−l);
  d=cloc(1:q,q)−[s;cloc(1:(q−1),q)];
  MC=Mf(rho,n−l);
  X=repmat(x,1,MC);
  W=zeros(dim,MC);
  for k=1:q
   dW=sqrt(d(k))∗randn(dim,MC);
   W=W+dW;
   X=X.∗exp((mu−sigma^2/2)∗d(k)+sigma∗dW);
   [U, Z]=cellfun(@approximateUZgbm, num2cell(l∗ones(1,MC)), num2cell(rho∗ones(1,MC)),...
       num2cell(X,1), num2cell(cloc(k,q)∗ones(1,MC)),'UniformOutput',false);
   y=f(cloc(k,q),X,cell2mat(U),cell2mat(Z));
   u=u+wloc(k,q)∗sum(y)/MC;
   z=z+wloc(k,q).∗sum(repmat(y,dim,1).∗W,2)./(MC∗(cloc(k,q)−s));
    if l>0
     [U, Z]=cellfun(@approximateUZgbm, num2cell((l−1)∗ones(1,MC)), num2cell(rho∗ones(1,MC)),...
         num2cell(X,1), num2cell(cloc(k,q)∗ones(1,MC)),'UniformOutput',false);
     y=f(cloc(k,q),X,cell2mat(U),cell2mat(Z));
     u=u−wloc(k,q)∗sum(y)/MC;
     z=z−wloc(k,q).∗sum(repmat(y,dim,1).∗W,2)./(MC∗(cloc(k,q)−s));
    end
  end
 end
end
```

MATLAB **code 2** A MATLAB function with input $\theta \in \Theta$, $n \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $x \in \mathbb{R}^d$, $t \in [0, T)$ and output one realization of $\mathbf{U}^{\theta}_{n,\rho}(t, x)$ satisfying (12).

```
function [u, z] = approximateUZabm(n,rho,x,s)
 global Mf Mg Q c w T dim f g mu sigma;
 cloc=(T−s)∗c/T+s;
 wloc=(T−s)∗w/T;
 MC=Mg(rho,n+1);
 W=sqrt(T−s)∗randn(dim,MC);
 X=repmat(x,1,MC)+mu∗(T−s)+sigma∗W;
 xi=g(X);
 u=sum(xi,2)/MC;
 z=sum(repmat(xi−g(x)∗ones(1,MC),dim,1).∗W,2)/(MC∗(T−s));
 for l=0:(n−1)
  q=Q(rho,n−l);
  d=cloc(1:q,q)−[s;cloc(1:(q−1),q)];
  MC=Mf(rho,n−l);
  X=repmat(x,1,MC);
  W=zeros(dim,MC);
  for k=1:q
   dW=sqrt(d(k))∗randn(dim,MC);
   W=W+dW;
   X=X+mu∗d(k)+sigma∗dW;
   [U, Z]=cellfun(@approximateUZabm, num2cell(l∗ones(1,MC)), num2cell(rho∗ones(1,MC)),...
       num2cell(X,1), num2cell(cloc(k,q)∗ones(1,MC)),'UniformOutput',false);
   y=f(cloc(k,q),X,cell2mat(U),cell2mat(Z));
   u=u+wloc(k,q)∗sum(y)/MC;
   z=z+wloc(k,q).∗sum(repmat(y,dim,1).∗W,2)./(MC∗(cloc(k,q)−s));
   if l>0
```

```
          [U, Z]=cellfun(@approximateUZabm, num2cell((l−1)∗ones(1,MC)), num2cell(rho∗ones(1,MC)),...
              num2cell(X,1), num2cell(cloc(k,q)∗ones(1,MC)),'UniformOutput',false);
          y=f(cloc(k,q),X,cell2mat(U),cell2mat(Z));
          u=u−wloc(k,q)∗sum(y)/MC;
          z=z−wloc(k,q).∗sum(repmat(y,dim,1).∗W,2)./(MC∗(cloc(k,q)−s));
        end
      end
    end
end
```

MATLAB **code 3** A MATLAB function with input $\theta \in \Theta$, $n \in \mathbb{N}_0$, $\rho \in \mathbb{N}$, $x \in \mathbb{R}^d$, $t \in [0, T)$ and output one realization of $\mathbf{U}^{\theta}_{n,\rho}(t, x)$ satisfying (11).

