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

[*]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>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

scheme is implicit and involves a nested iterative procedure, 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 parametric PDEs, see e.g. (Dolgov et al., 2015; Eigel et al., 2017; Dektor et al., 2020), Hamilton-Jacobi-Bellman PDEs (Horowitz et al., 2014; Stefansson & Leong, 2016; Gorodetsky et al., 2018; Dolgov et al., 2019; Oster et al., 2019; Fackeldey et al., 2020; Chen & Lu, 2021), and PDEs of other types, see e.g. (Khoromskij, 2012; Kormann, 2015; Lubasch et al., 2018).

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

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

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 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 $\widehat{V}_n(\widehat{X}_n) \approx \widehat{Y}_n \approx V(\widehat{X}_n, t_n)$ to be learnt in the update step $n+1 \to n$ which can either be based on (8a) or on (8b):

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

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

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

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 $c \in \mathbb{R}^{m\times m\times\cdots\times m} \equiv \mathbb{R}^{m^d}$ 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.

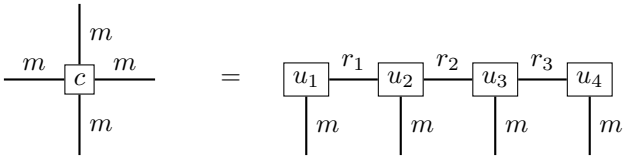


Figure 1. An order 4 tensor and a tensor train representation.

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 (Landsberg, 2012; 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:

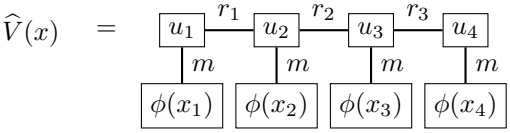


Figure 2. Graphical representation of $\widehat{V} : \mathbb{R}^4 \to \mathbb{R}$.

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

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

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. $b^{\mathrm{loc},i}(x) = [b_1^{\mathrm{loc},i}(x), b_2^{\mathrm{loc},i}(x), \dots, b_{M_i}^{\mathrm{loc},i}(x)]$.

More precisely, the local basis functions can be identified using the open edges in Figure 3 as follows. Assuming u_2 is being optimized, we notice that the tensor $\phi(x_1) \circ u_1$ is a 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 3. 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.

Further details as well as explicit formulas are given in Appendix A.1.

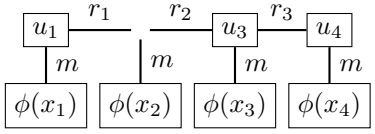
$b^{\mathrm{loc},i}(x) \;=$ 

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 $c_g \in \mathbb{R}$.

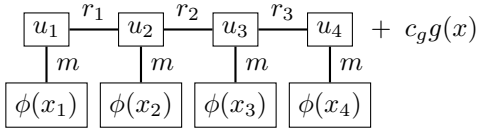
$\widehat{V}(x) \;=$ 

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.

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

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

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

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 $H^2(a,b)$-orthonormal [5], where a and b are set to be approximately equal to the minimum and maximum of the samples $\widehat{X}_n$, respectively. In Appendix D.1 we state the exact choices of a and b for the individual numerical tests. 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).

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.

Clearly, the iterative optimization of (20) is computationally more costly than the explicit scheme described in Section 3.1 that relies on a single optimization of the type (17) per time step. However, implicit schemes typically ensure improved convergence orders as well as robustness (Kloeden & Platen, 1992) and therefore hold the promise of more accurate approximations (see Section 4 for experimental confirmation). We note that the NN based approaches considered as baselines in Section 4 perform gradient descent for both the explicit and implicit schemes and therefore no significant differences in the corresponding runtimes are expected. For convenience, we summarize the developed methods in Algorithm 3.2.

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

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 (explicitly or by iterative schemes)
  set $\widehat{V}_n$ to be the minimizer
end for

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.

