Solving high-dimensional parabolic PDEs using the tensor train format

Lorenz Richter [*1 2 3] **Leon Sallandt** [*4] **Nikolas Nüsken** [5]

Abstract

High-dimensional partial differential equations (PDEs) are ubiquitous in economics, science and engineering. However, their numerical treatment poses formidable challenges since traditional grid-based methods tend to be frustrated by the curse of dimensionality. In this paper, we argue that tensor trains provide an appealing approximation framework for parabolic PDEs: the combination of reformulations in terms of backward stochastic differential equations and regression-type methods in the tensor format holds the promise of leveraging latent low-rank structures enabling both compression and efficient computation. Following this paradigm, we develop novel iterative schemes, involving either explicit and fast or implicit and accurate updates. We demonstrate in a number of examples that our methods achieve a favorable trade-off between accuracy and computational efficiency in comparison with state-of-the-art neural network based approaches.

1. Introduction

While partial differential equations (PDEs) offer one of the most elegant frameworks for modeling in economics, science and engineering, their practical use is often limited by the fact that solving those equations numerically becomes notoriously difficult in high-dimensional settings. The so-called "curse of dimensionality" refers to the phenomenon that the computational effort scales exponentially in the dimension, rendering classical grid based methods infeasible. In recent years there have been fruitful developments in combining Monte Carlo based algorithms with neural networks in order to tackle high-dimensional problems in a way that seemingly does not suffer from this curse, resting primarily on stochastic representations of the PDEs under consideration (E et al., 2017; Raissi et al., 2019; E et al., 2019; Huré et al., 2020; Nüsken & Richter, 2020). Many of the suggested algorithms perform remarkably well in practice and some theoretical results proving beneficial approximation properties of neural networks in the PDE setting are now available (Jentzen et al., 2018). Still, a complete picture remains elusive, and the optimization aspect in particular continues to pose challenging and mostly open problems, both in terms of efficient implementations and theoretical understanding. Most importantly for practical applications, neural network training using gradient descent type schemes may often take a very long time to converge for complicated PDE problems.

Instead of neural networks (NN), we propose relying on the tensor train (TT) format (Oseledets, 2011) to approximate the solutions of high-dimensional PDEs. As we argue in the course of this article, the salient features of tensor trains make them an ideal match for the stochastic methods alluded to in the previous paragraph: First, tensor trains have been designed to tackle high-dimensional problems while still being computationally cheap by exploiting inherent low-rank structures (Kazeev & Khoromskij, 2012; Kazeev et al., 2016; Dolgov et al., 2012) typically encountered in physically inspired PDE models. Second, built-in orthogonality relations allow fast and robust optimization in regression type problems arising naturally in stochastic backward formulations of parabolic PDEs. Third, the function spaces corresponding to tensor trains can be conveniently extended to incorporate additional information such as initial or final conditions imposed on the PDE to be solved. Last but not least, tensor trains allow for extremely efficient and explicit computation of first and higher order derivatives.

To develop TT-based solvers for parabolic PDEs, we follow (Bouchard & Touzi, 2004; Huré et al., 2020) and first identify a backward stochastic differential equation (BSDE) representation of the PDE, naturally giving rise to iterative backward schemes for a numerical treatment. We suggest two versions of our algorithm, allowing to adjust the trade-off between accuracy and speed according to the application: The first scheme is explicit, relying on L^2 projections (Gobet et al., 2005) that can be solved efficiently using an alternating least squares algorithm and explicit expressions for the minimizing parameters (see Section 3.1). The second scheme is implicit and involves a nested iterative procedure,

[*]Equal contribution [1]Freie Universität Berlin, Germany [2]BTU Cottbus-Senftenberg, Germany [3]dida Datenschmiede GmbH, Germany [4]Technische Universität Berlin, Germany [5]Universität Potsdam, Germany. Correspondence to: Lorenz Richter <lorenz.richter@fu-berlin.de>, Leon Sallandt <sallandt@math.tu-berlin.de>.

holding the promise of more accurately resolving highly nonlinear relationships at the cost of an increased computational load. For theoretical underpinning, we prove the convergence of the nested iterative scheme in Section 3.2.

To showcase the performance of the TT-schemes, we evaluate their outputs on various high-dimensional PDEs (including toy examples and real-world problems) in comparison with NN-based approximations. In all our examples, the TT results prove competitive, and often considerably more accurate when low-rank structures can be identified and captured by the underlying ansatz spaces. At the same time, the runtimes of the TT-schemes are usually significantly smaller, with the explicit L^2-projection-based algorithm beating the corresponding NN alternative by orders of magnitude in terms of computational time. Even the more accurate algorithm based on nested nonlinear iterations often proves to be substantially faster than NN training.

1.1. Previous work

Using numerical discretizations of BSDEs to solve PDEs originated in (Bouchard & Touzi, 2004; Gobet et al., 2005), while regression based methods for PDE-related problems in mathematical finance have already been proposed in (Longstaff & Schwartz, 2001). An iterative method motivated by BSDEs and approached with neural networks has been introduced in (E et al., 2017), making the approximation of high-dimensional PDE problems feasible. Solving explicit backwards schemes with neural networks has been suggested in (Beck et al., 2019) and an implicit method similar to the one developed in this paper has been suggested in (Huré et al., 2020). Another interesting method to approximate PDE solutions relies on minimizing a residual term on uniformly sampled data points as suggested in (Sirignano & Spiliopoulos, 2018; Raissi et al., 2019). Rooted in quantum physics under the name *matrix product states*, tensor trains have been introduced to the mathematical community in (Oseledets, 2011) to tackle the curse of dimensionality. Note that tensor trains are a special case of hierarchical tensor networks, which have been developed in (Hackbusch & Kühn, 2009). For good surveys and more details, see (Hackbusch, 2014; Hackbusch & Schneider, 2014; Szalay et al., 2015; Bachmayr et al., 2016). Tensor trains have already been applied to specific types of PDEs (Dolgov et al., 2019; Oster et al., 2019; Khoromskij, 2012) and stochastic exit-time control problems (Fackeldey et al., 2020).

The paper is organized as follows: In Section 2 we motivate our algorithm by recalling the stochastic PDE representation in terms of BSDEs as well as two appropriate discretization schemes. In Section 3 we review the tensor train format as a highly efficient framework for approximating high-dimensional functions by detecting low-rank structures and discuss how those structures can be exploited in the numerical solution of BSDEs. Finally, in Section 4 we provide multiple high-dimensional numerical examples to illustrate our claims.

2. Solving PDEs via BSDEs

In this section we recall how backward stochastic differential equations (BSDEs) can be used to design iterative algorithms for approximating the solutions of high-dimensional PDEs. Throughout this work, we consider parabolic PDEs of the form

$$(\partial_t + L)V(x,t) + h(x,t,V(x,t),(\sigma^\top \nabla V)(x,t)) = 0 \quad (1)$$

for $(x,t) \in \mathbb{R}^d \times [0,T]$, a nonlinearity $h : \mathbb{R}^d \times [0,T] \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$, and a differential operator

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

with coefficient functions $b : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \times [0,T] \to \mathbb{R}^{d\times d}$. The terminal value is given by

$$V(x,T) = g(x), \quad (3)$$

for a specified function $g : \mathbb{R}^d \to \mathbb{R}$. Note that by using the time inversion $t \mapsto T - t$, the terminal value problem (1)-(3) can readily be transformed into an initial value problem.

BSDEs were first introduced in (Bismut, 1973) and their systematic study began with (Pardoux & Peng, 1990). Loosely speaking, they can be understood as nonlinear extensions of the celebrated Feynman-Kac formula (Pardoux, 1998), relating the PDE (1) to the stochastic process X_s defined by

$$\mathrm{d}X_s = b(X_s,s)\,\mathrm{d}s + \sigma(X_s,s)\,\mathrm{d}W_s, \quad X_0 = x_0, \quad (4)$$

where b and σ are as in (2) and W_s is a standard d-dimensional Brownian motion. The key idea is then to define the processes

$$Y_s = V(X_s,s), \qquad Z_s = (\sigma^\top \nabla V)(X_s,s) \quad (5)$$

as representations of the PDE solution and its gradient, and apply Itô's lemma to obtain

$$\mathrm{d}Y_s = -h(X_s,s,Y_s,Z_s)\,\mathrm{d}s + Z_s \cdot \mathrm{d}W_s, \quad (6)$$

with terminal condition $Y_T = g(X_T)$. Noting that the processes Y_s and Z_s are adapted[1] to the filtration generated by the Brownian motion W_s, they should indeed be understood as backward processes and not be confused with time-reversed processes. A convenient interpretation of the

[1] Intuitively, this means that the processes Y_s and Z_s must not depend on future values of the Brownian motion W_s.

relations in (5) is that solving for the processes Y_s and Z_s under the constraint (6) corresponds to determining the solution of the PDE (1) (and its gradient) along a random grid which is provided by the stochastic process X_s defined in (4).

2.1. Numerical approximation of BSDEs

The BSDE formulation (6) opens the door for Monte Carlo algorithms aiming to numerically approximate Y_s and Z_s, and hence yielding approximations of solutions to the PDE (1) according to (5), see (Bouchard & Touzi, 2004; Gobet et al., 2005). In this section we discuss suitable discretizations of (6) and corresponding optimization problems that will provide the backbone for TT-schemes to be developed in Section 3.