The MATLAB function `modelparameters` called in line 7 of MATLAB code 1 returns the parameters $T \in (0, \infty)$, $d \in \mathbb{N}$, $f : [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$, $g : \mathbb{R}^d \to \mathbb{R}$, $\eta : \mathbb{R}^d \to \mathbb{R}$, $\bar{\mu} \in \mathbb{R}$, and $\bar{\sigma} \in \mathbb{R}$ for each example considered in Sects. 3.1–3.3. MATLAB code 4 presents the implementation for the setting in Sect. 3.1 in the case $d = 100$ and $T = 2$.

```
function [T,dim,f,g,eta,mu,sigma] = modelparameters()
    T=2;
    dim=100;
    sigma=0.2;
    mu=0;
    beta=0.03;
    f = @(t,x,y,z) beta∗(subplus(y)−y);
    K1=45.6789;
    K2=49.5468;
    L = (K2−K1)/2;
    g = @(x) subplus(min(x,[],1)−K1)−subplus(min(x,[],1)−K2)−L;
    eta=@(x) x;
end
```

MATLAB **code 4** A MATLAB function that returns the parameter values for the pricing with counterparty credit risk example of Sect. 3.1.

The MATLAB function `approxparameters` called in line 8 of MATLAB code 1 provides for every example considered in Sects. 3.1, 3.2 (respectively Sect. 3.3) and every $\rho \in \{1, 2, \ldots, 7\}$ (respectively $\rho \in \{1, 2, \ldots, 5\}$) the numbers of Monte-Carlo samples $(m^g_{k,l,\rho})_{k,l \in \mathbb{N}_0}$ and $(m^f_{k,l,\rho})_{k,l \in \mathbb{N}_0}$ and the quadrature formulas $(q^{k,l,\rho}_s)_{k,l \in \mathbb{N}_0, s \in [0,T)}$. More precisely, we assume for every $s \in [0, T]$, $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ with $k \geq l$ that $q^{k,l,\rho}_s$ is the Gauss–Legendre quadrature formula on (s, T) with $\operatorname{round}(\varphi(\rho^{(k-l)/2}))$ nodes, where $\varphi : [1, \infty) \to [2, \infty)$ is the approximation of the inverse gamma function provided by MATLAB code 6. To compute the Gauss–Legendre nodes and weights we use the MATLAB function `lgwt` that was written by Greg von Winckel and that can be downloaded from www.mathworks.com. In addition, for every $k, l \in \mathbb{N}_0$, $\rho \in \mathbb{N}$ we choose in Sects. 3.1, 3.2 that $m^f_{k,l,\rho} = \operatorname{round}(\rho^{(k-l)/2})$ and $m^g_{k,l,\rho} = \rho^{k-l}$ and in Sect. 3.3 that $m^f_{k,l,\rho} = \rho^{k-l}$ and $m^g_{k,l,\rho} = \rho^{k-l}$. For the numerical results in Sects. 3.1, 3.2 MATLAB code 5 presents the implementation of `approxparameters`. For the numerical results in Sect. 3.3 line 10 in MATLAB code 5 is replaced by `Mf(rho,k)=rho^k;`. The reason for choosing in Sects. 3.1, 3.2 fewer Monte-Carlo samples $(m^f_{k,l,\rho})_{k,l \in \mathbb{N}_0, \rho \in \mathbb{N}}$ than in Sect. 3.3 is that in the former cases for every $s \in [0, T)$ the variance $\operatorname{Var}(f(s, X^{0,x_0}_s, \mathbb{E}[g(X^{s,x}_T)(1, \frac{W_T - W_s}{T-s})]\big|_{x = X^{0,x_0}_s}))$ of the nonlinearity is of smaller magnitude than the variance $\operatorname{Var}(g(X^{0,x_0}_T))$ of the terminal condition. Therefore, the nonlinearity requires fewer Monte-Carlo samples to obtain a Monte-Carlo error of the same magnitude as the terminal condition. Averaging the nonlinearity less saves computational effort and allows to employ a higher maximal number of Picard iterations (7 in Sects. 3.1, 3.2 compared to 5 in Sect. 3.3).