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

$$(\partial_t + \Delta) V(x,t) - |\nabla V(x,t)|^2 = 0, \quad (21a)$$
$$V(x,T) = g(x), \quad (21b)$$

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

$$b = \mathbf{0}, \quad \sigma = \sqrt{2}\,\mathrm{Id}_{d\times d}, \quad h(x,s,y,z) = -\frac{1}{2}|z|^2 \quad (22)$$

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:

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(x+\sqrt{T-t}\sigma\xi)}\right], \quad (23)$$

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 $\left|\frac{\widehat{V}_n(x_0) - V_{\mathrm{ref}}(x_0,0)}{V_{\mathrm{ref}}(x_0,0)}\right|$ with $V_{\mathrm{ref}}(x_0, 0) = 4.589992$ 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 worse in terms of accuracy than its implicit counterpart in all our experiments, but takes a very similar amount of computation time we will omit reporting it for the remaining experiments. In Figures 5 and 6 we plot the reference solutions computed by (23) 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. We can see that the TT approximations perform particularly well in the higher dimensional case $d = 100$.

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

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

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.

The accuracy of the TT approximations is surprising given that the ansatz functions are constant in space. We further investigate this behavior in Table 2 and observe that the required polynomial degree decreases with increasing dimension. While similar "blessings of dimensionality" have been reported and discussed (see, for instance, Figure 3 in (Bayer et al., 2021) and Section 1.3 in (Khoromskij, 2012)), a thorough theoretical understanding is still lacking. To

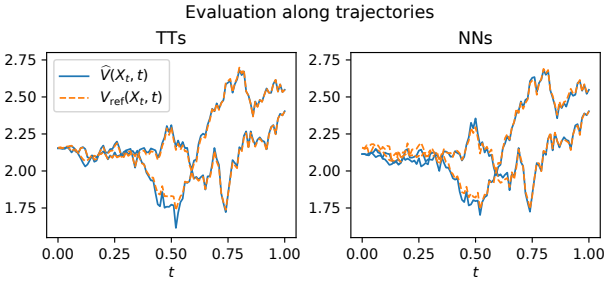


Figure 5. Reference solutions compared with implicit TT and NN approximations along two trajectories in $d = 10$.

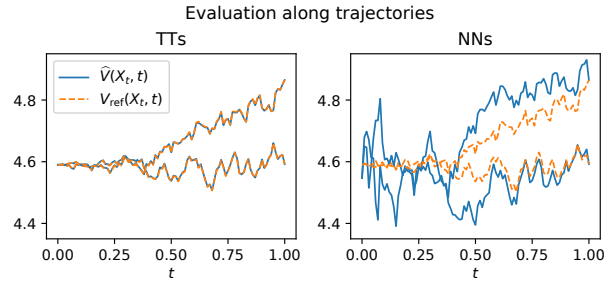


Figure 6. Reference solutions compared with implicit TT and NN approximations along two trajectories in $d = 100$.

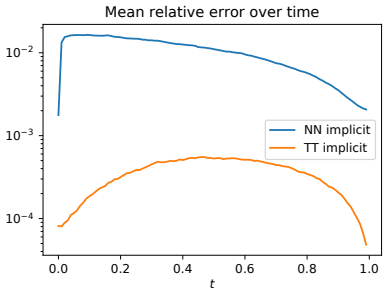


Figure 7. Mean relative error for TT and NN attempts.

guide intuition, we would like to point out that the phenomenon that high-dimensional systems become in some sense simpler is well known from the theory of interacting particle systems ("propagation of chaos", see (Sznitman, 1991)): In various scenarios, the joint distribution of a large number of particles tends to approximately factorize as the number of particles increases (that is, as the dimensionality of the joint state space grows large). It is plausible that similar approximate factorizations are relevant for high-dimensional PDEs and that tensor methods are useful (i) to

detect this effect and (ii) to exploit it. In this experiment, the black-box nature of neural networks does not appear to reveal such properties.