To this end, let us define a discrete version of the process (4) on a time grid $0 = t_0 < t_1 < \cdots < t_N = T$ by

$$\widehat{X}_{n+1} = \widehat{X}_n + b(\widehat{X}_n, t_n)\Delta t + \sigma(\widehat{X}_n, t_n)\xi_{n+1}\sqrt{\Delta t}, \quad (7)$$

where $n \in \{0, \dots, N-1\}$ enumerates the steps, $\Delta t = t_{n+1} - t_n$ is the stepsize, $\xi_{n+1} \sim \mathcal{N}(0, \mathrm{Id}_{d\times d})$ are normally distributed random variables and $\widehat{X}_0 = x_0$ provides the initial condition. Two[2] discrete versions of the backward process (6) are given by

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_{n+1}\Delta t + \widehat{Z}_n \cdot \xi_{n+1}\sqrt{\Delta t}, \quad (8a)$$

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_n\Delta t + \widehat{Z}_n \cdot \xi_{n+1}\sqrt{\Delta t}, \quad (8b)$$

where we have introduced the shorthands

$$h_n = h(\widehat{X}_n, t_n, \widehat{Y}_n, \widehat{Z}_n), \quad (9a)$$

$$h_{n+1} = h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1}). \quad (9b)$$

Finally, we complement (8a) and (8b) by specifying the terminal condition $\widehat{Y}_N = g(\widehat{X}_N)$. The reader is referred to Appendix E for further details.

Both of our schemes solve the discrete processes (8a) and (8b) backwards in time, an approach which is reminiscent of the dynamic programming principle in optimal control theory (Fleming & Rishel, 2012), where the problem is divided into a sequence of subproblems. To wit, we start with the known terminal value $\widehat{Y}_N = g(\widehat{X}_N)$ and move backwards in iterative fashion until reaching $\widehat{Y}_0$. Throughout this procedure, we posit functional approximations $\widehat{V}_n(\widehat{X}_n) \approx \widehat{Y}_n \approx V(\widehat{X}_n, n\Delta t)$ to be learnt in the update step $n+1 \to n$ which can either be based on (8a) or on (8b):

Starting with the former, it can be shown by leveraging the relationship between conditional expectations and L^2-projections (see Appendix E) that solving (8a) is equivalent to minimizing

$$\mathbb{E}\left[\left(\widehat{V}_n(\widehat{X}_n) - h_{n+1}\Delta t - \widehat{V}_{n+1}(\widehat{X}_{n+1})\right)^2\right] \quad (10)$$

with respect to $\widehat{V}_n$. Keeping in mind that $\widehat{V}_{n+1}$ is known from the previous step this results in an explicit scheme. Methods based on (10) have been extensively analyzed in the context of linear ansatz spaces for $\widehat{V}_n$ and we refer to (Zhang, 2004; Gobet et al., 2005) as well as to Appendix E.

Moving on to (8b), we may as well penalize deviations in this relation by minimizing the alternative loss

$$\mathbb{E}[(\widehat{V}_n(\widehat{X}_n) - \widehat{h}_n\Delta t + \widehat{Z}_n \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1}))^2], \quad (11)$$

with respect to $\widehat{V}_n$, see (Huré et al., 2020). In analogy to (9a) we use the shorthand notation

$$\widehat{h}_n = h(\widehat{X}_n, t_n, \widehat{V}_n(\widehat{X}_n), \sigma^\top(\widehat{X}_n, t_n)\nabla\widehat{V}_n(\widehat{X}_n)), \quad (12)$$

noting that since $\widehat{h}_n$ depends on $\widehat{V}_n$, approaches based on (11) will necessarily lead to implicit schemes. At the same time, we expect algorithms based on (11) to be more accurate in highly nonlinear scenarios as the dependence in h is resolved to higher order.

3. Solving BSDEs via tensor trains

In this section we discuss the functional approximations $\widehat{V}_n$ in terms of the tensor train format, leading to efficient optimization procedures for (10) and (11). Encoding functions defined on high-dimensional spaces using traditional methods such as finite elements, splines or multi-variate polynomials leads to a computational complexity that scales exponentially in the state space dimension d. However, interpreting the coefficients of such ansatz functions as entries in a high-dimensional tensor allows us to use tensor compression methods to reduce the number of parameters. To this end, we define a set of functions $\{\phi_1, \dots, \phi_m\}$ with $\phi_i : \mathbb{R} \to \mathbb{R}$, e.g. one-dimensional polynomials or finite elements. The approximation $\widehat{V}$ of $V : \mathbb{R}^d \to \mathbb{R}$ takes the form

$$\widehat{V}(x_1, \dots, x_d) = \sum_{i_1=1}^{m} \cdots \sum_{i_d=1}^{m} c_{i_1,\dots,i_d}\phi_{i_1}(x_1)\cdots\phi_{i_d}(x_d), \quad (13)$$

motivated by the fact that polynomials and other tensor product bases are dense in many standard function spaces (Sickel & Ullrich, 2009). Note that for the sake of simplicity we choose the set of ansatz functions to be the same in every dimension (see Appendix A for more general statements). As expected, the coefficient tensor $c \in \mathbb{R}^{m\times m\times\cdots\times m} \equiv$

[2] It can be shown that both converge to the continuous-time process (6) as $\Delta t \to 0$, see (Kloeden & Platen, 1992).

$\mathbb{R}^{m^d}$ suffers from the curse of dimensionality since the number of entries increases exponentially in the dimension d. In what follows, we review the tensor train format to compress the tensor c.

For the sake of readability we will henceforth write $c_{i_1,\dots,i_d} = c[i_1,\dots,i_d]$ and represent the contraction of the last index of a tensor $w_1 \in \mathbb{R}^{r_1\times m\times r_2}$ with the first index of another tensor $w_2 \in \mathbb{R}^{r_2\times m\times r_3}$ by

$$w = w_1 \circ w_2 \in \mathbb{R}^{r_1\times m\times m\times r_3}, \tag{14a}$$

$$w[i_1,i_2,i_3,i_4] = \sum_{j=1}^{r_2} w_1[i_1,i_2,j]w_2[j,i_3,i_4]. \tag{14b}$$

In the literature on tensor methods, graphical representations of general tensor networks are widely used. In these pictorial descriptions, the contractions $\circ$ of the component tensors are indicated as edges between vertices of a graph. As an illustration, we provide the graphical representation of an order-4 tensor and a tensor train representation (see Definition 1 below) in Figure 1. Further examples can be found in Appendix A.

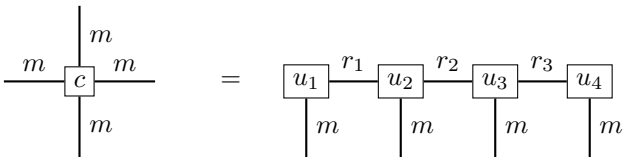


Figure 1. An order 4 tensor and a tensor train representation.

Tensor train representations of c can now be defined as follows (Oseledets, 2011).

Definition 1 (Tensor Train)**.** Let $c \in \mathbb{R}^{m\times\cdots\times m}$. A factorization

$$c = u_1 \circ u_2 \circ \cdots \circ u_d, \tag{15}$$

where $u_1 \in \mathbb{R}^{m\times r_1}$, $u_i \in \mathbb{R}^{r_{i-1}\times m\times r_i}$, $2 \le i \le d-1$, $u_d \in \mathbb{R}^{r_{d-1}\times m}$, is called *tensor train representation* of c. We say that u_i are *component tensors*. The tuple of the dimensions $(r_1,\dots,r_{d-1})$ is called the representation rank and is associated with the specific representation (15). In contrast to that, the tensor train rank (TT-rank) of c is defined as the minimal rank tuple $\mathbf{r} = (r_1,\dots,r_{d-1})$, such that there exists a TT representation of c with representation rank equal to $\mathbf{r}$. Here, minimality of the rank is defined in terms of the partial order relation on $\mathbb{N}^d$ given by

$$\mathbf{s} \preceq \mathbf{t} \iff s_i \le t_i \text{ for all } 1 \le i \le d,$$

for $\mathbf{r} = (r_1,\dots,r_d)$, $\mathbf{s} = (s_1,\dots,s_d) \in \mathbb{N}^d$.

It can be shown that every tensor has a TT-representation with minimal rank, implying that the TT-rank is well defined (Holtz et al., 2012b). An efficient algorithm for computing a minimal TT-representation is given by the Tensor-Train-Singular-Value-Decomposition (TT-SVD) (Oseledets & Tyrtyshnikov, 2009). Additionally, the set of tensor trains with fixed TT-rank forms a smooth manifold, and if we include lower ranks, an algebraic variety is formed (Kutschan, 2018).

Introducing the compact notation

$$\phi : \mathbb{R} \to \mathbb{R}^m, \quad \phi(x) = [\phi_1(x),\dots,\phi_m(x)],$$

the TT-representation of (13) is then given as

$$\widehat{V}(x) = \sum_{i_1}^{m}\cdots\sum_{i_d}^{m}\sum_{j_1}^{r_1}\cdots\sum_{j_{d-1}}^{r_{d-1}} u_1[i_1,j_1]u_2[j_1,i_2,j_2]\cdots \cdots u_d[j_{d-1},i_d]\phi(x_1)[i_1]\cdots\phi(x_d)[i_d]. \tag{16}$$

The corresponding graphical TT-representation (with $d = 4$ for definiteness) is then given as follows:

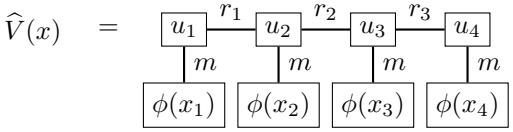


Figure 2. Graphical representation of $\widehat{V} : \mathbb{R}^4 \to \mathbb{R}$.

