Pricing high-dimensional Bermudan options with hierarchical tensor formats

Christian Bayer, Martin Eigel, Leon Sallandt, Philipp Trunschke

March 3, 2021

Abstract

An efficient compression technique based on hierarchical tensors for popular option pricing methods is presented. It is shown that the "curse of dimensionality" can be alleviated for the computation of Bermudan option prices with the Monte Carlo least-squares approach as well as the dual martingale method, both using high-dimensional tensorized polynomial expansions. This discretization allows for a simple and computationally cheap evaluation of conditional expectations. Complexity estimates are provided as well as a description of the optimization procedures in the tensor train format. Numerical experiments illustrate the favourable accuracy of the proposed methods. The dynamical programming method yields results comparable to recent Neural Network based methods.

1 Introduction

Pricing of American or Bermudan type options, i.e., options with an early exercise feature, is one of the most classical, but also most difficult problems of computational finance, producing a vast amount of literature. Some examples of popular classes of methods include PDE methods (see, for instance, [AP05]), tree and stochastic mesh methods (see, for instance, [Gla13]), and policy iteration (see, e.g., [BS18]). In this paper we consider two other very popular methodologies, namely least squares Monte Carlo methods based on the dynamic programming principles pioneered by [LS01] and dual martingale methods introduced by [Rog02], both of which were, of course, widely adapted and considerably improved since then. We refer to [Lud20] for a recent overview together with an open-source implementation.

Both least squares Monte Carlo methods and duality methods require efficient and accurate approximation of functions from a potentially large class. Indeed, the key step of the least squares Monte Carlo method involves the computation of a *continuation value*, i.e., of the conditional expectation $\mathbb{E}_t[v(t+\Delta t, X_{t+\Delta t})]$ of a future *value function* at time t.[1] (For sake of presentation, let

[1]Actual algorithms may rather involve actual future payoffs such as in [LS01]. Note that we ignore discounting at this time.

us assume that we are using an asset price model based on a Markov process X, which contains the asset prices S, but possibly also further components, such as stochastic volatilities or interest rates.) This conditional expectation is then approximated within a finite dimensional space spanned by *basis functions* – often chosen to be polynomials. When the dimension d of the underlying process X is high, we encounter a curse of dimensionality, i.e., we expect that the number of basis functions needed to achieve a certain accuracy increases exponentially in the dimension d. This is especially true when the basis functions are chosen by "tensorization" of one-dimensional basis functions. E.g., the dimension of the space of polynomials of (total) degree p in d variables is $\binom{d+p}{d}$. Such a polynomial basis become inefficient when $d \gg 1$, a realistic scenario for options on baskets or indices. For instance, options on SPY (with 100 assets) are American, implying that $d \geqslant 100$, depending on the choice of the model – in the sense that continuation values also depend on volatilities not just the asset prices in stochastic volatility models, for example. Hence, other classes of basis functions are needed.

Duality methods are typically based on parameterizations of families of candidate martingales. In the Markovian case, we may restrict ourselves to martingales representable as stochastic integrals of functions $\phi(t, X_t)$ against the driving Brownian motion, and we again see a potential curse of dimensionality in terms of the dimension of X.

When the underlying model is *not Markovian* – as, e.g., common for *rough volatility models*, see, e.g., [BFG16] – the involved dimensions can increase drastically, as then both continuation values and candidate martingales theoretically depend on the entire trajectory of the process X until time t. There are only very few rigorously analyzed methods for such non-Markovian problems. We specifically refer to [Lel18; Lel19], both of which are based on Wiener chaos expansions of the value process and the candidate martingale, respectively. In this framework, conditional expectations can be computed explicitly, but the curse of dimension enters via the chaos decomposition itself, see Section 2.2 for details.

In either case, we are faced with "natural" d-dimensional bases which quickly increase in size as d increases. While the curse of dimension is often a real, inescapable fact of complexity theory (in the sense of a worst case dependence over sufficiently general classes of approximation problems), real life problems often exhibit structural properties which lead to a notion of "effective dimension" of a problem which may increase much slower than the actual dimension d – see, for instance, [WS05] for a similar phenomenon in finance. This insight has lead to efficient approximation strategies for high-dimensional functions of low effective dimension of some sort in numerical analysis. In this paper, we propose to use hierarchical tensor formats, more precisely *tensor trains*, to provide efficient approximations of nominally high-dimensional functions, provided that they allow for accurate *low-rank approximations*.

Hierarchical tensors (HT) [BSU16; HS14] rely on the classical concept of *separation of variables* by means of a generalization of the singular value decomposition (SVD) to higher-order tensors, preserving many of its well-known properties. The hierarchical SVD (HSVD) yields a notion of multilinear rank and provides an approach to obtain a quasi-optimal low-rank approximation by rank truncation. For fixed multilinear ranks, the representation and operation complexities of these formats scale only linearly in the order of the tensor. Central to the HSVD is a tree-based representation of a recursive decomposition of the tensor space into nested subspaces. For the described algorithms, we use the common tensor train (TT) format [OT09; Ose11; Ose13], which is a "linearization" of the HT representation with general binary trees. Similar to matrices, the set of hierarchical tensors of fixed multilinear rank is not convex but forms a smooth manifold. Hence, appropriate optimization techniques such as alternating and Riemannian schemes are available.

Tensor trains are a new technique in computational finance. In fact, we are only aware of one other paper in the field using these tensor representations, namely [GKS20]. In that paper, the authors consider parametric option pricing problems. That is, they are given a model with parameters ζ and options with parameters η. The price of these options in the model is then a function $P(\theta)$, $\theta := (\zeta, \eta)$, of the model and option parameters, and we can expect P to be regular. Some tasks in financial engineering require rapid option pricing, e.g., for calibrating model parameters to market prices. Following [Gaß+18], [GKS20] propose to approximate $\theta \mapsto P(\theta)$ by Chebyshev interpolation. If θ is high-dimensional, such a interpolation may already involve a very large number of Chebyshev polynomials, and they then proceed to "compress" the representation using tensor trains.

No discussion of computational methods for high-dimensional problems can today ignore the trend of using machine learning techniques, in particular deep neural networks, to often great success. In the context of American or Bermudan options, we mention the recent paper by [BCJ19], who are able to accurately price high-dimensional Bermudan options in dimensions up to 500 using deep learning techniques based on parameterization of *randomized* stopping times, see also [BTW20]. A natural question then is if the successes of deep learning for solving high dimensional problems (*"overcoming" the curse of dimension*) can also be achieved by other, more traditional methods of numerical analysis.

Main contributions

Our intention is to advocate the use of hierarchical tensor formats for high-dimensional problems in computational finance. For this, we provide an overview of the main ideas of these formats and illustrate the application of tensor trains with two popular methods using tensorized polynomial spaces for the discretization. The considered problem sizes would be infeasible without some efficient model order reduction technique. We demonstrate in particular that the achieved accuracy is comparable to recent Neural Network approaches.

Tensor networks have already been used to alleviate the curse of dimensionality in physics [Vid03], parametric PDEs [BSU16; EPS17; Eig+19; Eig+20] as well as other control problems [DKK19; OSS19; Fac+20]. They may significantly reduce the computational complexity [Hac12] and are able to represent sparse functions with a constant overhead [BCD17]. In this paper we demonstrate the usefulness of tensor networks in computational finance on two examples with discretizations in polynomial tensor product spaces in d dimensions with degree p of the form

$$X = \sum_{\alpha \in [p]^d} X_\alpha P_\alpha \tag{1}$$

with coefficient tensor $X \in \mathbb{R}^{p^d}$. The first example showcases the application of the alternating least squares algorithm [HRS12c] for the best approximation problem in the primal method of Longstaff and Schwartz [LS01] where the discounted value is given by

$$v(x) = \sum_{\alpha \in \Lambda} V_\alpha \prod_{k=1}^{d'} B_{\alpha_k}(x_k). \tag{2}$$

In the second example we present the application of a Riemannian optimization algorithm [KSV14] to solve the convex minimization problem in the dual method of Lelong [Lel18]. For both examples we examine the reduction of the space and time complexity. In the numerical experiments we compare the originally published and the new methods on standard problems. The reduced complexity allows to apply the Longstaff-Schwartz algorithm to problems with up to 1000 assets. Problems of this size have only been reported recently with state-of-the-art machine learning methods [BCJ19]. Moreover, in comparison to the Neural Network approach, our method requires significantly fewer samples. Even though the application of the tensor compression to the dual method turned out to be quite involved (in terms of the tensor optimization), the resulting algorithm produces comparable or better results while considerably reducing the dimensionality of the underlying equation. This renders this approach tractable for more assets and higher accuracy computations.