```
function [Mf,Mg,Q,c,w,n] = approxparameters(rhomax)
    global T;
    n=1:1:rhomax;
    Q=zeros(rhomax);
    Mf=Q;
    Mg=zeros(rhomax,rhomax+1);
    for rho=1:rhomax
        for k=1:n(rho)
            Q(rho,k)=round(inverse_gamma(rho^(k/2)));
            Mf(rho,k)=round((rho)^((k)/2));
            Mg(rho,k)=rho^(k−1);
        end
        Mg(rho,rho+1)=rho^rho;
    end
    qmax=max(max(Q));
    c=zeros(qmax);
    w=c;
    for k=1:qmax
        [ctemp,wtemp] = lgwt(k,0,T);
        c(:,k)=[flip(ctemp);zeros(qmax−k,1)];
        w(:,k)=[flip(wtemp);zeros(qmax−k,1)];
    end
end
```

MATLAB **code 5** A MATLAB function that returns the approximation parameters.

```
function y=inverse_gamma(x)
    c=0.036534;
    L= log((x+c)/sqrt(2∗pi));
    y=L/lambertw(L/exp(1))+0.5;
end
```

MATLAB **code 6** A MATLAB function to approximate the inverse Gamma function.

Solutions of one-dimensional PDEs can be efficiently approximated by finite difference approximation schemes. MATLAB code 7 implements such an approximation scheme in the setting of Proposition 2.2 and MATLAB code 8 implements such an approximation scheme in the setting of Proposition 2.1.

```
function y=approximateUfinitediffgbm(t0,x0,N)
    [T,dim,f,g,eta,mu,sigma]=modelparameters();
    h=(T−t0)/N;
    t=t0:h:T;
    u=1+mu∗h+sigma∗sqrt(h);
    d=1+mu∗h−sigma∗sqrt(h);
    x=x0∗d^N∗(u/d).^(0:N);
    M=(1/2∗[full(gallery('tridiag',ones(N−1,1),ones(N,1),zeros(N−1,1)));[zeros(1,N−1),1]]);
    L=1/(2∗sqrt(h))∗([full(gallery('tridiag',ones(N−1,1),−ones(N,1),zeros(N−1,1)));[zeros(1,N−1),1]]);
    y=g(x);
    for i=N:−1:1
        x=x(1:i)/d;
        z=y∗L(1:i+1,1:i);
        y=y∗M(1:i+1,1:i);
        y=y+h∗f(t(i),x,y,z);
    end
end
```

MATLAB **code 7** A MATLAB code to approximate the solution u of (14) at $t_0 \in [0, T)$, $x_0 \in \mathbb{R}$ with a finite difference approximation scheme in the setting of Proposition 2.2 with $d = 1$.

```
function y=approximateUfinitediffabm(t0,x0,N)
    [T,dim,f,g,eta,mu,sigma]=modelparameters();
    h=(T-t0)/N;
    t=t0:h:T;
    u=mu*h+sigma*sqrt(h);
    d=mu*h-sigma*sqrt(h);
    x=x0+d*N+(0:N)*(u-d);
    M=(1/2*[full(gallery('tridiag',ones(N-1,1),ones(N,1),zeros(N-1,1)));[zeros(1,N-1),1]]);
    L=1/(2*sqrt(h))*([full(gallery('tridiag',ones(N-1,1),-ones(N,1),zeros(N-1,1)));[zeros(1,N-1),1]]);
    y=g(x);
    for i=N:-1:1
        x=x(1:i)-d;
        z=y*L(1:i+1,1:i);
        y=y*M(1:i+1,1:i);
        y=y+h*f(t(i),x,y,z);
    end
end
```

MATLAB **code 8** A MATLAB code to approximate the solution u of (14) at $t_0 \in [0, T)$, $x_0 \in \mathbb{R}$ with a finite difference approximation scheme in the setting of Proposition 2.1 with $d = 1$.

The command `ploterrorvsruntime(v,value,time)` (the matrices `value` and `time` are produced in MATLAB code 1 and the value `v` by MATLAB code 7 or 8) plots the error (17) against the runtime (cf. the left-hand side of Figs. 1–3).