3.1. Optimization on the TT manifold

The multilinear structure of the tensor product enables efficient optimization of (10) and (11) within the manifold structure by means of reducing a high-dimensional linear equation in the coefficient tensor to small linear subproblems on the component tensors[3]. For this, we view (10) and (11) abstractly as least squares problems on a linear space $\mathcal{U} \subset L^2(\Omega)$, where $\Omega \subset \mathbb{R}^d$ is a bounded Lipschitz domain. Our objective is then to find

$$\operatorname*{arg\,min}_{\widehat{V}\in\mathcal{U}} \sum_{j=1}^{J} |\widehat{V}(x_j) - R(x_j)|^2, \tag{17}$$

where $\{x_1,\dots,x_J\} \subset \Omega$ are data points obtained from samples of $\widehat{X}_n$, and $R : \Omega \to \mathbb{R}$ stands for the terms in (10) and (11) that are not varied in the optimization. Choosing a basis $\{b_1,\dots,b_M\}$ of $\mathcal{U}$ we can represent any function $w \in \mathcal{U}$ by $w(x) = \sum_{m=1}^{M} c_m b_m(x)$ and it is well known that the solution to (17) is given in terms of the coefficient vector

$$c = (A^\top A)^{-1}A^\top r \in \mathbb{R}^M, \tag{18}$$

[3] In the case of (11), an additional nested iterative procedure is required, see Section 3.2.

where $A = [a_{ij}] \in \mathbb{R}^{J\times M}$ with $a_{ij} = b_j(x_i)$ and $r_j = R(x_j) \in \mathbb{R}^J$.

The *alternating least-squares (ALS) algorithm* (Holtz et al., 2012a) reduces the high-dimensional system (18) in the coefficient tensor c to small linear subproblems in the component tensors u_i as follows: Since the tensor train format (15) is a multilinear parametrization of c, fixing every component tensor but one (say u_i) isolates a remaining low-dimensional linear parametrization with associated local linear subspace $\mathcal{U}_{\mathrm{loc},i}$. The number M_i of remaining parameters (equivalently, the dimension of $\mathcal{U}_{\mathrm{loc},i}$) is given by the number of coefficients in the component tensor u_i, i.e. $M_i = r_{i-1}\, m\, r_i$. If the ranks r_i, r_{i-1} are significantly smaller than M, this results in a low-dimensional hence efficiently solvable least-squares problem. Iterating over the component tensors u_i then leads to an efficient scheme for solving high-dimensional least-squares problems with low rank structure. Basis functions in $\mathcal{U}_{\mathrm{loc},i}$ are obtained from the order 3 tensor b^{loc} depicted in Figure 3 (note the three open edges). A simple reshape to an order one tensor then yields the desired basis functions, stacked onto each other, i.e. $b^{\mathrm{loc},i}(x) = [b_1^{\mathrm{loc},i}(x), b_2^{\mathrm{loc},i}(x), \dots, b_{M_i}^{\mathrm{loc},i}(x)]$. Further details as well as explicit formulas are given in Appendix A.1.

$b^{\mathrm{loc},i}(x)$ =

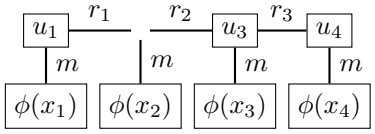


Figure 3. Graphical representation of the local basis functions for $i = 2$.

In many situations the terminal condition g, defined in (3), is not part of the ansatz space just defined. This is always the case if g is not in tensor-product form. However, as the ambient space $\mathbb{R}^{m^d}$ is linear, g can be straightforwardly added[4] to the ansatz space, potentially increasing its dimension to $m^d + 1$. Whenever a component tensor u_i is optimized in the way described above, we simply add g to the set of local basis functions, obtaining as a new basis

$$b_g^{\mathrm{loc},i} = \{b_1^{\mathrm{loc},i}, \dots, b_m^{\mathrm{loc},i}, g\}, \tag{19}$$

only marginally increasing the complexity of the least-squares problem. In our numerical tests we have noticed substantial improvements using the extension (19). Incorporating the terminal condition, the representation of the PDE solution takes the form depicted in Figure 4, for some $c_g \in \mathbb{R}$.

[4]We note that the idea of enhancing the ansatz space has been suggested in (Zhang, 2017) in the context of linear parametrizations.

$\widehat{V}(x)$ =

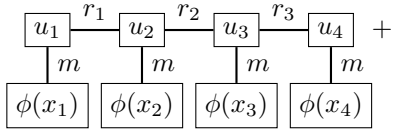


Figure 4. Graphical representation of $\widehat{V} : \mathbb{R}^4 \to \mathbb{R}$.

Summing up, we briefly state a basic ALS algorithm with our adapted basis $b^{\mathrm{loc},i}$:

Algorithm 1 simple ALS algorithm

Input: initial guess $u_1 \circ u_2 \circ \cdots \circ u_d$.
Output: result $u_1 \circ u_2 \circ \cdots \circ u_d$.
repeat
  for $i = 1$ **to** d **do**
    identify the local basis functions (19), parametrized by u_k, $k \neq j$
    optimize u_i using the local basis by solving the local least squares problem
  end for
until $noChange$ is $true$

The drawback of Algorithm 1 is that the ranks of the tensor approximation have to be chosen in advance. However, there are more involved rank-adaptive versions of the ALS algorithm, providing a convenient way of finding suitable ranks. In this paper we make use of the rank-adaptive *stable alternating least-squares algorithm (SALSA)* (Grasedyck & Krämer, 2019). However, as we will see in Section 4, we can in fact oftentimes find good solutions by setting the rank to be $(1, \dots, 1) \in \mathbb{N}^{d-1}$, enabling highly efficient computations.

By straightforward extensions, adding the terminal condition g to to set of local ansatz functions can similarly be implemented into more advanced, rank adaptive ALS algorithms, which is exactly what we do for our version of SALSA.

3.2. Handling implicit regression problems

The algorithms described in the previous section require the regression problem to be explicit such as in (10). In contrast, the optimization in (11) is of implicit type, as $\widehat{h}_n$ contains the unknown $\widehat{V}_n$. In order to solve (11), we therefore choose an initial guess $\widehat{V}_n^0$ and iterate the optimization of

$$\mathbb{E}[(\widehat{V}_n^{k+1}(\widehat{X}_n) - h(\widehat{X}_n, t_n, \widehat{Y}_n^k, \widehat{Z}_n^k)\Delta t + \widehat{Z}_n^k \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1}))^2] \tag{20}$$

with respect to $\widehat{V}_n^{k+1}$ until convergence (see Appendix C for a discussion of appropriate stopping criteria). In the

above display, $\widehat{Y}_n^k = \widehat{V}_n^k(\widehat{X}_n)$ and $\widehat{Z}_n^k = (\sigma^\top \nabla \widehat{V}_n^k)(\widehat{X}_n)$ are computed according to (5). For theoretical foundation, we guarantee convergence of the proposed scheme when the step size Δt is small enough.

Theorem 3.1. *Assume that $\mathcal{U} \subset L^2(\Omega) \cap C_b^\infty(\Omega)$ is a finite dimensional linear subspace, that $\sigma(x,t)$ is nondegenerate for all $(x,t) \in [0,T] \times \mathbb{R}^d$, and that h is globally Lipschitz continuous in the last two arguments. Then there exists $\delta > 0$ such that the iteration (20) converges for all $\Delta t \in (0, \delta)$.*

Proof. See Appendix B. □

Remark 2. In order to ensure the boundedness assumption in Theorem 3.1 and to stabilize the computation we add a regularization term involving the Frobenius norm of the coefficient tensor to the objective in (20). Choosing an orthonormal basis we can then relate the Frobenius norm to the associated norm in the function space by Parseval's identity. In our numerical tests we set our one-dimensional ansatz functions to be approximately $H^2(a,b)$-orthonormal[5]. The corresponding tensor space $(H^2(a,b))^{\otimes d} = H^2_{\text{mix}}([a,b])^d$ can be shown to be continuously embedded in $W^{1,\infty}(\Omega)$, guaranteeing boundedness of the approximations and their derivatives (Sickel & Ullrich, 2009).

For convenience, we summarize the developed methods in Algorithm 3.2.

Algorithm 2 PDE approximation

Input: initial parametric choice for the functions $\widehat{V}_n$ for $n \in \{0, \dots, N-1\}$
Output: approximation of $V(\cdot, t_n) \approx \widehat{V}_n$ along the trajectories for $n \in \{0, \dots, N-1\}$
Simulate K samples of the discretized SDE (7).
Choose $\widehat{V}_N = g$.
for $n = N-1$ **to** 0 **do**
  approximate either (10) or (11) (both depending on $\widehat{V}_{n+1}$) using Monte Carlo
  minimize this quantity (explictly or by iterative schemes)
  set $\widehat{V}_n$ to be the minimizer
end for

Remark 3 (Parameter initializations)**.** Since we expect $V(\cdot, t_n)$ to be close to $V(\cdot, t_{n+1})$ for any $n \in \{0, \dots, N-1\}$, we initialize the parameters of $\widehat{V}_n^0$ as those obtained for $\widehat{V}_{n+1}$ identified in the preceding time step.

4. Numerical examples

In this section we consider some examples of high-dimensional PDEs that have been addressed in recent articles and treat them as benchmark problems in order to compare against our algorithms with respect to approximation accuracy and computation time. We refer to Appendix C for implementation details and to Appendix D for additional experiments.

[5] Here, $H^2(a,b)$ refers to the second-order Sobolev space, see (Sickel & Ullrich, 2009).