We conclude that tensor networks can be very beneficial technique for high-dimensional problems in financial mathematics. They rival the performance of Neural Networks, show similar approximation and complexity properties, and exhibit richer mathematical structures that can be exploited (such as in the Riemannian optimization described in Section 3.3).

2 Bermudan option pricing

In what follows we introduce our frameworks and notations for the Bermudan option pricing problem. Furthermore, we recall the celebrated Longstaff-Schwartz algorithm as well as Lelong's version of Rogers' duality approach based on a Wiener chaos expansion.

We fix some finite time horizon $T > 0$ and a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0 \leqslant t \leqslant T}, \mathbb{P})$, where $(\mathcal{F}_t)_{0 \leqslant t \leqslant T}$ is supposed to be the natural augmented filtration of a d-dimensional Brownian motion B – the natural setting for the Wiener chaos expansion lying at the core of our duality algorithm. On this space, we consider an adapted Markov process $(S_t)_{0\leqslant t\leqslant T}$ with values in $\mathbb{R}^{d'}$ modeling a d'-dimensional underlying asset. The number of assets d' can be smaller than the dimension d of the Brownian motion to encompass the case of stochastic volatility models or stochastic interest rate. To simplify notation, we consider the case that S generates the filtration and $d' = d$.

We assume that $\mathbb{P}$ is an associated risk neutral measure. We consider an adapted payoff process $\widetilde{Z}$ and introduce its discounted value process

$$\left(Z_t = \exp(-\int_0^t r(s)\,\mathrm{d}s)\widetilde{Z}_t\right)_{0\leqslant t\leqslant T}.$$

We assume that the paths of Z are right continuous and that $\sup_{t\in[0,T]}|Z_t| \in L^2(\Omega, \mathcal{F}_T, \mathbb{P})$. The process $\widetilde{Z}$ can obviously take the simple form $(\varphi(S_t))_{t\leqslant T}$ for some function φ, but it can also depend on the whole path of the underlying asset S up to the current time. We consider the Bermudan option paying $\widetilde{Z}_{t_k}$ to its holder if exercised at time $0 = t_1 < \cdots < t_N = T$. Standard arbitrage pricing theory defines the discounted time-t value of the Bermudan option to be

$$U_{t_n} = \operatorname*{ess\,sup}_{\tau\in\mathcal{T}_{t_n}} \mathbb{E}[Z_\tau|\mathcal{F}_\tau] \tag{3}$$

where $\mathcal{T}_t$ denotes the discrete set of $\mathcal{F}$-stopping times with values in $[t, T]$.

We now recall two of the many algorithms for pricing Bermudan options available in the literature, beginning with the classical Longstaff-Schwartz algorithm. These algorithms will be used to test the efficiency gains achievable by hierarchical tensor formats in the context of option pricing.

2.1 Primal (Longstaff-Schwartz)

In the Longstaff-Schwartz algorithm [LS01], the dynamic programming principle corresponding to the discounted time-t value of the Bermudan option (3), is used. It reads

$$U_{t_n} = \max\{Z_{t_n}, \mathbb{E}[U_{t_{n+1}}|\mathcal{F}_{t_n}]\} \tag{4}$$

with final condition $U_{t_N} = Z_{t_N}$. If $\mathbb{E}[U_{t_n+1}|\mathcal{F}_{t_n}]$ is known, an optimal stopping-time policy can be synthesized explicitly by stopping if and only if $Z_{t_n} \geqslant \mathbb{E}[U_{t_{n+1}}|\mathcal{F}_{t_n}]$. Thus, the problem of finding the optimal stopping time and also the valuation of the option can be reduced to finding $\mathbb{E}[U_{t_{n+1}}|\mathcal{F}_{t_n}]$, which is exactly what the Longstaff-Schwartz algorithm approximates. As this algorithm is pretty standard, we do not give a detailed explanation and instead simply state the algorithm. Note that we abbreviate the notation by dropping the t in the discretization, i.e. $S_{t_n} = S_n$. We define the ITM ("in the money") operator which is mapping a set of assets to the subset where the current payoff is positive.

Algorithm 1: Longstaff-Schwartz

```
input  : Number of samples $M$, exercise dates $0 = t_1 < \cdots < t_N = T$,
         initial value $s_0$.
output: Conditional expectations $v_n(x) = \mathbb{E}[U_{n+1}|S_n = x]$, $n \leqslant N$.
Set $S_0^m = s_0$ and compute trajectories: $S_n^m$ for $m = 1, \dots M$,
 $n = 1, \dots N$.
Set
        $$Y^m = Z_n^m \qquad (5)$$
for $k = n - 1$ to $1$ do
    Find ITM paths $S_{\tilde{m}}$ for $m \in \mathrm{ITM} \subset \{1, \dots, M\}$. Set
        $$v_n(\cdot) \approx \operatorname*{arg\,min}_{v \in \mathcal{M}} \frac{1}{|\mathrm{ITM}|} \sum_{\tilde{m} \in \mathrm{ITM}} |v(S_n^{\tilde{m}}) - Y^{\tilde{m}}|^2. \qquad (6)$$
    for $m = 1$ to $M$ do
        if $m \in \mathrm{ITM}$ and $Z_n^m > v_k(S_n^m)$ then
            $Y^n = Z_n^m$.
    end
end
Set $v_0(s_0) = \sum_{m=1}^M Y^m$.
```

Note that in this formulation of the algorithm, the set $\mathcal{M}$ in (6) is traditionally a linear space of polynomials. Adding the payoff function to the ansatz space is a common trick to improve the result, see e.g. [Gla13]. In this work we use the set of tensor trains, which we explain in Section 5.

The key computational challenge is the approximation of the conditional expectation

$$v(S_n) = \mathbb{E}[U_{n+1}|S_n] = \sum_{\alpha \in \mathbb{N}^{d'}} v_\alpha B_\alpha(S_n) \tag{7}$$

for some $L^2(\mathbb{R}^{d'}, \mathcal{B}(\mathbb{R}^{d'}), S_*\mathbb{P})$-orthogonal basis $\{B_k\}_{k \in \mathbb{N}}$, where we tacitly assume the payoff having finite second moments. Since this is an L^2-orthogonal projection we can choose a finite set of multi-indices $\Lambda \subset \mathbb{N}^{d'}$ and approximate $\mathbb{E}[Y|S_n]$ by minimizing

$$\left\| Y - \sum_{\alpha \in \Lambda} v_\alpha B_\alpha(S_n) \right\|^2 \approx \frac{1}{m} \sum_{i=1}^{m} \left(Y^m - \sum_{\alpha \in \Lambda} v_\alpha B_\alpha(S_n^m) \right)^2. \tag{8}$$

We use the index set $\Lambda = [p]^{d'}$ and mitigate the "curse of dimensionality" by representing v in the tensor train format as defined in Section 3.