```
function ploterrorvsruntime(v,value,time)
    merror=mean(abs(value-v))/abs(v);
    mtime=mean(time);
    loglog(mtime,merror,'black-o');
    hold on
    loglog(mtime,1./(mtime).^(1/3)*mtime(1)^(1/3)*merror(1),'black');
    ylabel('relative␣approximation␣error');
    xlabel('runtime␣(seconds)');
    legend('relative␣approximation␣error','slope␣-1/3');
end
```

MATLAB **code 9** A MATLAB function to plot relative approximation errors against runtime.

The command `plotincrementsvsruntime(value,time)` (the matrices `value` and `time` are produced in MATLAB code 1) plots the increments (18) against the runtime (cf. the right-hand side of Figs. 1–3).

```
function plotincrementvsruntime(value,time)
    [~,rhomax]=size(value);
    merror=mean(abs(value(:,2:rhomax)-value(:,1:rhomax-1)))/abs(mean(value(:,rhomax)));
    mtime=mean(time(:,1:rhomax-1));
    loglog(mtime,merror,'black-o');
    hold on
    loglog(mtime,1./(mtime).^(1/3)*mtime(1)^(1/3)*merror(1),'black');
    ylabel('relative␣approximation␣increments');
    xlabel('runtime␣(seconds)');
    legend('relative␣approximation␣increments','slope␣-1/3');
end
```

MATLAB **code 10** A MATLAB function to plot relative approximation increments against runtime.

The three graphs of Fig. 5 are produced with the help of MATLAB codes 11 and 12. More precisely, up to rounding errors and the fact that random numbers are replaced by pseudo random numbers, MATLAB code 11 generates for every $d \in \{5, 6, \ldots, 100\}$ one realization of $\mathbf{U}^0_{6,6}(0, x_0)$ with $x_0 = (100, \ldots, 100) \in \mathbb{R}^d$ and records the associated runtimes. MATLAB code 11 calls MATLAB code 12 to plot the three graphs in Fig. 5 where, for the right-hand side of Fig. 5, lines 4 and 11 in MATLAB code 11 are replaced by `average=20;` and `rhomax=4;`, respectively.

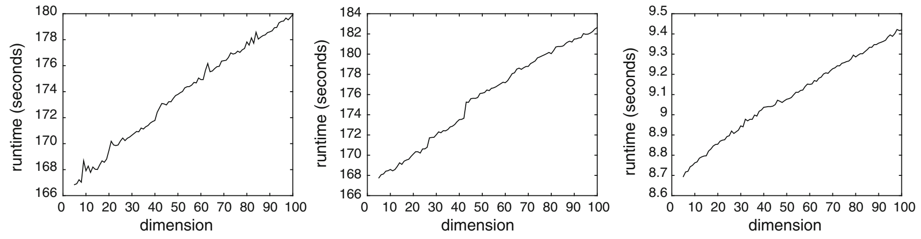


Fig. 5 Left: Runtime needed to compute one realization of $\mathbf{U}^1_{6,6}(0, x_0)$ against dimension $d \in \{5, 6, \ldots, 100\}$ for the pricing with counterparty credit risk example in Sect. 3.1. Middle: Runtime needed to compute one realization of $\mathbf{U}^1_{6,6}(0, x_0)$ against dimension $d \in \{5, 6, \ldots, 100\}$ for the pricing with different interest rates example in Sect. 3.2. Right: Average runtime needed to compute 20 realizations of $\mathbf{U}^1_{4,4}(0, x_0)$ against dimension $d \in \{5, 6, \ldots, 100\}$ for the Allen–Cahn equation in Sect. 3.3

```
global Mf Mg Q c w T dim f g mu sigma eta;
rng(2016)
format long
average=1;
rhomax=6;
dmax=100;
dmin=5;
[T,dim,f,g,eta,mu,sigma]=modelparameters();
[Mf,Mg,Q,c,w,n] = approxparameters(rhomax);
value=zeros(average,dmax−dmin+1);
time=value;
for d=dmin:dmax
  for k=1:average
    dim=d
    tic
    [a, b]=approximateUZgbm(n(rhomax),rhomax,100∗ones(d,1),0);
    value(k,d−dmin+1)=a;
    time(k,d−dmin+1)=toc;
  end
end
name = [datestr(now, 'yymmddTHHMMSS') '.mat'];
save(name,'n','Q','Mf','Mg','value','time')
plotruntimevsdim(dmin, dmax, time)
```

MATLAB **code 11** A MATLAB code to compute one realization of $\mathbf{U}^1_{6,6}(0, x_0)$ for all $d \in \{5, 6, \ldots, 100\}$.