4.1. Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman equation (HJB) is a PDE for the so-called value function that represents the minimal cost-to-go in stochastic optimal control problems from which the optimal control policy can be deduced. As suggested in (E et al., 2017), we consider the HJB equation

$$(\partial_t + \Delta) V(x,t) - |\nabla V(x,t)|^2 = 0, \tag{21}$$
$$V(x,T) = g(x), \tag{22}$$

with $g(x) = \log\left(\frac{1}{2} + \frac{1}{2}|x|^2\right)$, leading to

$$b = \mathbf{0}, \quad \sigma = \sqrt{2}\,\mathrm{Id}_{d\times d}, \quad h(x,s,y,z) = -\frac{1}{2}|z|^2 \tag{23}$$

in terms of the notation established in Section 2. One appealing property of this equation is that (up to Monte Carlo approximation) a reference solution is available:

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(x+\sqrt{T-t}\sigma\xi)}\right], \tag{24}$$

where $\xi \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}_{d\times d})$ is a normally distributed random variable (see Appendix D.1 for further details).

In our experiments we consider $d = 100, T = 1, \Delta t = 0.01, x_0 = (0, \dots, 0)^\top$ and $K = 2000$ samples. In Table 1 we compare the explicit scheme stated in (10) with the implicit scheme from (11), once with TTs and once with NNs. For the tensor trains we try different polynomial degrees, and it turns out that choosing constant ansatz functions is the best choice, while fixing the rank to be 1. For the NNs we use a DenseNet like architecture with 4 hidden layers (all the details can be found in Appendices C and D).

We display the approximated solutions at $(x_0, 0)$, the corresponding relative errors $\left|\frac{\widehat{V}_n(x_0) - V_{\text{ref}}(x_0,0)}{V_{\text{ref}}(x_0,0)}\right|$ with $V_{\text{ref}}(x_0,0) = 4.589992$ being provided in (E et al., 2017), their computation times, as well as PDE and reference losses, which are specified in Appendix C. We can see that the TT approximation is both more accurate and much faster than the NN-based approaches, improving also on the results in (E et al., 2017; Beck et al., 2019). As it turns out that the explicit scheme for NNs is always worse than its implicit counterpart, but takes a very similar amount of computation time we will omit reporting it for the other experiments. In Figures 5 and 6 we plot the reference solutions computed by (24) along two trajectories of the discrete forward process (7) in dimensions $d = 10$ and $d = 100$ and compare to the implicit TT and NN-based approximations correspondingly.

	TT_{impl}	TT_{expl}	NN_{impl}	NN_{expl}
$\widehat{V}_0(x_0)$	4.5903	4.5909	4.5822	4.4961
relative error	5.90e^{-5}	3.17e^{-4}	1.71e^{-3}	2.05e^{-2}
reference loss	3.55e^{-4}	5.74e^{-4}	4.23e^{-3}	1.91e^{-2}
PDE loss	1.99e^{-3}	3.61e^{-3}	90.89	91.12
comp. time	41	25	44712	25178

Table 1. Comparison of approximation results for the HJB equation in $d = 100$.

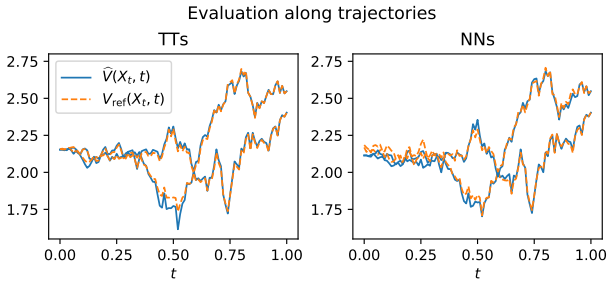


Figure 5. Reference solutions compared with implicit TT and NN approximations along two trajectories in $d = 10$.

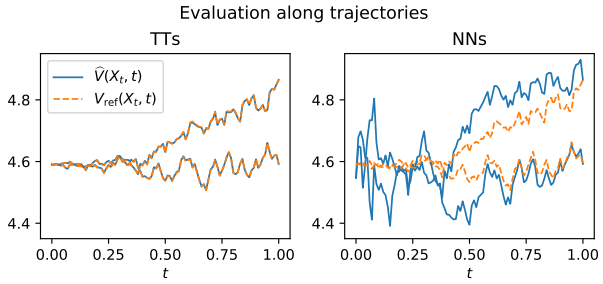


Figure 6. Reference solutions compared with implicit TT and NN approximations along two trajectories in $d = 100$.

We can see that the TT approximations perform particularly well in higher dimensions.

In Figure 7 we plot the mean relative error over time, as defined in Appendix C, indicating that both schemes are stable and where again the implicit TT scheme yields better results than the NN scheme.

Having such accurate results with only choosing constant ansatz functions in the TT approximation is surprising and we further investigate this behavior in Table 6, where we observe that the required polynomial degree decreases with increasing dimension, indicating that in this particular case the approximation problem seems to have structures that can be particularly well exploited by TTs. We argue that the black-box nature of neural networks does not reveal such properties.

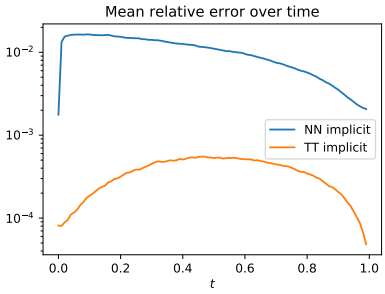


Figure 7. Mean relative error for TT and NN attempts.

d	Polynomial degree				
	0	1	2	3	4
1	3.62e^{-1}	3.60e^{-1}	2.47e^{-3}	3.86e^{-4}	4.27e^{-2}
2	1.03e^{-1}	1.02e^{-1}	1.87e^{-2}	1.79e^{-2}	1.79e^{-2}
5	1.55e^{-2}	1.54e^{-2}	1.03e^{-3}	9.52e^{-4}	1.96e^{-2}
10	2.84e^{-3}	2.86e^{-3}	1.37e^{-3}	1.34e^{-3}	1.10e^{-1}
50	1.17e^{-4}	1.29e^{-4}	2.79e^{-4}	3.35e^{-4}	6.96e^{-5}
100	5.90e^{-5}	4.99e^{-5}	8.65e^{-5}	1.23e^{-4}	3.62e^{-5}

Table 2. Relative errors of the TT approximations $\widehat{V}_n(x_0)$ for different dimensions and polynomial degrees.

4.2. HJB with double-well dynamics

In another example we consider again an HJB equation, however this time making the drift in the dynamics nonlinear, as suggested in (Nüsken & Richter, 2020). The PDE becomes

$$(\partial_t + L)\, V(x,t) - \frac{1}{2}|(\sigma^\top \nabla V)(x,t)|^2 = 0, \tag{25}$$

$$V(x,T) = g(x), \tag{26}$$

with L as in (2), where now the drift is given as the gradient of the double-well potential

$$b = -\nabla \Psi, \qquad \Psi(x) = \sum_{i,j=1}^{d} C_{ij}(x_i^2 - 1)(x_j^2 - 1) \tag{27}$$

and the terminal condition is $g(x) = \sum_{i=1}^{d} \nu_i (x_i - 1)^2$ for $\nu_i > 0$. Similarly as before a reference solution is available,

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x \right], \tag{28}$$

where X_t is the forward diffusion as specified in (4) (see again Appendix D.1 for details).

First, we consider diagonal matrices $C = 0.1\,\mathrm{Id}_{d\times d}, \sigma = \sqrt{2}\,\mathrm{Id}_{d\times d}$, implying that the dimensions do not interact, and take $T = 0.5, d = 50, \Delta t = 0.01, K = 2000, \nu_i = 0.05$. Details on the TT and NN configurations can again be found in Appendix D. Since in the solution of the PDE the dimensions do not interact either, we can compute a reference solution with finite differences. In Table 3 we see that the TT and NN approximations are compatible with TTs having an advantage in computational time. We assume that the TT result could possibly be improved by choosing a better fit of ansatz functions, as due to the local behavior of the Double-Well potential non-global ansatz functions might be a better choice.

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	9.3949	9.6942
relative error	$3.15\mathrm{e}^{-2}$	$7.27\mathrm{e}^{-4}$
reference loss	$2.15\mathrm{e}^{-2}$	$4.25\mathrm{e}^{-3}$
PDE loss	$2.43\mathrm{e}^{-2}$	$2.66\mathrm{e}^{-1}$
computation time	96	1987

Table 3. Approximation results for the HJB equation with non-interacting double well potential in $d = 50$.

Let us now consider a non-diagonal matrix $C = \mathrm{Id}_{d\times d} + (\xi_{ij})$, where $\xi_{ij} \sim \mathcal{N}(0, 0.01)$ are sampled once at the beginning of the experiment and further choose $\sigma = \sqrt{2}\,\mathrm{Id}_{d\times d}, \nu_i = 0.5, T = 0.3$. We aim at the solution at $x_0 = (-1, \dots, -1)^\top$ and compute a reference solution with (28) using 10^7 samples. We see in Table 4 that TTs are much faster than NNs, while yielding a similar performance.

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	34.278	34.228
relative error	$2.95\mathrm{e}^{-4}$	$1.20\mathrm{e}^{-3}$
reference loss	$3.26\mathrm{e}^{-2}$	$4.00\mathrm{e}^{-2}$
PDE loss	6.60	14.86
computation time	21	1693

Table 4. Approximation results for the HJB equation with interacting double well potential in $d = 20$.