2.2 Chaos-martingale minimization

Rogers [Rog02] reformulates the problem of computing U_0 as the following dual optimization problem

$$U_0 = \inf_{M \in H_0^2} \mathbb{E}\left[\max_{n=1,\ldots,N}(Z_{t_n} - M_{t_n})\right]$$

where H_0^2 denotes the set of square integrable martingales vanishing at zero. This approach requires us to optimize over the space of all (square integrable) martingales. As any martingale M can be expressed as conditional expectations $t \mapsto \mathbb{E}[X|\mathcal{F}_t]$ for some square integrable random variable X, we may equivalently solve

$$U_0 = \inf_{X \in L_0^2(\Omega,\mathcal{F}_T,\mathbb{P})} \mathbb{E}\left[\max_{n=1,\ldots,N}(Z_{t_n} - \mathbb{E}[X|\mathcal{F}_{t_n}])\right], \tag{9}$$

where $L_0^2(\Omega, \mathcal{F}_T, \mathbb{P})$ is the set of square integrable $\mathcal{F}_T$-random variables with zero mean. This allows us to minimize over a (seemingly) simpler space – namely the space of square integrable random variables rather than the space of martingales – at the cost of expensive calculations of conditional expectations.

The ingenious idea of Lelong [Lel18] was to use a specific parameterization of the space of square integrable random variables in which conditional expectations w.r.t. the filtration $(\mathcal{F}_t)$ can be computed explicitly at virtually no cost. Indeed, a finite-dimensional approximation of $X \in L_0^2(\Omega, \mathcal{F}_T, \mathbb{P})$ with the above property is given by the truncated Wiener chaos expansion

$$\widetilde{X} = \sum_{\alpha\in\Lambda} \widetilde{X}_\alpha H_\alpha(G_1, \ldots, G_N), \tag{10}$$

where $\Lambda \subseteq \mathbb{N}^{N\times d'}$ is a predefined set of multi-indices, H_α is the tensorized Hermite polynomial with multi-index α and $G_1, \ldots, G_N$ are d'-dimensional Gaussian increments. The tensorized Hermite polynomials are defined by

$$H_\alpha(G_1, \ldots, G_N) := \prod_{n=1}^{N}\prod_{k=1}^{d'} h_{\alpha_{nk}}(G_{n,k}) \tag{11}$$

where $h_{\alpha_{nk}}$ are the univariate Hermite polynomials with index α_{nk}. Defining the subset $\Lambda^n := \{\alpha \in \Lambda : \forall k > n, \alpha_k = 0\}$ it is easy to see that

$$\mathbb{E}[\widetilde{X}|\mathcal{F}_{t_n}] = \sum_{\alpha\in\Lambda^n} \widetilde{X}_\alpha H_\alpha(G_1, \ldots, G_N). \tag{12}$$

This means that the linear expectation operator $\mathbb{E}[\,\bullet\,|\mathcal{F}_{t_n}]$ can be represented with the coefficient tensor simply by dropping trailing terms of the chaos expansion. The expectation in (9) can thus be estimated by the sample average

$$U_0 = \inf_{\substack{\widetilde{X}_0=0\\ \widetilde{X}_\alpha\in\mathbb{R}}} \frac{1}{m}\sum_{i=1}^{m}\left[\max_{n=1,\ldots,N}\left(Z_{t_n}^{(i)} - \sum_{\alpha\in\Lambda^n}\widetilde{X}_\alpha H_\alpha(G_1^{(i)}, \ldots, G_N^{(i)})\right)\right], \tag{13}$$

where $(Z^{(i)}, G^{(i)})_{1 \leqslant i \leqslant m}$ are i.i.d. samples from the distribution of (Z, G). It is shown in [Lel18] that this is an infimum of a convex, continuous and piece-wise linear cost function over a convex domain and can be calculated easily by a gradient descent descent method with an Armijo line search.

The choice of the multi-index set Λ plays an important role in the preformance and applicability of this algorithm. In [Lel18] Λ is chosen such that the polynomial degree $\sum_{n=1}^{N} \sum_{k=1}^{d'} \alpha_{nk}$ is bounded by p. This bounds the number of entries of $\widetilde{X}$ that have to be stored by $\binom{Nd'+p}{Nd'} \in \mathcal{O}\left(\frac{(Nd'+p)^p}{p!}\right)$. For fixed p this can scale unfavourably when the number of exercise dates N or the dimension of the Brownian motion (i.e. the number of assets) d' increases. We propose to choose $\Lambda = \Lambda_p^N$ such that $\sum_{k=1}^{d'} \alpha_{nk} \leqslant p$ for $\alpha_n \in \Lambda_p$. and to use the tensor train format to alleviate the ensuing "curse of dimensionality". We introduce the relevant notions and central concepts in the following section.

3 Low-rank tensor representations

We are concerned with an efficient representation of expansions of the form $\sum_{\alpha \in \Lambda} U_\alpha \prod_{j=1}^{d} P_{\alpha_j}$ in tensorized polynomials P_α determined by some finite set $\Lambda \subset \mathcal{F} := \{\alpha \in \mathbb{R}^{\mathbb{N}} \ : \ |\operatorname{supp} \alpha| < \infty\}$ of finitely supported multi-indices. This representation is used for the considered algorithms with tensorized expansions given by (10) and (33). The set Λ typically is given as a tensor set $\Lambda = \times_{j=1}^{d} \mathcal{I}_n := [n]^d$ or as anisotropical set $\Lambda = \times_{j=1}^{d} \mathcal{I}_{p_j}$, where in our setting p_j denotes the maximal polynomial degree in dimension $j = 1, \dots, d$. Apparently, $\#\Lambda$ is in $\mathcal{O}(p^d)$ with $p := \max\{p_j \ : \ j = 1, \dots, d\}$. To cope with this exponential complexity, a potentially very efficient approach is the use of low-rank tensor representations as e.g. presented in [HS14; Nou17]. Since these modern model reduction techniques are not widely known in the finance community yet, we provide a brief review in order to elucidates some of the central principles. In the presentation, we follow [RSS17; BSU16].

3.1 Tensor product spaces and subspace approximation

We consider finite dimensional linear spaces $U_i = \mathbb{R}^{p_i}$ and define the tensor product space

$$\mathcal{H}_d := \bigotimes_{j=1}^{d} U_j. \tag{14}$$

Fixing the canonical basis for all U_j, any tensor $\mathbf{u} \in \mathcal{H}_n$ can be represented by

$$\mathbf{u} = \sum_{\nu_0=1}^{p_1} \cdots \sum_{\nu_n=0}^{p_n} \mathbf{U}(\nu_1, \dots, \nu_n) \mathbf{e}_{\nu_1}^1 \otimes \cdots \otimes \mathbf{e}_{\nu_n}^n, \quad \mathbf{U} \in \mathbb{R}^{p_1} \otimes \cdots \otimes \mathbb{R}^{p_n}. \tag{15}$$

Hence, given this basis, any multi-index $\nu \in \mathcal{F}$ can be identified with a component in the (coefficient) tensor $\mathbf{U}$, i.e.

$$\nu = (\nu_1, \dots, \nu_n) \mapsto \mathbf{U}(\nu_1, \dots, \nu_n) \in \mathbb{R}. \tag{16}$$

The goal is to obtain a compressed representation of (15) in an analytically and numerically more favourable format by exploiting an assumed low-rank structure. Hierarchical representations have appealing properties making them attractive for the treatment of the problems at hand. For example, they contain sparse polynomials, but are much more flexible at a price of a slightly larger overhead, see e.g. [BCD18; BD16] for a comparison concerning parametric PDEs.