```
function plotruntimevsdim(dmin, dmax, time)
    mtime=mean(time,1);
    plot((dmin:dmax),mtime,'black')
    hold on
    ylabel('runtime␣(seconds)')
    xlabel('dimension')
end
```

MATLAB **code 12** A MATLAB code to plot the runtime against the dimension.

References

1. Amadori, A.L.: Nonlinear integro-differential evolution problems arising in option pricing: a viscosity solutions approach. Differ. Integral Equ. **16**(7), 787–811 (2003)
2. Avellaneda, M., Levy, A., Parás, A.: Pricing and hedging derivative securities in markets with uncertain volatilities. Appl. Math. Finance **2**(2), 73–88 (1995)

3. Bally, V., Pagès, G.: Error analysis of the optimal quantization algorithm for obstacle problems. Stoch. Process. Appl. **106**(1), 1–40 (2003)
4. Bally, V., Pagès, G.: A quantization algorithm for solving multi-dimensional discrete-time optimal stopping problems. Bernoulli **9**(6), 1003–1049 (2003)
5. Bayraktar, E., Milevsky, M.A., Promislow, S.D., Young, V.R.: Valuation of mortality risk via the instantaneous Sharpe ratio: applications to life annuities. J. Econ. Dyn. Control **33**(3), 676–691 (2009)
6. Bayraktar, E., Young, V.: Pricing options in incomplete equity markets via the instantaneous Sharpe ratio. Ann. Finance **4**(4), 399–429 (2008)
7. Bender, C., Denk, R.: A forward scheme for backward SDEs. Stoch. Process. Appl. **117**(12), 1793–1812 (2007)
8. Bender, C., Schweizer, N., Zhuo, J.: A primal-dual algorithm for BSDEs. Math. Finance **27**, 866–901 (2015)
9. Bergman, Y.Z.: Option pricing with differential interest rates. Rev. Financ. Stud. **8**(2), 475–500 (1995)
10. Bouchard, B., Touzi, N.: Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. Stoch. Process. Appl. **111**(2), 175–206 (2004)
11. Briand, P., Delyon, B., Mémin, J.: Donsker-type theorem for BSDEs. Electron. Commum. Probab. **6**, 1–14 (2001)
12. Briand, P., Labart, C.: Simulation of BSDEs by Wiener chaos expansion. Ann. Appl. Probab. **24**(3), 1129–1171 (2014)
13. Burgard, C., Kjaer, M.: Partial differential equation representations of derivatives with bilateral counterparty risk and funding costs. J. Credit Risk **7**(3), 1–19 (2011)
14. Chang, D., Liu, H., Xiong, J.: A branching particle system approximation for a class of FBSDEs. Probab. Uncertain. Quantit. Risk **1**(1), 9 (2016)
15. Chassagneux, J.-F.: Linear multistep schemes for BSDEs. SIAM J. Numer. Anal. **52**(6), 2815–2836 (2014)
16. Chassagneux, J.-F., Crisan, D.: Runge–Kutta schemes for backward stochastic differential equations. Ann. Appl. Probab. **24**(2), 679–720 (2014)
17. Cheridito, P., Soner, H.M., Touzi, N., Victoir, N.: Second-order backward stochastic differential equations and fully nonlinear parabolic PDEs. Commun. Pure Appl. Math. **60**(7), 1081–1110 (2007)
18. Crépey, S., Gerboud, R., Grbac, Z., Ngor, N.: Counterparty risk and funding: the four wings of the TVA. Int. J. Theor. Appl. Finance **16**(02), 1350006 (2013)
19. Creutzig, J., Dereich, S., Müller-Gronbach, T., Ritter, K.: Infinite-dimensional quadrature and approximation of distributions. Found. Comput. Math. **9**(4), 391–429 (2009)
20. Crisan, D., Lyons, T.: Minimal entropy approximations and optimal algorithms for the filtering problem. Monte Carlo Methods Appl. **8**(4), 343–356 (2002)
21. Crisan, D., Manolarakis, K.: Probabilistic methods for semilinear partial differential equations. Applications to finance. ESAIM Math. Model. Numer. Anal. **44**(05), 1107–1133 (2010)
22. 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)