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

$$(\partial_t + L)\, V(x,t) - \frac{1}{2}|(\sigma^\top \nabla V)(x,t)|^2 = 0, \tag{24a}$$

$$V(x,T) = g(x), \tag{24b}$$

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

$$b = -\nabla \Psi, \qquad \Psi(x) = \sum_{i,j=1}^{d} C_{ij}(x_i^2 - 1)(x_j^2 - 1) \tag{25}$$

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,

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x \right], \tag{26}$$

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 $T = 0.5, d = 50, \Delta t = 0.01, K = 2000, \nu_i = 0.05$. We set the TT-rank to 2, use polynomial degree 3 and refer to Appendix D for further details on the TT and NN configurations. 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.

Let us now consider a non-diagonal matrix $C = \mathrm{Id}_{d\times d} + (\xi_{ij})$, where $\xi_{ij} \sim \mathcal{N}(0, 0.1)$ 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 (26) using 10^7 samples. We see in Table 4 that TTs are much faster than NNs, while yielding a similar performance. Note that due to the non-diagonality of C it is expected that the TTs are of rank larger than 2. For the explicit case we do not cap the ranks of the TT and the rank-adaptive solver finds ranks of mostly 4 and never larger than 6. Motivated by these results we cap the ranks at $r_i \leq 6$ in the implicit case and indeed they are obtained for nearly every dimension, as seen from the ranks below,

$$[5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5].$$

The results were obtained with polynomial degree 7.

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{TT}_{\mathrm{expl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	9.6876	9.6865	9.6942
relative error	$1.41e^{-3}$	$1.53e^{-3}$	$7.27e^{-4}$
reference loss	$1.36e^{-3}$	$3.25e^{-3}$	$4.25e^{-3}$
PDE loss	$3.62e^{-2}$	11.48	$2.66e^{-1}$
computation time	95	16	1987

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

	$\mathrm{TT}_{\mathrm{impl}}$	$\mathrm{TT}_{\mathrm{expl}}$	$\mathrm{NN}_{\mathrm{impl}}$
$\widehat{V}_0(x_0)$	35.015	34.756	34.917
relative error	$1.52e^{-3}$	$2.82e^{-3}$	$4.24e^{-3}$
reference loss	$1.30e^{-2}$	$1.59e^{-2}$	$6.38e^{-2}$
PDE loss	79.9	341	170.64
computation time	460	15	16991

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

$$\partial_t V(x,t) + \frac{1}{2}\sum_{i,j=1}^{d} \sqrt{x_i x_j}\gamma_i \gamma_j \partial_{x_i}\partial_{x_j} V(x,t)$$
$$+ \sum_{i=1}^{d} a_i(b_i - x_i)\partial_{x_i} V(x,t) - \left(\max_{1\leq i\leq d} x_i\right) V(x,t) = 0. \tag{27}$$

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. For this test it is again sufficient to set the TT-rank to 1 and the polynomial degree to 3. The TT approximations seem to be slightly better and we note that the explicit TT scheme is again much faster.

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

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

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	9.04e^{-2}	7.80e^{-4}	1.05e^{-3}	5.06e^{-4}
comp. 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.

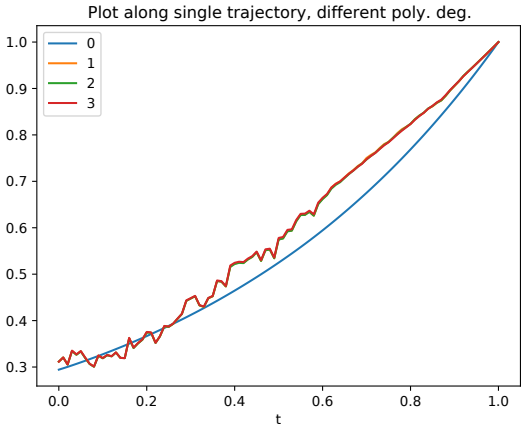


Figure 8. Reference trajectory for different polynomial degrees.