To introduce the concept of *subspace approximations*, which is central to the complexity properties of tensor formats, we start with the classical *Tucker format*. Given a tensor $\mathbf{U}$ and a *rank tuple* $\mathbf{r} := (r_j)_{j=1}^d$, the approximation problem reads: find optimal subspaces $V_j \subset U_j$ such that

$$\min_{\mathbf{V}\in\mathcal{V}_d} \|\mathbf{U}-\mathbf{V}\| \qquad \text{with} \quad \mathcal{V}_d := \bigotimes_{j=1}^{d} V_j \tag{17}$$

is minimized over $V_1, \dots, V_d$, with $\dim V_j = r_j$. An equivalent problem is to find the corresponding basis vectors $\{b^j_{k_j}\}_{k_j=1,\dots,r}$ of V_j which can be written in the form

$$b^j_{k_j} := \sum_{\nu_j=1}^{p_j} b^j(\nu_j, k_j)\mathbf{e}^j_{\nu_j}, \qquad k_j = 1, \dots, r_j < p_j. \tag{18}$$

Note that this can be understood as the construction of a reduced basis. The optimal tensor $\mathbf{V}$ can thus be represented by

$$\mathbf{V} = \sum_{k_1}^{r_1} \cdots \sum_{k_d=1}^{r_d} \mathbf{c}(k_1, \dots, k_d) b^1_{k_1} \otimes \cdots \otimes b^d_{k_d} \in \mathcal{V}_d. \tag{19}$$

In case of orthonormal bases $\{b^j_{k_j}\}_{k_j=1,\dots,r_j}$, the *core tensor* $\mathbf{c} \in \bigotimes_{j=1}^d \mathbb{R}^{p_j}$ is given entry-wise by projection,

$$\mathbf{c}(k_1, \dots, k_d) = (\mathbf{v}, b^1_{k_1} \otimes \cdots \otimes b^d_{k_d}). \tag{20}$$

With a complexity of $\mathcal{O}(p_j r_j)$ for each basis $\{b^j_{k_j}\}_{k_j=1,\dots,r_j}$ and a complexity of $\mathcal{O}(r^d)$ for the core tensor $\mathbf{c}$, the complexity of the Tucker representation (19) is $\mathcal{O}(pdr + r^d)$ with $r := \max\{r_j : j = 1, \dots, d\}$ and $p := \max\{p_j : j = 1, \dots, d\}$. As such, the Tucker representation is not sufficient to cope with exponential representation complexity and the format exhibits other problems such as non-closedness. Nevertheless, the ideas described above eventually lead to a very efficient format by hierarchization of the bases as described in what follows.

3.2 Hierarchical tensor representations

The *hierarchical Tucker* (HT) format introduced in [HK09] is an extension of the notion of subspace approximation to a hierarchical setting determined by a dimension tree as shown in Figure 3.2 where the indices $j = 1, \dots, d$ correspond to the spaces U_j of the tensor space $\mathcal{H}_d$. Note that by cutting any edge in the tree, two subtrees are generated. Collecting the indices for each subtree, a tensor of order two (a matrix) arises. By this, fundamental principles from matrix analysis, in particular the singular value decomposition (SVD), can be transferred to the higher-order tensor setting.

To illustrate the central idea, consider the optimal Tucker-subspaces $V_1 \otimes V_2 \subseteq U_1 \otimes U_2 = \mathbb{R}^{p_1} \otimes \mathbb{R}^{p_2}$. For the approximation of $\mathbf{u} \in \mathcal{H}_d$, often only a subspace $V_{\{1,2\}} \subset V_1 \otimes V_2$ with dimension $\dim(V_{\{1,2\}}) = r_{\{1,2\}} < r_1 r_2 = \dim(V_1 \otimes V_2)$ is required. In fact, $V_{\{1,2\}}$ is defined by a basis

$$V_{\{1,2\}} = \operatorname{span}\left\{ b^{\{1,2\}}_{k_{\{1,2\}}} : \ k_{\{1,2\}} = 1, \dots, r_{\{1,2\}} \right\} \tag{21}$$

with basis vectors

$$b^{\{1,2\}}_{k_{\{1,2\}}} = \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} \mathbf{b}^{\{1,2\}}(k_1, k_2, k_{\{1,2\}}) b^1_{k_1} \otimes b^2_{k_2}, \quad k_{\{1,2\}} = 1, \dots, r_{\{1,2\}} \tag{22}$$

and coefficient tensors $\mathbf{b}^{\{1,2\}} \in \mathbb{R}^{r_1 \times r_2 \times r_{\{1,2\}}}$ where

$$b^{\{j\}}_{k_{\{j\}}} := \sum_{\nu_j=1}^{p_j} \mathbf{b}^{\{j\}}(\nu_j, k_{\{j\}}) \mathbf{e}^j_{\nu_j}, \qquad j = 1,2 \text{ and } k_{\{j\}} = 1, \dots, r_j < p_j. \tag{23}$$

are the basis vectors of the Tucker representation (18). This can be generalized to the tensor product space $\mathcal{H}_d$ by the introduction of a *partition tree* (or *dimension tree*) $\mathbb{D}$ with vertices $\alpha \subset D := \{1, \dots, d\}$ and leaves $\{1\}, \dots, \{d\}$ where D is called the root of the tree. Each vertex α that is not a leaf can be partitioned as $\alpha = \alpha_1 \cup \alpha_2$ with $\alpha_1 \cap \alpha_2 = \varnothing$ and $\alpha_1, \alpha_2 \neq \varnothing$. Although not required, one we restrict the topology to a binary tree and denote by α_1, α_2 the children of α. Figure 3.2 is an illustration of the unbalanced tree $\mathbb{D} = \{\{1\}, \{2\}, \{1,2\}, \{3\}, \{1,2,3\}, \dots, \{d\}, \{1, \dots, d\}\}$ where e.g. $\alpha = \{1,2,3\} = \alpha_1 \cup \alpha_2 = \{1,2\} \cup \{3\}$.

Let $\alpha_1, \alpha_2 \subset D$ be the two children of $\alpha \in D$. Then $V_\alpha \subset V_{\alpha_1} \otimes V_{\alpha_2}$ is defined by a basis

$$b^\alpha_\ell = \sum_{i=1}^{r_{\alpha_1}} \sum_{j=1}^{r_{\alpha_2}} \mathbf{b}^\alpha(i,j,\ell) b^{\alpha_1}_i \otimes b^{\alpha_2}_j, \tag{24}$$

where the tensors $(i,j,\ell) \mapsto \mathbf{b}^\alpha(i,j,\ell)$ are called *transfer* or *components tensors* and $\mathbf{b}^D = \mathbf{b}^{\{1,\dots,d\}}$ is called the *root tensor*. To represent a tensor in this hierarchical format it suffices to store the transfer tensors $\mathbf{b}^\alpha$ along with the root tensor $\mathbf{b}^D$. More specifically, $\mathbf{u} \in \mathcal{H}_d$ is obtained from $(\mathbf{b}^\alpha)_{\alpha \in \mathbb{D}}$, via the multilinear function τ

$$(\mathbf{b}^\alpha)_{\alpha \in \mathbb{D}} \mapsto \mathbf{u} = \tau(\{\mathbf{b}^\alpha : \ \alpha \in \mathbb{D}\}), \tag{25}$$

which is defined by the recursive application of the basis representation (24). The mapping τ is a multilinear function in its arguments $\mathbf{b}^\alpha$. A graphical representation of this mapping is depicted in Figure 3.2. In this pictorial description, the contractions of component tensors (24) are indicated as edges between vertices of a graph and the indices of the tensor are represented by open edges. This hierarchical representation has complexity $\mathcal{O}(pdr + dr^3)$ with $p = \max\{p_1, \dots, p_d\}$ and $r = \max\{r_\alpha : \alpha \in \mathbb{D}\}$.

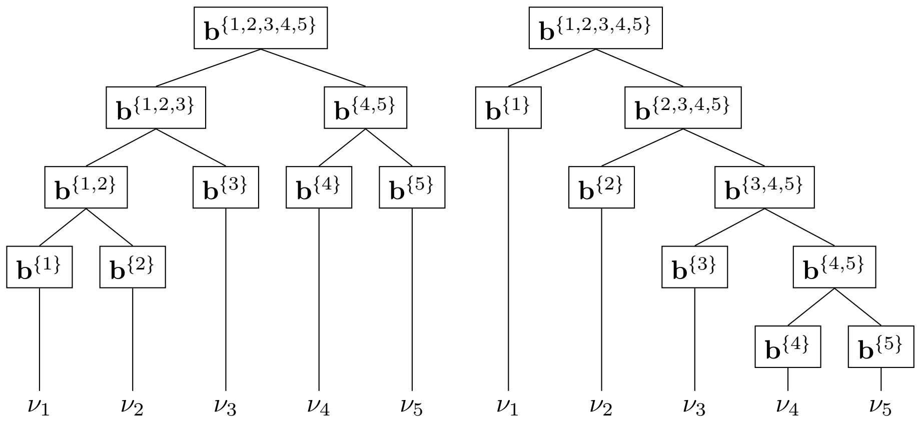


Figure 1: Dimension trees $\mathbb{D}$ for $d = 5$. Balanced HT tree (left) and linearized TT tree (right).