23. Crisan, D., Manolarakis, K.: Second order discretization of backward SDEs and simulation with the cubature method. Ann. Appl. Probab. **24**(2), 652–678 (2014)
24. Crisan, D., Manolarakis, K., Touzi, N.: On the Monte Carlo simulation of BSDEs: an improvement on the Malliavin weights. Stoch. Process. Appl. **120**(7), 1133–1158 (2010)
25. Da Prato, G., Zabczyk, J.: Differentiability of the Feynman–Kac semigroup and a control application. Atti Accad. Naz. Lincei Cl. Sci. Fis. Mat. Natur. Rend. Lincei (9) Mat. Appl. **8**(3), 183–188 (1997)
26. Delarue, F., Menozzi, S.: A forward-backward stochastic algorithm for quasi-linear PDEs. Ann. Appl. Probab. **16**, 140–184 (2006)
27. E, W., Hutzenthaler, M., Jentzen, A., Kruse, T.: Multilevel Picard iterations for solving smooth semilinear parabolic heat equations. arXiv:1607.03295 (2016)
28. El Karoui, N., Peng, S., Quenez, M.C.: Backward stochastic differential equations in finance. Math. Finance **7**(1), 1–71 (1997)
29. Elworthy, K., Li, X.-M.: Formulae for the derivatives of heat semigroups. J. Funct. Anal. **125**(1), 252–286 (1994)
30. Escobedo, M., Herrero, M.A.: Boundedness and blow up for a semilinear reaction–diffusion system. J. Differ. Equ. **89**(1), 176–202 (1991)
31. Fahim, A., Touzi, N., Warin, X.: A probabilistic numerical method for fully nonlinear parabolic PDEs. Ann. Appl. Probab. **21**, 1322–1364 (2011)
32. Forsyth, P.A., Vetzal, K.R.: Implicit solution of uncertain volatility/transaction cost option pricing models with discretely observed barriers. Appl. Numer. Math. **36**(4), 427–445 (2001)
33. Fujita, H.: On the blowing up of solutions of the Cauchy problem for $u_t = \Delta u + u^{1+\alpha}$. J. Fac. Sci. Univ. Tokyo Sect. I **13**, 109–124 (1966)

34. Geiss, C., Geiss, S., Gobet, E.: Generalized fractional smoothness and l^p-variation of BSDEs with non-Lipschitz terminal condition. Stoch. Process. Appl. **122**(5), 2078–2116 (2012)
35. Geiss, C., Geiss, S.: On approximation of a class of stochastic integrals and interpolation. Stoch. Stoch. Rep. **76**(4), 339–362 (2004)
36. Geiss, C., Labart, C.: Simulation of BSDEs with jumps by Wiener Chaos expansion. Stoch. Process. Appl. **126**(7), 2123–2162 (2016)
37. Giles, M.: Improved multilevel Monte Carlo convergence using the Milstein scheme. In: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, pp. 343–358. Springer, Berlin (2008)
38. Giles, M.B.: Multilevel Monte Carlo path simulation. Oper. Res. **56**(3), 607–617 (2008)
39. Gobet, E., Labart, C.: Solving BSDE with adaptive control variate. SIAM J. Numer. Anal. **48**(1), 257–277 (2010)
40. Gobet, E., Lemor, J.-P.: Numerical simulation of BSDEs using empirical regression methods: theory and practice. arXiv:0806.4447 (2008)
41. 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)
42. Gobet, E., Makhlouf, A.: l_2-time regularity of BSDEs with irregular terminal functions. Stoch. Process. Appl. **120**(7), 1105–1132 (2010)
43. Gobet, E., Turkedjiev, P.: Approximation of backward stochastic differential equations using Malliavin weights and least-squares regression. Bernoulli **22**(1), 530–562 (2016)
44. Gobet, E., Turkedjiev, P.: Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions. Math. Comput. **85**(299), 1359–1391 (2016)
45. Graham, C., Talay, D.: Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation, vol. 68 of Stochastic Modelling and Applied Probability. Springer, Heidelberg (2013)
46. Guo, W., Zhang, J., Zhuo, J.: A monotone scheme for high-dimensional fully nonlinear PDEs. Ann. Appl. Probab. **25**(3), 1540–1580 (2015)