4.3. Cox–Ingersoll–Ross model

Our last example is taken from financial mathematics. As suggested in (Jiang & Li, 2021) we consider a bond price in a multidimensional Cox–Ingersoll–Ross (CIR) model, see also (Hyndman, 2007; Alfonsi et al., 2015). The underlying PDE is specified as

$$\partial_t V(x,t) + \frac{1}{2}\sum_{i,j=1}^{d} \sqrt{x_i x_j}\gamma_i\gamma_j \partial_{x_i}\partial_{x_j} V(x,t) + \sum_{i=1}^{d} a_i(b_i - x_i)\partial_{x_i} V(x,t) - \left(\max_{1\le i\le d} x_i\right) V(x,t) = 0. \tag{29}$$

Here, $a_i, b_i, \gamma_i \in [0,1]$ are uniformly sampled at the beginning of the experiment and $V(T,x) = 1$. We set $d = 100$.

We aim to estimate the bond price at the initial condition $x_0 = (1, \dots, 1)^\top$. As there is no reference solution known, we rely on the PDE loss to compare our results. Table 5 shows that all three approaches yield similar results, while having a rather small PDE loss. The TT approximations seem to be slightly better and we note that the explicit TT scheme is again much faster.

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{TT}_{\mathrm{expl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	0.312	0.306	0.31087
PDE loss	$5.06\mathrm{e}^{-4}$	$5.04\mathrm{e}^{-4}$	$7.57\mathrm{e}^{-3}$
computation time	5281	197	9573

Table 5. $K = 1000, d = 100, x_0 = [1, 1, \dots, 1]$

In Table 6 we compare the PDE loss using different polynomial degrees for the TT ansatz function and see that we do not get any improvements with polynomials of degree larger than 1.

	Polynom. degree			
	0	1	2	3
$\widehat{V}_0(x_0)$	0.294	0.312	0.312	0.312
PDE loss	$9.04\mathrm{e}^{-2}$	$7.80\mathrm{e}^{-4}$	$1.05\mathrm{e}^{-3}$	$5.06\mathrm{e}^{-4}$
computation time	110	3609	4219	5281

Table 6. PDE loss and computation time for TTs with different polynomial degrees

Noticing the similarity between the results for polynomial degrees 1, 2, and 3, we further investigate by computing the value function along a sample trajectory in Figure 8, where we see that indeed the approximations with those polynomial degrees are indistinguishable.

Acknowledgements This research has been partially funded by Deutsche Forschungsgemeinschaft (DFG) through the grant CRC 1114 'Scaling Cascades in Complex Systems' (projects A02 and A05, project number 235221301). Leon Sallandt acknowledges support from the Research Training Group 'Differential Equation- and Data-driven Models in Life Sciences and Fluid Dynamics: An Interdisciplinary Research Training Group

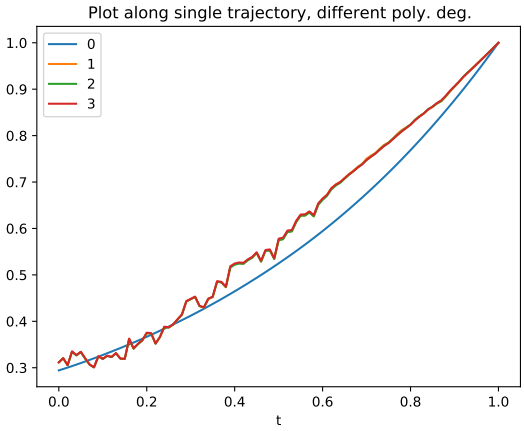


Figure 8. Reference trajectory for different polynomial degrees.

(DAEDALUS)'(GRK 2433) funded by the German Research Foundation (DFG). We would like to thank Reinhold Schneider for giving valuable input and for sharing his broad insight in tensor methods and optimization.

References

Abdelfattah, A., Baboulin, M., Dobrev, V., Dongarra, J., Earl, C., Falcou, J., Haidar, A., Karlin, I., Kolev, T., Masliah, I., et al. High-performance tensor contractions for GPUs. *Procedia Computer Science*, 80:108–118, 2016.

Alfonsi, A. et al. *Affine diffusions and related processes: simulation, theory and applications*, volume 6. Springer, 2015.

Bachmayr, M., Schneider, R., and Uschmajew, A. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Found. Comput. Math.*, 16(6):1423–1472, December 2016. ISSN 1615-3375. doi: 10.1007/s10208-016-9317-9. URL https://doi.org/10.1007/s10208-016-9317-9.

Beck, C., Becker, S., Cheridito, P., Jentzen, A., and Neufeld, A. Deep splitting method for parabolic PDEs. *arXiv preprint arXiv:1907.03452*, 2019.

Bismut, J.-M. Conjugate convex functions in optimal stochastic control. *Journal of Mathematical Analysis and Applications*, 44(2):384–404, 1973.

Bouchard, B. and Touzi, N. Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their applications*, 111(2):175–206, 2004.

Dolgov, S., Kalise, D., and Kunisch, K. Tensor decompositions for high-dimensional Hamilton-Jacobi-Bellman equations. *arXiv preprint arXiv:1908.01533*, 2019.

Dolgov, S. V., Khoromskij, B. N., and Oseledets, I. V. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

E, W. and Yu, B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6 (1):1–12, 2018.

E, W., Han, J., and Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

E, W., Hutzenthaler, M., Jentzen, A., and Kruse, T. On multilevel picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. *Journal of Scientific Computing*, 79(3):1534–1571, 2019.

Fackeldey, K., Oster, M., Sallandt, L., and Schneider, R. Approximative policy iteration for exit time feedback control problems driven by stochastic differential equations using tensor train format. *arXiv preprint arXiv:2010.04465*, 2020.

Fleming, W. H. and Rishel, R. W. *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media, 2012.

Fleming, W. H. and Soner, H. M. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

Gobet, E. *Monte-Carlo methods and stochastic processes: from linear to non-linear*. CRC Press, 2016.

Gobet, E., Lemor, J.-P., Warin, X., et al. A regression-based Monte Carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability*, 15(3):2172–2202, 2005.

Grasedyck, L. and Krämer, S. Stable als approximation in the tt-format for rank-adaptive tensor completion. *Numerische Mathematik*, 143(4):855–904, 2019.

Hackbusch, W. Numerical tensor calculus. *Acta numerica*, 23:651–742, 2014. ISSN 1474-0508. doi: 10.1017/S0962492914000087.

Hackbusch, W. and Kühn, S. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009. ISSN 1069-5869. doi: 10.1007/s00041-009-9094-9. URL http://dx.doi.org/10.1007/s00041-009-9094-9.

Hackbusch, W. and Schneider, R. *Tensor Spaces and Hierarchical Tensor Representations*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08159-5. doi: 10.1007/978-3-319-08159-5_12. URL https://doi.org/10.1007/978-3-319-08159-5_12.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Hartmann, C. and Richter, L. Nonasymptotic bounds for suboptimal importance sampling. *arXiv preprint arXiv:2102.09606*, 2021.

Hartmann, C., Richter, L., Schütte, C., and Zhang, W. Variational characterization of free energy: Theory and algorithms. *Entropy*, 19(11):626, 2017.

Hartmann, C., Kebiri, O., Neureither, L., and Richter, L. Variational approach to rare event simulation using least-squares regression. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(6):063107, 2019.

Holtz, S., Rohwedder, T., and Schneider, R. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012a. doi: 10.1137/100818893. URL https://doi.org/10.1137/100818893.

Holtz, S., Rohwedder, T., and Schneider, R. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120 (4):701–731, 2012b.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Huber, B. and Wolf, S. Xerus - a general purpose tensor library. https://libxerus.org/, 2014–2017.

Huré, C., Pham, H., and Warin, X. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 89(324):1547–1579, 2020.

Hyndman, C. B. Forward-backward SDEs and the CIR model. *Statistics & probability letters*, 77(17):1676–1682, 2007.

Jentzen, A., Salimova, D., and Welti, T. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

Jiang, Y. and Li, J. Convergence of the deep BSDE method for FBSDEs with non-lipschitz coefficients. *arXiv preprint arXiv:2101.01869*, 2021.

Karatzas, I. and Shreve, S. E. *Brownian Motion and Stochastic Calculus*. Springer, 1998.

Kazeev, V., Oseledets, I., Rakhuba, M., and Schwab, C. QTT-finite-element approximation for multiscale problems. Tech. Report 2016-06, Seminar for Applied Mathematics, ETH Zürich, 2016 . . . , 2016.

Kazeev, V. A. and Khoromskij, B. N. Low-rank explicit QTT representation of the laplace operator and its inverse. *SIAM journal on matrix analysis and applications*, 33(3): 742–758, 2012.

Khoromskij, B. N. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1–19, 2012.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Kutschan, B. Tangent cones to tensor train varieties. *Linear Algebra and its Applications*, 544:370–390, 2018.

Longstaff, F. A. and Schwartz, E. S. Valuing American options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

Nüsken, N. and Richter, L. Solving high-dimensional Hamilton-Jacobi-Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *arXiv preprint arXiv:2005.05409*, 2020.

Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Oseledets, I. V. and Tyrtyshnikov, E. E. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5): 3744–3759, 2009.

Oster, M., Sallandt, L., and Schneider, R. Approximating the stationary Hamilton-Jacobi-Bellman equation by hierarchical tensor products. *arXiv preprint arXiv:1911.00279*, 2019.

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

Pardoux, E. and Peng, S. Adapted solution of a backward stochastic differential equation. *Systems & Control Letters*, 14(1):55–61, 1990.

Pham, H. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer Science & Business Media, 2009.

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

Sickel, W. and Ullrich, T. Tensor products of Sobolev-Besov spaces and applications to approximation from the hyperbolic cross. *Journal of Approximation Theory*, 161 (2):748–786, 2009.

Sirignano, J. and Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

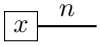
Szalay, S., Pfeffer, M., Murg, V., Barcza, G., Verstraete, F., Schneider, R., and Örs Legeza. Tensor product methods and entanglement optimization for ab initio quantum chemistry. *International j. of quantum chemistry*, 115(19):1342–1391, 2015. ISSN 1097-461x. doi: 10.1002/qua.24898.