Tensor trains Tensor trains are a subset of the general hierarchical tensors described above. They were introduced to the numerical mathematics community in [OT09; OT10] but have been known to physicists for a long time as matrix product states (MPS). The linear structure is depicted in Figure 3.2 (right), which corresponds to taking $V_{1,\dots,j+1} \subset V_{\{1,\dots,j\}} \otimes V_{\{j+1\}}$. In the example, we consider the unbalanced tree $\mathbb{D} = \{\{1\}, \{2\}, \{1,2\}, \{3\}, \{1,2,3\}, \dots, \{d\}, \{1,\dots,d\}\}$. Applying the recursive construction, any tensor $\mathbf{u} \in \mathcal{H}_d$ can be written as

$$(\nu_1, \dots, \nu_d) \mapsto \mathbf{U}(\nu_1, \dots, \nu_d) = \sum_{k_0}^{r_0} \cdots \sum_{k_d}^{r_d} \mathbf{U}^1(k_0, \nu_1, k_1)\mathbf{U}^2(k_1, \nu_2, k_2) \cdots \mathbf{U}^d(k_{d-1}, \nu_d, k_d), \quad (26)$$

where

$$\begin{aligned} \mathbf{U}^1(\nu_1, k_1) &:= \sum_{\ell=1}^{r_1} \mathbf{b}^{\{1\}}(\nu_1, \ell)\mathbf{b}^{D}(k_1, \ell), \\ \mathbf{U}^j(k_{j-1}, \nu_j, k_j) &:= \sum_{\ell=1}^{r_j} \mathbf{b}^{\{j\}}(\nu_j, \ell)\mathbf{b}^{\{j,\dots,d\}}(k_{j-1}, k_j, \ell), \qquad j = 2, \dots, d-1 \\ \mathbf{U}^d(k_{d-1}, \nu_d) &:= \mathbf{b}^{\{d\}}(\nu_d, k_{d-1}). \end{aligned}$$

This can be reformulated as matrix products

$$\mathbf{U}(\nu_1, \dots, \nu_d) = \prod_{j=1}^{d} \mathbf{b}_j(\nu_j) = \tau(\mathbf{b}^1, \dots, \mathbf{b}^d)(\nu), \quad (27)$$

with component matrices $b_j(\nu_j) \in \mathbb{R}^{r_{j-1}\times r_j}$ given by

$$(b_j(\nu_j))_{k_{j-1},k_j} = \mathbf{b}^j(k_{i-j}, \nu_j, k_j), \quad 1 < j < d, \tag{28}$$

and

$$(b_1(\nu_1))^\intercal_{k_1} = \mathbf{b}^1(\nu_1, k_1), \quad (b_d(\nu_d))_{k_d} = \mathbf{b}^d(k_d, \nu_d). \tag{29}$$

It has to be pointed out that the representation (27) is not unique since in general there exist $\mathbf{b}^\alpha \neq \mathbf{c}^\alpha$ such that $\tau(\{\mathbf{b}^\alpha : \alpha \in \mathbb{D}\}) = \tau(\{\mathbf{c}^\alpha : \alpha \in \mathbb{D}\})$. This can also be seen easily in (27) when introducing arbitrary orthogonal matrices and their respective inverses in between the component tensors.

An illustration of the tensor train structure (26) is depicted in Figure 2 (right), which is equivalent to the tree structure shown on the left-hand side.

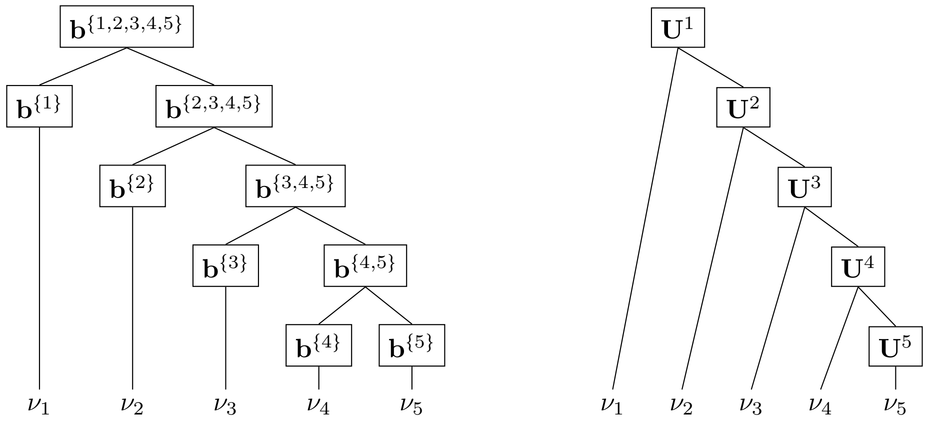


Figure 2: An order 5 tensor in tensor train representation and its linear representation using component tensors as in (26).

It turns out that every tensor has a TT-representation with minimal rank, which means that the TT-rank is well-defined. Moreover, an efficient algorithm for computing a minimal TT-representation is given by the TT Singular Value Decomposition (TT-SVD) [HRS12b]. Additionally, the set of tensor trains with fixed TT-rank $\mathbf{r}$ denoted by $\mathcal{T}_\mathbf{r} \subseteq \mathcal{H}_d$ forms a smooth manifold. If all lower ranks are included, an algebraic variety denoted by $\mathcal{T}_{\leqslant \mathbf{r}}$ is formed [Kut17].

3.3 Tensor Trains as differentiable manifolds

The multilinear structure of the tensor product enables efficient optimization within the manifold structure. Endowed with the Euclidean metric induced by the Frobenius scalar product, the set $\mathcal{T}_\mathbf{r}$ becomes an embedded Riemannian manifold [HRS11; UV20; Wol19]. This allows the formulation of different line search algorithms utilizing the *Riemannian gradient*. For a function $J : \mathcal{H}_n \to \mathbb{R}$ the Riemannian gradient at $X \in \mathcal{T}_r$ can be computed by projecting the Euclidean gradient onto the tangent space $\mathbb{T}_X$ at X (see e.g. [Ste16; AMS08]), i.e.

$$P_{\mathbb{T}_X} \nabla J(X), \tag{30}$$

where $P_{\mathbb{T}_{\widetilde{X}}}$ is the projector onto the tangent space of $\mathcal{T}_r$ at the point $\widetilde{X}$. Just as the negative Euclidean gradient, the negative Riemannian gradient can be used as a descent direction for minimizing $V^m_{p,N}$. In theory, the strategy is to move in that direction along a geodesic until a local minimum is reached. Starting from $\widetilde{X}$, the function that moves in the direction $Z \in \mathbb{T}_{\widetilde{X}}$ along a geodesic for a distance of $\|Z\|$ is called the *exponential map* $\exp_{\widetilde{X}}(Z)$. Unfortunately, there is no analytic expression for the exponential map available for $\mathcal{T}_r$. Instead, one usually resorts to a so-called *retraction* $\mathcal{R}_{\widetilde{X}}(Z)$ which is an approximation of the exponential map, see [AMS08] for details. In the tensor train format, an example of a retraction is defined by the TT-SVD via

$$\mathcal{R}_{\widetilde{X}}(Z) = \text{TT-SVD}(\widetilde{X} + Z) \tag{31}$$

as shown by [Ste16]. Using these techniques, a steepest descent update with step size β on the manifold $\mathcal{T}_r$ is given by

$$\widetilde{X}_{k+1} = \mathcal{R}_{\widetilde{X}_k}(-\beta P_{\mathbb{T}_{\widetilde{X}_k}} \nabla V^m_{p,N}(\widetilde{X}_k)). \tag{32}$$