5. Conclusions and outlook

In this paper, we have developed tensor train based approaches towards solving high-dimensional parabolic PDEs, relying on reformulations in terms of BSDEs. For the discretization of the latter, we have considered both explicit and implicit schemes, allowing for a trade-off between approximation accuracy and computational cost. Notably, the tensor train format specifically allows us to take advantage of the additional structure inherent in least-squares based formulations, particularly in the explicit case.

More elaborate numerical treatments for BSDEs (involving, for instance, multi-step and/or higher-order discretizations) have been put forward in the literature (Chassagneux, 2014; Crisan et al., 2014; Macris & Marino, 2020). Combining these with tensor based methods remains a challenging and interesting topic for future research. Finally, we believe that the "blessing of dimensionality" observed in Section 4.1 deserves a mathematically rigorous explanation; progress in this direction may further inform the design of scalable schemes for high-dimensional PDEs.

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). L. S. acknowledges support from the Research Training Group 'Differential Equation- and Data-driven Models in Life Sciences and Fluid Dynamics: An Interdisciplinary Research Training Group (DAEDALUS)'(GRK 2433) funded by Deutsche Forschungsgemeinschaft (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.

Bayer, C., Eigel, M., Sallandt, L., and Trunschke, P. Pric-

ing high-dimensional bermudan options with hierarchical tensor formats. *arXiv preprint arXiv:2103.01934*, 2021.

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.

Chassagneux, J.-F. Linear multistep schemes for bsdes. *SIAM Journal on Numerical Analysis*, 52(6):2815–2836, 2014.

Chen, Y. and Lu, Z. Tensor decomposition and high-performance computing for solving high-dimensional stochastic control system numerically. *Journal of Systems Science and Complexity*, pp. 1–14, 2021.

Crisan, D., Manolarakis, K., et al. Second order discretization of backward sdes and simulation with the cubature method. *Annals of Applied Probability*, 24(2):652–678, 2014.

Dektor, A., Rodgers, A., and Venturi, D. Rank-adaptive tensor methods for high-dimensional nonlinear pdes. *arXiv preprint arXiv:2012.05962*, 2020.

Dolgov, S., Khoromskij, B. N., Litvinenko, A., and Matthies, H. G. Polynomial chaos expansion of random coefficients and the solution of stochastic partial differential equations in the tensor train format. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1109–1135, 2015.

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.

Eigel, M., Pfeffer, M., and Schneider, R. Adaptive stochastic galerkin fem with hierarchical tensor representations. *Numerische Mathematik*, 136(3):765–803, 2017.

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.

Gorodetsky, A., Karaman, S., and Marzouk, Y. High-dimensional stochastic optimal control using continuous tensor decompositions. *The International Journal of Robotics Research*, 37(2-3):340–377, 2018.

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.

Horowitz, M. B., Damle, A., and Burdick, J. W. Linear hamilton jacobi bellman equations in high dimensions. In *53rd IEEE Conference on Decision and Control*, pp. 5880–5887. IEEE, 2014.

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.

Kormann, K. A semi-Lagrangian Vlasov solver in tensor train format. *SIAM Journal on Scientific Computing*, 37 (4):B613–B632, 2015.

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

Landsberg, J. M. Tensors: geometry and applications. *Representation theory*, 381(402):3, 2012.

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.

Lubasch, M., Moinier, P., and Jaksch, D. Multigrid renormalization. *Journal of Computational Physics*, 372:587–602, 2018.

Macris, N. and Marino, R. Solving non-linear kolmogorov equations in large dimensions by using deep learning: a numerical comparison of discretization schemes. *arXiv preprint arXiv:2012.07747*, 2020.

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.

Stefansson, E. and Leong, Y. P. Sequential alternating least squares for solving high dimensional linear hamilton-jacobi-bellman equation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3757–3764. IEEE, 2016.

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.

Sznitman, A.-S. Topics in propagation of chaos. In *Ecole d'été de probabilités de Saint-Flour XIX—1989*, pp. 165–251. Springer, 1991.

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.