47. Guyon, J., Henry-Labordère, P.: The uncertain volatility model: a Monte Carlo approach. J. Comput. Finance **14**(3), 385–402 (2011)
48. Györfi, L., Kohler, M., Krzyzak, A., Walk, H.: A Distribution-Free Theory of Nonparametric Regression. Springer, Berlin (2006)
49. Heinrich, S.: Monte Carlo complexity of global solution of integral equations. J. Complex. **14**(2), 151–175 (1998)
50. Heinrich, S.: Multilevel Monte Carlo methods. In: Margenov, S., Waśniewski, J., Yalamov, P. (eds.) Large-Scale Scientific Computing, pp. 58–67. Springer, Berlin (2001)
51. Heinrich, S.: The randomized information complexity of elliptic PDE. J. Complex. **22**(2), 220–249 (2006)
52. Heinrich, S., Sindambiwe, E.: Monte Carlo complexity of parametric integration. J. Complex. **15**(3), 317–341 (1999). (Dagstuhl Seminar on Algorithms and Complexity for Continuous Problems (1998))
53. Henry-Labordère, P.: Counterparty risk valuation: a marked branching diffusion approach. arXiv:1203.2369 (2012)
54. Henry-Labordere, P., Oudjane, N., Tan, X., Touzi, N., Warin, X.: Branching diffusion representation of semilinear PDEs and Monte Carlo approximation. arXiv:1603.01727 (2016)
55. 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)
56. Hutzenthaler, M., Jentzen, A.: On a perturbation theory and on strong convergence rates for stochastic ordinary and partial differential equations with non-globally monotone coefficients. arXiv:1401.0295 (2014)
57. Hutzenthaler, M., Jentzen, A., Kloeden, P.E.: Strong and weak divergence in finite time of Euler's method for stochastic differential equations with non-globally Lipschitz continuous coefficients. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **467**, 1563–1576 (2011)
58. Hutzenthaler, M., Jentzen, A., Kloeden, P.E.: Strong convergence of an explicit numerical method for SDEs with nonglobally Lipschitz continuous coefficients. Ann. Appl. Probab. **22**(4), 1611–1641 (2012)
59. Hutzenthaler, M., Jentzen, A., Kloeden, P.E.: Divergence of the multilevel Monte Carlo Euler method for nonlinear stochastic differential equations. Ann. Appl. Probab. **23**(5), 1913–1966 (2013)
60. Kloeden, P.E., Platen, E.: Numerical Solution of Stochastic Differential Equations, vol. 23 of Applications of Mathematics (New York). Springer, Berlin (1992)
61. Laurent, J.-P., Amzelek, P., Bonnaud, J.: An overview of the valuation of collateralized derivative contracts. Rev. Deriv. Res. **17**(3), 261–286 (2014)
62. Le Cavil, A., Oudjane, N., Russo, F.: Forward Feynman–Kac type representation for semilinear nonconservative partial differential equations. arXiv:1608.04871 (2016)

63. Leland, H.: Option pricing and replication with transaction costs. J. Finance **40**(5), 1283–1301 (1985)
64. Lemor, J.-P., Gobet, E., Warin, X.: Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations. Bernoulli **12**(5), 889–916 (2006)
65. Lionnet, A., Dos Reis, G., Szpruch, L.: Time discretization of FBSDE with polynomial growth drivers and reaction-diffusion PDEs. Ann. Appl. Probab. **25**(5), 2563–2625 (2015)
66. Litterer, C., Lyons, T.: High order recombination and an application to cubature on Wiener space. Ann. Appl. Probab. **22**, 1301–1327 (2012)
67. López-Mimbela, J.A., Wakolbinger, A.: Length of Galton–Watson trees and blow-up of semilinear systems. J. Appl. Probab. **35**(4), 802–811 (1998)
68. Lyons, T., Victoir, N.: Cubature on Wiener space. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **460**, 169–198 (2004). (Stochastic analysis with applications to mathematical finance (2004))