Convergence of Riemannian optimization algorithms is typically only considered for smooth functions. When this can be assumend, the convergence can be sped up by using higher-order algorithms such as the conjugated gradient method. This additionally requires a method of “moving” tangent vectors $Z_{k-1} \in \mathbb{T}_{\widetilde{X}_{k-1}}$ from the tangent space at point $\widetilde{X}_{k-1}$ to the tangent space $\mathbb{T}_{\widetilde{X}_k}$ at point $\widetilde{X}_k$. Again, the optimal differential geometric tool, the *parallel transport*, is computationally infeasible on the tensor train manifold. However, the *vector transport* introduced by [AMS08] defines a class of approximations, which can be used to accomplish this task. In the tensor train format, such a vector transport is given by the projection $P_{\mathbb{T}_{\widetilde{X}_k}} Z_{k-1}$.

4 A version of the Longstaff-Schwartz algorithm based on the Tensor Train format

We now combine the tensor train format introduced in Section 3.2 with the Longstaff-Schwartz algorithm for computing Bermudan option prices as detailed in Algorithm 1. To make the approximation problem (8) concrete a set of basis functions $\{\mathbf{B}_\alpha\}_{\alpha\in\Lambda}$ has to be chosen. We prefer to work on a compact subdomain of the reals, which we choose such that the probability of assets lying outside the domain is minimal. As a heuristic method for determining the truncation, we set

$$a = \min_{m,n,k}(S^m_n)_k \qquad \text{and} \qquad b = \max_{m,n,k}(S^m_n)_k$$

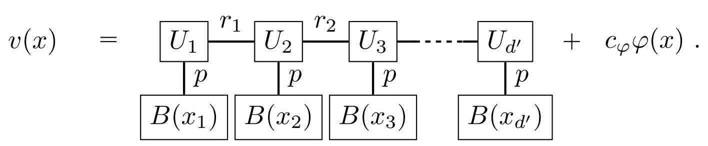
and choose the $H^2(a,b)$-orthogonal basis functions $\{B_1,\dots,B_p\}$ spanning the space of polynomials of degree p. We then represent the approximation of the discounted value of the option $v:\mathbb{R}^{d'}\to\mathbb{R}$ by

$$v(x)=\sum_{\alpha\in\Lambda}V_\alpha\prod_{k=1}^{d'}B_{\alpha_k}(x_k), \tag{33}$$

where we approximate the coefficient tensor $\mathbf{V}\in(\mathbb{R}^p)^{\otimes d'}$ in the TT format. As is common practice in Longstaff-Schwartz type algorithms we augment this basis by the payoff function φ. With the definition

$$B:\mathbb{R}\to\mathbb{R}^p,\quad B(x)=[B_1(x),\dots,B_p(x)],$$

i.e. B stacks the one-dimensional basis functions into a vector such that they can be contracted with the component tensors, the resulting approximation $v:\mathbb{R}^{d'}\to\mathbb{R}$ is graphically represented by



Note that on the r.h.s. of this equation every open-index of U_i and $B(x_i)$ for $1\leqslant i\leqslant d'$ is contracted, which indeed results in a scalar value $v(x)$.

To solve the resulting minimization problem (8) we use a rank adaptive version of the *alternating least-squares (ALS)* algorithm [HRS12a], the *stable alternating least-squares algorithm (SALSA)* [GK19]. Using this algorithm relieves us from having to guess an appropriate rank of the solution beforehand. As a termination condition we check whether the error on the samples or on a validation set decreases sufficiently during one iteration. In our implementation this validation set is chosen to have 20% of the size of the training set.

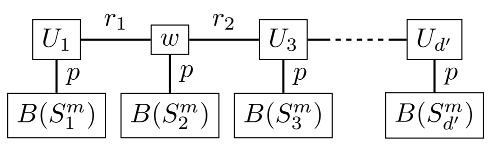
We now describe how we modify ALS (or SALSA) to handle the additional term $c_\varphi\varphi(x)$. The classical ALS algorithm optimizes the component tensors $\{U_1,\dots,U_{d'}\}$ in an alternating fashion. For each $k=1,\dots,d'$ all component tensors $\{U_j\}_{j\neq k}$ are fixed and only U_k is optimized. This procedure is then repeated alternatingly until a convergence criterion is met.

We modify this scheme by optimizing c_φ as well as U_k for each k. Since the mapping $(U_k,c_\varphi)\mapsto v$ is linear, the resulting problem is a classical linear least squares problem

$$(U_k,c_\varphi)=\operatorname*{arg\,min}_{w,c}\frac{1}{m}\sum_{i=1}^{m}|Y^m-A_k^m(w,c)|^2.$$

To exemplify this, for $k=2$ the operator A_k^m is diagrammatically represented by

$A_k^m(w, c_\varphi) =$



$+ \; c_\varphi \varphi(S^m)$.

After reshaping the pair $(w, c) \in \mathbb{R}^{r_1 \times p \times r_2} \times \mathbb{R}$ into a vector of size $r_1 p r_2 + 1$, the operator can be written as $A \in \mathbb{R}^{m \times (r_1 p_2 r_2 + 1)}$ and the problem becomes

$$X = \arg\min_x \frac{1}{m} \|\mathbf{Y} - Ax\|_2^2,$$

where $\mathbf{Y} = [Y^1, \dots, Y^m]$.

Complexity analysis

Using a tensor train representation instead of the full tensor allows us to reduce the space complexity from $\mathcal{O}(p^{d'})$ to $\mathcal{O}(d' p r^2)$ with $r = \max\{r_1, \dots, r_{d'-1}\}$. For moderate r this leads to a dramatic reduction in memory usage which we observe in our experiments. Figure 3 shows that the rank-adaptive algorithm computes solutions with $r < 6$ and we numerically verify that for $d' > 100$ a rank of $r = 1$ is sufficient for obtaining values within the reference interval from the literature. This allows us to compute the price of max-call options with up to 1000 assets.

Since ALS is an iterative method its time complexity can only be provided per iteration and amounts to

$$\mathcal{O}(Nm|\Lambda_p|^2 r^4) \tag{34}$$

floating point operations per iteration. As with every iterative algorithm the number of iterations needed depends on the specific problem. In our numerical tests we generally needed less than 10 iterations.

5 Dual martingale minimization with tensor trains

To use the tensor train format in the dual formulation, we define the set $\mathcal{P}_{\hat{0}} = \{\widetilde{X} : \widetilde{X}_0 = 0\}$ and rewrite (13) as

$$U_0 = \inf_{\widetilde{X} \in \mathcal{T}_r \cap \mathcal{P}_{\hat{0}}} V_{p,N}^m(\widetilde{X}), \tag{35}$$

where $\mathcal{T}_r$ denotes the set of TT tensors of rank r and $V_{p,N}^m$ is the cost function that is minimized in (13). Performing this optimization directly on the parameters of the tensor train is ill-posed since its parametrization is not unique. A common way to solve this is to use the manifold structure of $\mathcal{T}_r$ and employ a Riemannian optimization algorithm. For this (35) has to be rephrased as an unconstrained smooth optimization problem.