Zhang, J. A numerical scheme for BSDEs. *The annals of applied probability*, 14(1):459–488, 2004.

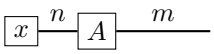
Zhang, J. *Backward stochastic differential equations*. Springer, 2017.

A. Graphical notation for tensor trains

In this section we provide some further material on tensor networks and their graphic notation. Let us start by noting that a vector $x \in \mathbb{R}^n$ can be interpreted as a tensor.



In the graphic representation contractions between indices are denoted by a line between the tensors. Below we contract a tensor $A \in \mathbb{R}^{n\times m}$ and $x \in \mathbb{R}^n$, which results in an element of $\mathbb{R}^m$, representing the usual matrix-vector product.



In Figure 9 an order 3 tensor $B \in \mathbb{R}^{n_1\times n_2\times n_3}$ is represented with three lines, not connected to any other tensor. As

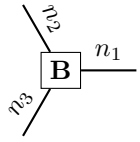


Figure 9. Graphical notation of simple tensors and tensor networks

another example, we can write the compact singular value decomposition in matrix form as $A = U\Sigma V$, with $U \in \mathbb{R}^{n,r}, \Sigma \in \mathbb{R}^{r,r}, V \in \mathbb{R}^{r,m}$, which we represent as a tensor network in Figure 10.

A.1. The local basis functions

We elaborate in detail on the local basis functions $b^{\mathrm{loc},i} = [b_1^{\mathrm{loc},i}, \dots, b_m^{\mathrm{loc},i}]$, as indicated in Figure 11. Assuming that we optimize u_2, we notice that the tensor $\phi(x_1) \circ u_1$ is actually a function mapping from $\mathbb{R} \to \mathbb{R}^{r_1}$, which means that we can identify r_1 many one-dimensional functions. Note that this corresponds to the left part of the tensor picture in Figure 11. Further, we have that $\phi(x_2)$ is a vector consisting of m one-dimensional functions, which is the middle part of the above tensor picture. The right part, consisting of the contractions between $\phi(x_2)$, u_3, u_4, and $\phi(x_4)$, is a set of two-dimensional functions with cardinality r_2. Taking the tensor product of the above functions yields an $r_1 m r_2$ dimensional function space of four-dimensional functions, which is exactly the span of the local basis functions.

More precisely, when optimizing the k-th component tensor, the local basis functions are given by setting $j_{k-1} \le r_{k-1}$, $i_k \le m$, and $j_k \le r_k$ within the following formula:

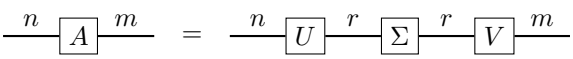


Figure 10. Graphical notation of simple tensors and tensor networks.

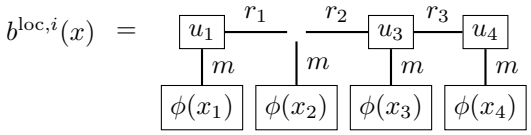


Figure 11. Graphical representation of the local basis functions for $i = 2$.

$$
\begin{aligned}
&b_{j_{k-1},i_k,j_k}(x) = \\
&\Bigg(\sum_{i_1,\dots,i_{k-1}}^{m,\dots,m}\ \sum_{j_1,\dots,j_{k-2}}^{r_1,\dots,r_{k-2}} u_1[i_1,j_1]\dots u_{k-1}[j_{k-2},i_{k-1},j_{k-1}] \\
&\qquad \phi(x_1)[i_1]\dots\phi(x_{k-1})[i_{k-1}]\Bigg)\phi(x_k)[i_k] \\
&\Bigg(\sum_{i_{k+1},\dots,i_d}^{m,\dots,m}\ \sum_{j_k,\dots,j_{d-1}}^{r_k,\dots,r_{d-1}} u_{k+1}[j_k,i_{k+1},j_{k+1}]\dots u_d[j_{d-1},i_d] \\
&\qquad \phi(x_{k+1})[i_{k+1}]\dots\phi(x_d)[i_d]\Bigg).
\end{aligned}
\tag{30}
$$

Note that in the above formula, every index except j_{k-1}, i_k and j_k is contracted, leaving an order three tensor. A simple reshape into one index then yields the local basis functions as used in this paper.

B. Proof of Theorem 3.1

Proof of Theorem 3.1. In this proof, we denote the underlying probability measure by $\mathbb{P}$, and the corresponding Hilbert space of random variables with finite second moments by $L^2(\mathbb{P})$. We define the linear subspace $\widetilde{\mathcal{U}} \subset L^2(\mathbb{P})$ by

$$
\widetilde{\mathcal{U}} = \left\{ f(\widehat{X}_n) : f \in \mathcal{U} \right\}, \tag{31}
$$

noting that $\widetilde{\mathcal{U}}$ is finite-dimensional by the assumption on $\mathcal{U}$, hence closed. The corresponding $L^2(\mathbb{P})$-orthogonal projection onto $\widetilde{\mathcal{U}}$ will be denoted by $\Pi_{\widetilde{\mathcal{U}}}$. By the nondegeneracy of σ, the law of $\widehat{X}_n$ has full support on Ω, and so $\|\cdot\|_{L^2(\mathbb{P})}$ is indeed a norm on $\widetilde{\mathcal{U}}$. Since $\widetilde{\mathcal{U}}$ is finite-dimensional, the linear operators

$$
\widetilde{\mathcal{U}} \ni f(\widehat{X}_n) \mapsto \frac{\partial f}{\partial x_i}(\widehat{X}_n) \in L^2(\mathbb{P}) \tag{32}
$$

are bounded, and consequently there exists a constant $C_1 > 0$ such that

$$\left\| \frac{\partial f}{\partial x_i}(\widehat{X}_n) \right\|_{L^2(\mathbb{P})} \leq C_1 \left\| f(\widehat{X}_n) \right\|_{L^2(\mathbb{P})}, \tag{33}$$

for all $i = 1, \dots, d$ and $f \in \mathcal{U}$. Furthermore, there exists a constant $C_2 > 0$ such that

$$\mathbb{E}\left[f^4(\widehat{X}_n) \right]^{1/4} := \left\| f(\widehat{X}_N) \right\|_{L^4(\mathbb{P})} \leq C_2 \left\| f(\widehat{X}_n) \right\|_{L^2(\mathbb{P})},$$

for all $f \in \mathcal{U}$, again by the finite-dimensionality of $\widetilde{\mathcal{U}}$ and the fact that on finite dimensional vector spaces, all norms are equivalent. By standard results on orthogonal projections, the solution to the iteration (20) is given by

$$V_n^{k+1}(\widehat{X}_n) = \Pi_{\widetilde{\mathcal{U}}}\big[- h(\widehat{X}_n, t_n, \widehat{Y}_n^k, \widehat{Z}_n^k)\Delta t + \widehat{Z}_n^k \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1}) \big].$$

We now consider the map $\Psi : \widetilde{\mathcal{U}} \to \widetilde{\mathcal{U}}$ defined by

$$f(\widehat{X}_n) \mapsto \Pi_{\widetilde{\mathcal{U}}}\big[- h(\widehat{X}_n, t_n, f(\widehat{X}_n), \sigma^\top \nabla f(\widehat{X}_n))\Delta t + \sigma^\top \nabla f(\widehat{X}_n) \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1}) \big].$$

For $F_1, F_2 \in \widetilde{\mathcal{U}}$ with $F_i = f_i(\widehat{X}_n)$, $f_i \in \mathcal{U}$, we see that

$$\begin{aligned}
&\|\Psi F_1 - \Psi F_2\|_{L^2(\mathbb{P})} \\
&= \big\| \Pi_{\widetilde{\mathcal{U}}}\big[- h(\widehat{X}_n, t_n, f_1(\widehat{X}_n), \sigma^\top \nabla f_1(\widehat{X}_n))\Delta t \\
&+ h(\widehat{X}_n, t_n, f_2(\widehat{X}_n), \sigma^\top \nabla f_2(\widehat{X}_n))\Delta t \\
&+ \sqrt{\Delta t}\left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n) \right) \cdot \xi_{n+1} \big] \big\|_{L^2(\mathbb{P})} \\
&\leq C_3 \left\| \Pi_{\widetilde{\mathcal{U}}} \right\|_{L^2(\mathbb{P}) \to L^2(\mathbb{P})} \Bigg(\Delta t \|F_1 - F_2\|_{L^2(\mathbb{P})} \\
&+ \sqrt{\Delta t} \left\| \left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n) \right) \cdot \xi_{n+1} \right\|_{L^2(\mathbb{P})} \Bigg)
\end{aligned}$$

for some constant C_3 that does not depend on Δt, where we have used the triangle inequality, the Lipschitz assumption on h, the boundedness of σ, and the estimate (33). Using the Cauchy-Schwarz inequality, boundedness of σ as well as (33) and (34), the last term can be estimated as follows,

$$\begin{aligned}
&\left\| \left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n) \right) \cdot \xi_{n+1} \right\|_{L^2(\mathbb{P})} \\
&\leq \left\| \left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n) \right)^2 \right\|_{L^2(\mathbb{P})}^{1/2} \left\| \xi_{n+1}^2 \right\|_{L^2(\mathbb{P})}^{1/2} \\
&\leq C_4 \|F_1 - F_2\|_{L^2(\mathbb{P})},
\end{aligned}$$

where C_4 is a constant independent of Δt. Collecting the previous estimates, we see that $\delta > 0$ can be chosen such that for all $t \in (0, \delta)$, the mapping Ψ is a contraction on $\widetilde{\mathcal{U}}$ when equipped with the norm $\|\cdot\|_{L^2(\mathbb{P})}$, that is,

$$\|\Psi F_1 - \Psi F_2\| \leq \lambda \|F_1 - F_2\|, \tag{37}$$

for some $\lambda < 1$ and all $F_1, F_2 \in \widetilde{\mathcal{U}}$. Finally, the statement follows from the Banach fixed point theorem. □