69. Ma, J., Protter, P., San Martin, J., Torres, S.: Numberical method for backward stochastic differential equations. Ann. Appl. Probab. **12**(1), 302–316 (2002)
70. Ma, J., Zhang, J.: Representation theorems for backward stochastic differential equations. Ann. Appl. Probab. **12**(4), 1390–1418 (2002)
71. Maruyama, G.: Continuous Markov processes and stochastic equations. Rend. Circ. Mat. Palermo (2) **4**, 48–90 (1955)
72. Nagasawa, M., Sirao, T.: Probabilistic treatment of the blowing up of solutions for a nonlinear integral equation. Trans. Am. Math. Soc. **139**, 301–310 (1969)
73. Pardoux, E., Peng, S.: Backward stochastic differential equations and quasilinear parabolic partial differential equations. In: Rozovskii, B.L., Sowers, R.B. (eds.) Stochastic Partial Differential Equations and Their Applications, pp. 200–217. Springer, Berlin (1992)
74. Pardoux, É., Peng, S.G.: Adapted solution of a backward stochastic differential equation. Syst. Control Lett. **14**(1), 55–61 (1990)
75. Peng, S.G.: Probabilistic interpretation for systems of quasilinear parabolic partial differential equations. Stoch. Stoch. Rep. **37**(1–2), 61–74 (1991)
76. Petersdorff, T.V., Schwab, C.: Numerical solution of parabolic equations in high dimensions. ESAIM: Math. Modell. Numer. Anal. Modélisation Mathématique et Analyse Numérique **38**(1), 93–127 (2004)
77. Prévôt, C., Röckner, M.: A Concise Course on Stochastic Partial Differential Equations, Vol. 1905 of Lecture Notes in Mathematics. Springer, Berlin (2007)
78. Ruijter, M.J., Oosterlee, C.W.: A Fourier cosine method for an efficient computation of solutions to BSDEs. SIAM J. Sci. Comput. **37**(2), A859–A889 (2015)
79. Ruijter, M.J., Oosterlee, C.W.: Numerical Fourier method and second-order Taylor scheme for backward SDEs in finance. Appl. Numer. Math. **103**, 1–26 (2016)
80. Skorohod, A.V.: Branching diffusion processes. Teor. Verojatnost. i Primenen. **9**, 492–497 (1964)
81. Smolyak, S.A.: Quadrature and interpolation formulas for tensor products of certain classes of functions. Dokl. Akad. Nauk SSSR **148**, 1042–1045 (1963)
82. Tadmor, E.: A review of numerical methods for nonlinear partial differential equations. Bull. Am. Math. Soc. (N.S.) **49**(4), 507–554 (2012)
83. Thomée, V.: Galerkin Finite Element Methods for Parabolic Problems, Vol. 25 of Springer Series in Computational Mathematics. Springer, Berlin (1997)
84. Turkedjiev, P.: Two algorithms for the discrete time approximation of Markovian backward stochastic differential equations under local conditions. Electron. J. Probab. (2015). https://doi.org/10.1214/EJP.v20-3022
85. Windcliff, H., Wang, J., Forsyth, P., Vetzal, K.: Hedging with a correlated asset: solution of a nonlinear pricing PDE. J. Comput. Appl. Math. **200**(1), 86–115 (2007)
86. Zhang, J.: A numerical scheme for BSDEs. Ann. Appl. Probab. **14**(1), 459–488 (2004)

Affiliations

Weinan E[1] · Martin Hutzenthaler[2] · Arnulf Jentzen[3] · Thomas Kruse[2]

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

Weinan E
weinan@math.princeton.edu

Martin Hutzenthaler
martin.hutzenthaler@uni-due.de

Thomas Kruse
thomas.kruse@uni-due.de

[1] Department of Mathematics and Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544-1000, USA

[2] Faculty of Mathematics, University of Duisburg-Essen, 45117 Essen, Germany

[3] Seminar für Angewandte Mathematik, ETH Zurich, 8092 Zurich, Switzerland