Define the projector $(P_{\hat{0}} \widetilde{X})_\alpha = (1 - \delta_{\alpha 0}) \widetilde{X}_\alpha$ and remove the constraint $\widetilde{X} \in \mathcal{P}_0$ by rewriting (35) as

$$U_0 = \inf_{\widetilde{X} \in \mathcal{T}_r} V_{p,N}^m(P_{\hat{0}} \widetilde{X}). \tag{36}$$

Since $P_{\hat{0}}$ is a linear operator, the modified cost function $V_{p,N}^m \circ P_{\hat{0}}$ retains the convexity, continuity and piece-wise linearity of $V_{p,N}^m$. We then mollify the $V_{p,N}^m$ by replacing the maximum with the smooth approximation

$$\underset{n=1,\dots,N}{\alpha\text{-max}}\, x_n = \frac{\sum_{n=1}^N x_n e^{\alpha x_n}}{\sum_{n=1}^N e^{\alpha x_n}}. \tag{37}$$

The resulting cost function reads

$$V_{p,N}(\widetilde{X}) = V_{p,N}^{m,\alpha}(\widetilde{X}) = \frac{1}{m}\sum_{i=1}^m \left[\underset{n=1,\dots,N}{\alpha\text{-max}} \left(Z_{t_n}^{(i)} - \sum_{\alpha\in\Lambda^n} \widetilde{X}_\alpha H_{\alpha_1}(G_1^{(i)}) \cdots H_{\alpha_n}(G_n^{(i)}) \right) \right]. \tag{38}$$

The respective optimization problem

$$U_0 = \inf_{\widetilde{X}\in\mathcal{T}_r} V_{p,N}^{m,\alpha}(P_{\hat{0}}\widetilde{X}) \tag{39}$$

can be solved by Riemannian algorithms. We use a conjugated gradient method with the FR-PR$_+$ update rule as defined in [NW06].

We also have to address the choice of the initial value for the optimization. Since the set $\mathcal{T}_r$ is not convex, a diligent choice is important in order to reach the global minimum. We obtain such a value for polynomial degree p by using the optimal value $\widetilde{X}^{(p-1)}$ for the polynomial degree $p-1$. This recursion stops at $p = 0$ where we know the optimal value to be $\widetilde{X}^{(0)} = 0$.

In our implementation we used a constant rank of 4 and chose $\alpha = 50$ which, empirically, held the smoothing induced error below 10^{-3}. As a termination condition we check if the error does not sufficiently decrease over a period of 10 iterations. Of all iterates obtained during the optimization we choose the one that has the lowest value on a validation set. In our implementation this validation set is chosen to have one ninth of the size of the training set.

Complexity analysis

In the dual method we observe the same dramatic reduction in space complexity as in the primal algorithm. The space complexity of $\mathcal{O}(p^{Nd'})$ for the full tensor is reduced to $\mathcal{O}(Nd'pr^2)$ for a tensor in the tensor train format with a rank uniformly bounded by r. This allows us to use the dual algorithm to compute the price of a basket put option with $N = 31$ exercise dates in Table 1.

Since gradient descent is again an iterative algorithm the time complexity can only be computed per iteration. Assuming that $\widetilde{X}$ is a tensor train tensor with rank r, the contraction

$$\sum_{\alpha\in\Lambda} \widetilde{X}_\alpha H_{\alpha_1}(G_1^{(i)}) \cdots H_{\alpha_n}(G_n^{(i)}) \tag{40}$$

can be computed with $\mathcal{O}(n|\Lambda_p|r^2 + (N-n)r^2)$ floating point operations. This means that both $V_{p,N}^{m,\alpha}(\widetilde{X})$ and its gradient can be computed with $\mathcal{O}(mN^2|\Lambda_p|r^2)$

floating point operations. Compare this to the $\mathcal{O}(mp^{Nd'})$ floating point operations required for the full tensor and to the $\binom{Nd'+p}{Nd'}$ operations for the sparse tensor. At least from a theoretical point of view, evaluation and optimization are faster in the tensor train format, namely

- exponentially faster when compared to the full tensor ansatz and
- when $p > 2$ up to a polynomial factor for the sparse ansatz.

These statements obviously depend on the rank r which is bounded by at most 4 in our experiments, meaning that the represented objects are in fact low-rank.

6 Numerical experiments

In this part, we present results obtained from the algorithms described above. Implementations in Python can be found at https://github.com/ptrunschke/tensor_option_pricing. For each experiment, we report low-biased estimators $v_0(S_{t_0})$ and $V_{p,N}^{m,\alpha}(\widetilde{X})$ based on re-simulated trajectories, see [Gla13]. More precisely, we generate independent trajectories of the underlying price process S and apply the stopping strategy implied by the already computed approximate value functions v_k, giving as a low-biased approximation to the true option price. Conversely, approximately optimal martingale parameterizations computed by the dual algorithm are used to compute a high-biased estimator, once again based on new trajectories, not used to produce the parameterization in the first place.

In the following we denote by n the number of possible exercise dates, including 0, by p the polynomial degree used in the approximation of the conditional probabilities and in the Wiener–Itô chaos expansion and by m the number of samples used. We further denote by m_{resim} the number of samples used for the resimulation. V_{LS} is the price computed by the resimulation of the Longstaff–Schwartz method and V_{dual} is the price computed by the dual method. The corresponding reference values are denoted by $V_{\text{LS}}^{\text{ref}}$ and $V_{\text{dual}}^{\text{ref}}$ respectively, and were obtained in the literature – see specific references for the individual examples.

6.1 Options in the Black–Scholes model

The d-dimensional Black Scholes model for $j \in \{1, \dots, d\}$ reads

$$dS_t^j = S_t^j(r_t - \delta_t)dt + \sigma^j L_j dB_t), \tag{41}$$

where B is a Brownian motion with values in $\mathbb{R}^d$, $\sigma = (\sigma_1, \dots, \sigma_d)$ is the vector of volatilities assumed to be deterministic and positive at all times, and L_j is the j-th row of the matrix L defined as a square root of the correlation matrix chosen to be of the form

$$\Gamma = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho \\ \rho & \cdots & \rho & 1 \end{pmatrix}, \tag{42}$$

where $\rho \in (-1/(d-1), 1]$ to ensure that Γ is positive definite. The initial condition for the SDE is given by the spot price S_0.

We will test the algorithms for different payoff functions ϕ, dimensions d and strike prices K.

6.2 A basket put option on correlated assets

We first consider the case of a put basket option on correlated assets. The payoff of this option writes as $\phi(S_t) = \left(K - \sum_{j=1}^{d} \omega_j S_t^j\right)_+$ where $\omega = (\omega_1, \ldots, \omega_d)$ is a vector of real valued weights. We report in Table 1 and Table 2 our values compared to the reference prices for two different sample sizes $m = 20000$ and $m = 10^5$. Blank cells in the tables indicate that reference values are not reported in the reference papers. The results of our experiments are reported in Table 1.

It can be seen that the values obtained by our version of Lelong's method are not as close to the reference price as are the values obtained by [Lel18]. From a theoretical perspective a lower value should always be possible given a sufficient rank. We thus attribute this to the lack of a rank adaption strategy in the dual problem and highlight this as an interesting direction for further research. It can moreover be seen that for $N = 31$ the values of V_{dual} increase with p. Because the manifold for $p = 2$ is a submanifold of $p = 3$ one would expect that this is impossible. Note however that the table shows resimulated prices only. Therefore we interpret this observation to indicate that a larger value of m is needed in this case. This is confirmed in Table 2.

For the Longstaff-Schwartz variant we use $m = 10^5$ and observe values close to the reference value. Furthermore, in the case $N = 31$ and $S_0^j = 100$, we observe that the result for $p = 2$ dominate the $p = 3$ case, indicating sub-optimal results. However, as seen in Table 2 we obtain better results for polynomial degree $p = 8$. Note that we have capped the TT-rank at 4 for the computation with $p = 8$. By doing that, the computational time only increased by a factor of 3 when compared to the run time for the case $p = 3$, being 40 seconds and 15 seconds respectively.