C. Implementation details

For the evaluation of our approximations we rely on reference values of $V(x_0, 0)$ and further define the following two loss metrics, which are zero if and only if the PDE is fulfilled along the samples generated by the discrete forward SDE (7). In the spirit of (Raissi et al., 2019), we define the *PDE loss* as

$$\begin{aligned}
\mathcal{L}_{\text{PDE}} = \frac{1}{KN} \sum_{n=1}^{N} \sum_{k=1}^{K} \Big(&(\partial_t + L) V(\widehat{X}_n^{(k)}, t_n) \qquad (38) \\
&+ h(\widehat{X}_n^{(k)}, t_n, V(\widehat{X}_n^{(k)}, t_n), (\sigma^\top \nabla V)(\widehat{X}_n^{(k)}, t_n)) \Big)^2, \qquad (39)
\end{aligned}$$

where $\widehat{X}_n^{(k)}$ are realizations of (7), the time derivative is approximated with finite differences and the space derivatives are computed analytically (or with automatic differentiation tools). We leave out the first time step $n = 0$ since the regression problem within the explicit and the implicit schemes for the tensor trains are not well-defined due to the fact that $\widehat{X}_0^k = x_0$ has the same value for all k. We still obtain a good approximation since the added regularization term brings a minimum norm solution with the correct point value $V(x_0, 0)$. Still, this does not aim at the PDE being entirely fulfilled at this point in time.

Further, we define the *relative reference loss* as

$$\mathcal{L}_{\text{ref}} = \frac{1}{K(N+1)} \sum_{n=0}^{N} \sum_{k=1}^{K} \left| \frac{V(\widehat{X}_n^{(k)}, t_n) - V_{\text{ref}}(\widehat{X}_n^{(k)}, t_n)}{V_{\text{ref}}(\widehat{X}_n^{(k)}, t_n)} \right|, \tag{40}$$

whenever a reference solution for all x and t is available.

All computation times in the reported tables are measured in seconds.

Our experiments have been performed on a desktop computer containing an AMD Ryzen Threadripper 2990 WX 32x 3.00 GHz mainboard and an NVIDIA Titan RTX GPU, where we note that only the NN optimizations were run on this GPU, since our TT framework does not include GPU support. It is expected that running the TT approximations on a GPU will improve time performances in the future (Abdelfattah et al., 2016).

All our code is available under https://github.com/lorenzrichter/PDE-backward-solver.

C.1. Details on neural network approximation

For the neural network architecture we rely on the *DenseNet*, which consists of fully-connected layers with additional skip connections as for instance suggested in (E & Yu, 2018) and being rooted in (Huang et al., 2017). To be precise, we define a version of the *DenseNet* that includes the terminal condition of the PDE (1) as an additive extension by

$$\Phi_\varrho(x) = A_L x_L + b_L + \theta g(x), \tag{41}$$

where x_L is specified recursively as

$$y_{l+1} = \varrho(A_l x_l + b_l), \qquad x_{l+1} = (x_l, y_{l+1})^\top \tag{42}$$

for $1 \leq l \leq L-1$ with $A_l \in \mathbb{R}^{r_l \times \sum_{i=0}^{l-1} r_i}, b_l \in \mathbb{R}^l, \theta \in \mathbb{R}$ and $x_1 = x$. The collection of matrices A_l, vectors b_l and the coefficient θ comprises the learnable parameters, and we introduce the vector $r := (d_{\text{in}}, r_1, \dots, r_{L-1}, d_{\text{out}})$ to represent a certain choice of a DenseNet architecture, where in our setting $d_{\text{in}} = d$ and $d_{\text{out}} = 1$. If not otherwise stated we fix the parameter θ to be 1. For the activation function $\varrho : \mathbb{R} \to \mathbb{R}$, that is to be applied componentwise, we choose $\tanh$.

For the gradient descent optimization we choose the Adam optimizer with the default parameters $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ (Kingma & Ba, 2014). In most of our experiments we chose a fixed learning rate η_{N-1} for the approximation of the first backward iteration step to approximate $\widehat{V}_{N-1}$ and another fixed learning rate η_n for all the other iteration steps to approximate $\widehat{V}_n$ for $0 \leq n \leq N-2$ (cf. Remark 3). Similarly, we denote with G_{N-1} and G_n the amount of gradient descent steps in the corresponding optimizations.

In Tables 7 and 8 we list our hyperparameter choices for the neural network experiments that we have conducted.

C.2. Details on tensor train approximation

For the implementation of the tensor networks we rely on the C++ library *xerus* (Huber & Wolf, 2014–2017) and the Python library *numpy* (Harris et al., 2020).

Within the optimization we have to specify the regularization parameter as noted in Remark 2, which we denot here by $\eta > 0$. We adapt this parameter in dependence of the current residual in the regression problem (20), i.e. $\eta = cw$, where $c > 0$ and w is the residual from the previous sweep of SALSA. In every all our experiments we set $c_\eta = 1$. Further, we have to specify the condition "*noChange* is *true*" within Algorithm 1. To this end we introduce a test set with equal size as our training set. We measure the residual within a single run of SALSA on the test set and the training set. If the change of the residual on either of this sets is below $\delta = 0.0001$ we set *noChange = true*. For the fixed-point iteration we have a two-fold stopping condition. We stop

HJB, $d = 10$, NN_{impl} Figure 5
$K = 2000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 8000, G_{N-1} = 40000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0001$

HJB, $d = 100$, NN_{impl} Table 1, Figures 6, 7
$K = 2000, \Delta t = 0.01$
$r = (100, 130, 130, 70, 70, 1)$
$G_n = 5000, G_{N-1} = 40000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0003$

HJB, $d = 100$, NN_{expl} Table 1, Figures 6, 7
$K = 2000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 500, G_{N-1} = 7000$
$\eta_n = 0.00005, \eta_{N-1} = 0.0003$

HJB double well $d = 50$, NN_{impl}, Table 3
$K = 2000, \Delta t = 0.01$
$r = (50, 30, 30, 1)$
$G_n = 2000, G_{N-1} = 25000$
$\eta_n = 0.0002, \eta_{N-1} = 0.0005$

HJB interacting double well $d = 20$, NN_{impl}, Table 4
$K = 2000, \Delta t = 0.01$
$r = (50, 20, 20, 20, 20, 1)$
$G_n = 3000, G_{N-1} = 30000$
$\eta_n = 0.0007, \eta_{N-1} = 0.001$

CIR, $d = 100$, NN_{impl} Table 5
$K = 1000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 2000$ for $0 \leq n \leq 15$
$G_n = 300$ for $16 \leq n \leq N-2$
$G_{N-1} = 10000$
$\eta_n = 0.00005, \eta_{N-1} = 0.0001$

Table 7. Neural network hyperparameters for the experiments in paper.

PDE with unbounded solution $d = 10$, NN_{impl}, Table 9
$K = 1000, \Delta t = 0.001$
$r = (10, 30, 30, 1)$
$G_n = 100, G_{N-1} = 10000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0001$

Allen-Cahn $d = 100$, NN_{impl}, Table 10
$K = 8000, \Delta t = 0.01$
$r = (10, 30, 30, 1)$
$G_n = 10000$ for $0 \leq n \leq 5$
$G_n = 6000$ for $6 \leq n \leq N-2$
$G_{N-1} = 15000$
$\eta_n = 0.0002, \eta_{N-1} = 0.001$

Table 8. Neural network hyperparameters for the additional experiments.

the iteration if either the Frobenius norm of the coefficients has a smaller relative difference than $\gamma_1 < 0.0001$ or if the values $\widehat{V}_n^{k+1}$ and $\widehat{V}_n^k$ and their gradients, evaluated at the points of the test set, have a relative difference smaller than $\gamma_2 < 0.00001$. Note that the second condition is essentially a discrete H^1 norm, which is necessary since by adding the final condition into the ansatz space the orthonormal basis property is violated.

D. Further numerical examples

In this section we elaborate on some of the numerical examples from the paper and provide two additional problems.

D.1. Hamilton-Jacobi-Bellman equation

Let us consider the HJB equation from Sections 4.1 and 4.2, which we can write as

$$(\partial_t + L)\, V(x,t) - \frac{1}{2}|(\sigma^\top \nabla V)(x,t)|^2 = 0, \tag{43a}$$

$$V(x,T) = g(x), \tag{43b}$$

in a generic form with the differential operator L being defined in (2). We can introduce the exponential transformation $\psi := e^{-V}$ and with the chain rule find that the transformed function fulfills the linear PDE

$$(\partial_t + L)\, \psi(x,t) = 0, \tag{44a}$$

$$\psi(x,T) = e^{-g(x)}. \tag{44b}$$

This is known as Hopf-Cole transformation, see also (Fleming & Soner, 2006; Hartmann et al., 2017). It is known that via the Feynman-Kac theorem (Karatzas & Shreve, 1998) the solution to this PDE has the stochastic representation

$$\psi(x,t) = \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x \right], \tag{45}$$

such that we readily get

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x \right], \tag{46}$$

which we can use as a reference solution by approximating the expectation value via Monte Carlo simulation, however keeping in mind that in high dimensions corresponding estimators might have high variances (Hartmann & Richter, 2021).

Let us stress again that our algorithms only aim to provide a solution of the PDE along the trajectories of the forward process (4). Still, there is hope that our approximations generalize to regions "close" to where samples are available. To illustrate this, consider for instance the d-dimensional forward process

$$X_s = x_0 + \sigma W_s, \tag{47}$$

as for instance in Section 4.1, where now $\sigma > 0$ is one-dimensional for notational convenience. We know that $X_t \sim \mathcal{N}(x_0, \sigma^2 t\, \mathrm{Id}_{d\times d})$ and therefore note that for the expected distance to the origin it holds

$$\mathbb{E}\left[|X_t - x_0|\right] < \sqrt{\mathbb{E}\left[|X_t - x_0|^2\right]} = \sigma\sqrt{dt}. \tag{48}$$

This motivates evaluating the approximations along the curve

$$X_t = x_0 + \sigma\sqrt{t}\mathbf{1}, \tag{49}$$

where $\mathbf{1} = (1, \ldots, 1)^\top$. Figure 12 shows that in this case we indeed have good agreement of the approximation with the reference solution when using TTs and that for NNs the deep neural network that we have specified in Table 7 generalizes worse than a shallower network with only two hidden layers consisting of 30 neurons each.

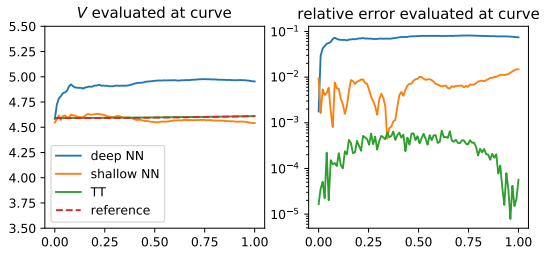


Figure 12. Approximations of the HJB equation in $d = 100$ evaluated along a representative curve.

D.2. PDE with unbounded solution

As an additional problem, we choose an example from (Huré et al., 2020) which offers an analytical reference solution.

For the PDE as defined in (1) we consider the coefficients

$$b(x,t) = \mathbf{0},\ \sigma(x,t) = \frac{\mathrm{Id}_{d\times d}}{\sqrt{d}},\ g(x) = \cos\left(\sum_{i=1}^{d} i x_i\right), \tag{50}$$

$$h(x,t,y,z) = k(x) + \frac{y}{2\sqrt{d}}\sum_{i=1}^{d} z_i + \frac{y^2}{2}, \tag{51}$$

where, with an appropriately chosen k, a solution can shown to be

$$V(x,t) = \frac{T-t}{d}\sum_{i=1}^{d}\left(\sin(x_i)\mathbb{1}_{x_i<0} + x_i\mathbb{1}_{x_i\geq 0}\right) + \cos\left(\sum_{i=1}^{d} i x_i\right). \tag{52}$$

In Table 9 we compare the results for $d = 10, K = 1000, T = 1, \Delta t = 0.001, x_0 = (0.5,\dots,0.5)^\top$. For the TT case it was sufficient to set the ranks to 1 and we see that the results are improved significantly if we increase the sample size K from 1000 to 20000. Note that even when increasing the sample size by a factor 20, the computational time is still lower than the NN implementation. It should be highlighted that adding the function g to the neural network (as explained in Appendix C) is essential for its convergence in higher dimensions and thereby mitigates the observed difficulties in (Huré et al., 2020)).

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{TT}^*_{\mathrm{impl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	-0.1887	-0.2136	-0.2137
relative error	$1.22\mathrm{e}^{-1}$	$6.11\mathrm{e}^{-3}$	$5.50\mathrm{e}^{-3}$
ref loss	$2.47\mathrm{e}^{-1}$	$7.57\mathrm{e}^{-2}$	$3.05\mathrm{e}^{-1}$
abs. ref loss	$2.52\mathrm{e}^{-2}$	$9.29\mathrm{e}^{-3}$	$1.69\mathrm{e}^{-2}$
PDE loss	2.42	0.60	1.38
computation time	360	1778	4520

Table 9. Approximation results for the PDE with an unbounded analytic solution. For $\mathrm{TT}^*_{\mathrm{impl}}$ we choose $K = 20000$, for the others we choose $K = 1000$.

D.3. Allen-Cahn like equation

Finally, let us consider the following Allen-Cahn like PDE with a cubic nonlinearity in $d = 100$:

$$(\partial_t + \Delta)V(x,t) + V(x,t) - V^3(x,t) = 0, \tag{53a}$$

$$V(x,T) = g(x), \tag{53b}$$

where we choose $g(x) = \left(2 + \frac{2}{5}|x|^2\right)^{-1}$, $T = \frac{3}{10}$ and are interested in an evaluation at $x_0 = (0,\dots,0)^\top$. This problem has been considered in (E et al., 2017), where a reference solution of $V(x_0,0) = 0.052802$ calculated by means of the branching diffusion method is provided. We consider a sample size of $K = 1000$ and a stepsize $\Delta t = 0.01$ and provide our approximation results in Table 10.

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{TT}_{\mathrm{expl}}$	$\mathrm{NN}_{\mathrm{impl}}$	$\mathrm{NN}^*_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	0.052800	0.05256	0.04678	0.05176
relative error	$4.75\mathrm{e}^{-5}$	$4.65\mathrm{e}^{-3}$	$1.14\mathrm{e}^{-1}$	$1.97\mathrm{e}^{-2}$
PDE loss	$2.40\mathrm{e}^{-4}$	$2.57\mathrm{e}^{-4}$	$9.08\mathrm{e}^{-1}$	$6.92\mathrm{e}^{-1}$
comp. time	24	10	23010	95278

Table 10. Approximations for Allen-Cahn PDE, where $\mathrm{NN}^*_{\mathrm{impl}}$ uses $K = 8000$ and the others $K = 1000$ samples.

E. Some background on BSDEs and their numerical discretizations

BSDEs have been studied extensively in the last three decades and we refer to (Pardoux, 1998; Pham, 2009; Gobet, 2016; Zhang, 2017) for good introductions to the topic. Let us note that given some assumptions on the coefficients b, σ, h and g one can prove existence and uniqueness of a solution to the BSDE system as defined in (4) and (6), see for instance Theorem 4.3.1 in (Zhang, 2017).

We note that the standard BSDE system can be generalized to

$$\mathrm{d}X_s = (b(X_s,s) + v(X_s,s))\,\mathrm{d}s + \sigma(X_s,s)\mathrm{d}W_s, \tag{54a}$$

$$X_0 = x, \tag{54b}$$

$$\mathrm{d}Y_s = (-h(X_s,s,Y_s,Z_s) + v(X_s,s)\cdot Z_s)\mathrm{d}s + Z_s\cdot\mathrm{d}W_s, \tag{54c}$$

$$Y_T = g(X_T), \tag{54d}$$

where $v : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ is any suitable control vector field that can be understood as pushing the forward trajectories into desired regions of the state space, noting that the relations

$$Y_s = V(X_s,s), \qquad Z_s = (\sigma^\top \nabla V)(X_s,s), \tag{55}$$

with $V : \mathbb{R}^d \times [0,T] \to \mathbb{R}$ being the solution to the parabolic PDE (1), hold true independent of the choice of v (Hartmann et al., 2019). Our algorithms readily transfer to this change in sampling the forward process by adapting the backward process and the corresponding loss functionals (10) and (11) accordingly.

In order to understand the different numerical discretization schemes in Section 2.1, let us note that we can write the backward process (5) in its integrated form for the times $t_n < t_{n+1}$ as

$$Y_{t_{n+1}} = Y_{t_n} - \int_{t_n}^{t_{n+1}} h(X_s,s,Y_s,Z_s)\mathrm{d}s + \int_{t_n}^{t_{n+1}} Z_s\cdot\mathrm{d}W_s. \tag{56}$$

In a discrete version we have to replace the integrals with suitable discretizations, where for the deterministic integral we can decide which endpoint to consider, leading to either of the following two discretization schemes

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_n \Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}, \tag{57a}$$

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_{n+1} \Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}, \tag{57b}$$

as defined in (8), where we recall the shorthands

$$h_n = h(\widehat{X}_n, t_n, \widehat{Y}_n, \widehat{Z}_n), \tag{58a}$$

$$h_{n+1} = h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1}). \tag{58b}$$

The L^2-projection scheme (10) can be motivated as follows. Consider the explicit discrete backward scheme as in (57b)

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1}) \Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}. \tag{59}$$

Taking conditional expectations w.r.t. to the σ-algebra generated by the discrete Brownian motion at time step n, denoted by $\mathcal{F}_n$, yields

$$\widehat{Y}_n = \mathbb{E}\left[\widehat{Y}_{n+1} + h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1}) \Delta t \middle| \mathcal{F}_n \right]. \tag{60}$$

We can now recall that a conditional expectation can be characterized as a best approximation in L^2, namely

$$\mathbb{E}[B|\mathcal{F}_n] = \underset{\substack{Y \in L^2 \\ \mathcal{F}_n - \text{measurable}}}{\arg\min} \; \mathbb{E}\left[|Y - B|^2 \right], \tag{61}$$

for any random variable $B \in L^2$, which brings

$$\widehat{Y}_n = \underset{\substack{Y \in L^2 \\ \mathcal{F}_n - \text{measurable}}}{\arg\min} \; \mathbb{E}\left[\left(Y - h_{n+1} \Delta t - \widehat{Y}_{n+1} \right)^2 \right]. \tag{62}$$

This then yields the explicit scheme depicted in (10). We refer once more to (Gobet et al., 2005) for extensive numerical analysis, essentially showing that the proposed scheme is of order $\frac{1}{2}$ in the time step Δt.