We also report that during the optimization within the Longstaff-Schwartz algorithm the TT-rank of the value function did not exceed 5 for any test-case, which means that a low-rank structure of the sought expectation values within the polynomial ansatz space is noticeable. This low-rank structure is a necessity for high-dimensional computation and will be analyzed in greater detail in the next example. In this example the number of samples used for training has a larger effect not only on the variances but also on the values.

p	N	S_0^j	V_{dual}	Stddev	$V_{\text{dual}}^{\text{ref}}$	V_{LS}	Stddev	V^{ref}
2	4	100	2.34	0.003	2.29	2.15	0.009	2.17
3	4	100	2.33	0.003	2.25	2.16	0.009	2.17
2	7	100	2.64	0.002	2.62	2.39	0.008	2.43
3	7	100	2.64	0.002	2.52	2.40	0.008	2.43
2	31	100	3.08	0.002		2.49	0.01	
3	31	100	3.12	0.002		2.36	0.01	
2	4	110	0.67	0.002	0.57	0.53	0.006	0.55
3	4	110	0.67	0.002	0.55	0.53	0.006	0.55
2	7	110	0.78	0.002	0.64	0.57	0.007	0.61
3	7	110	0.77	0.002	0.64	0.57	0.007	0.61
2	31	110	3.94	0.002		0.61	0.008	
3	31	110	3.95	0.002		0.61	0.008	

Table 1: Prices for the put basket option with parameters $d = 5$, $T = 3$, $r = 0.05$, $\delta^j = 0$, $\sigma^j = 0.2$, $\rho = 0$, $K = 100$, $\omega_j = \frac{1}{d}$, $m = 20000$, $m_{\text{resim}} = 10^6$. Values for $V_{\text{dual}}^{\text{ref}}$ and V^{ref} are taken from [Lel18]. Number of samples for Longstaff-Schwartz: $m_{\text{LS}} = 10^5$. Empty spaces denote unavailable reference values.

p	S_0^j	V_{dual}	Stddev
2	100	2.88	0.001
3	100	2.88	0.001
2	110	0.80	0.001
3	110	0.80	0.001

p	S_0^j	V_{LS}	Stddev
8	100	2.56	0.01

Table 2: Prices for the put basket option with parameters $d = 5$, $N = 31$, $T = 3$, $r = 0.05$, $\delta^j = 0$, $\sigma^j = 0.2$, $\rho = 0$, $K = 100$, $\omega_j = \frac{1}{d}$, $\mathbf{m = 10^5}$, $m_{\text{resim}} = 10^6$. Empty spaces denote unavailable reference values

6.3 Bermudan max-call options

In this section we consider max-call options and in particular the scalability of the tensor train approach for the Longstaff-Schwartz algorithm for higher dimensions. The reference values for this problem were taken from [AB04; BCJ19]. The payoff function of a max-call option takes the form

$$\left(\max_{1\leqslant i\leqslant d} \omega_i S^i - K\right)_+ . \tag{43}$$

In Table 3 we report results for the dual algorithm. In contrast to the case of the put basket option, we see that we are close to the values computed by the original method [Lel18] and in some cases improve the previously reported results. This indicates the viability of this approach. A rank-adaptive algorithm could probably further improve the efficiency of our method in high dimensions.

In Table 4 we consider the Longstaff-Schwartz algorithm in moderate to extreme dimensions. We increase the number of samples to 10^6 and test every

p	d	m	S_0^j	V_{dual}	Stddev	$V_{\text{dual}}^{\text{ref}}$	V^{ref}
2	2	20000	90	8.85	0.004	10.05	8.15
3	2	20000	90	8.83	0.004	8.6	8.15
2	5	20000	90	21.68	0.014	21.2	16.77
3	5	40000	90	21.40	0.015	20.13	16.77
2	2	20000	100	14.68	0.004	16.3	14.01
3	2	20000	100	14.65	0.004	15	14.01
2	5	20000	100	32.37	0.017	31.8	26.34
3	5	40000	100	31.95	0.017	29	26.34

Table 3: Prices for the call option on the maximum of d assets with parameters $N = 10$, $T = 3$, $r = 0.05$, $\delta^j = 0.1$, $\sigma^j = 0.2$, $\rho = 0$, $K = 100$, $m_{\text{resim}} = 10^6$. Values for $V_{\text{dual}}^{\text{ref}}$ and V^{ref} are taken from [Lel18].

polynomial degree up to $p = 7$. We observe that we rarely see any significant improvement when using polynomial degree larger than 4 or 5. However, throughout the table polynomial degree $p = 6$ appears to obtain the overall best results, with small improvements over the other polynomial degrees. Moreover, we see that while we are not exactly as high as the reference value for low dimensions, i.e. $d \leqslant 20$, the results for higher dimension are accurate. A possible explanation for this is that the value function might have simpler structure in high dimension.

Finally, in Table 5 we use a trick, where after sampling all the paths, we sort the assets at every time point by decreasing magnitude, see, e.g., [AB04, p. 1230]. We observe, that , the unsorted algorithm performs better than the sorted, while both stay closely under the reference interval. We observe, that while the unsorted algorithm is already performing well, sorting the assets yields an increase in performance in every dimension. Moreover, for the sorted case, polynomial degree of 3 appears to be sufficient to obtain optimal results. Finally, we observe some numerical instabilities for our implementation of the sorted algorithm when the dimension is $d = 750$ or $d = 1000$ and the polynomial degree is larger than 3. We assume that by using a better polynomial basis these instabilities can be resolved. However, as polynomial degree 3 was sufficient in the lower-dimensional case we did not further investigate this instability. We state that within these experiments the standard deviation of the resimulations was never larger than 0.1.

It is worth noting, that the results in very high dimensions were obtained by calculating only 10^6 trajectories while the reference values were computed using more than 24×10^6 paths using state-of-the-art machine learning techniques, see [BCJ19]. This underlines the potential of tensor train approaches for optimal stopping, especially in high dimensions.

In Figure 3 we analyze the average and the maximal rank of the value function and observe a decrease of the ranks in higher dimensions. We state that from $d = 100$ a separate test run where we fix the ranks to 1 yield comparable results, implying that a rank 1 solution can yield close to optimal results. This means, that the value function indeed has a simple structure in high dimension.

d	p							V_{ref}
	1	2	3	4	5	6	7	
2	13.66	13.79	13.81	13.76	13.80	13.83	13.78	13.902
3	18.34	18.30	18.39	18.48	18.50	18.55	18.53	18.69
5	25.66	25.58	25.70	25.97	25.75	25.84	25.93	$[26.115, 26.164]$
10	37.77	37.65	38.01	38.12	38.25	38.27	38.14	$[38.300, 38.367]$
20	51.10	51.34	51.49	51.64	51.62	51.63	51.62	$[51.549, 51.803]$
30	59.11	59.30	59.50	59.63	59.62	59.63	59.63	$[59.476, 59.872]$
50	69.22	69.23	69.70	69.56	69.57	69.51	69.57	$[69.560, 69.945]$
100	83.14	83.18	83.29	83.33	83.37	83.39	83.16	$[83.357, 83.862]$
200	97.21	97.07	97.31	97.43	97.41	97.46	97.21	$[97.381, 97.889]$
500	116.13	116.07	116.17	116.31	116.31	116.36	116.14	$[116.210, 116.685]$
750	124.56	124.56	124.61	124.72	124.73	124.78	124.59	
1000	130.65	130.63	130.66	130.78	130.83	130.84	130.67	

Table 4: $n = 9$, $T = 3$, $r = 0.05$, $\delta = 0.1$, $\sigma = 0.2$, $\rho = 0$, $S_0^j = 100$, $K = 100$, $\omega_j = 1$, $m = 10^6$, $m_{\text{resim}} = 10^6$ not using reordering
From: [AB04; BCJ19]

d	p							V_{ref}
	1	2	3	4	5	6	7	
2	13.67	13.76	13.82	11.63	13.84	13.84	13.85	13.902
3	18.39	18.51	18.60	18.61	18.61	18.62	18.62	18.69
5	25.83	26.01	26.06	26.07	26.07	26.07	26.07	$[26.115, 26.164]$
10	38.08	38.24	38.29	38.31	38.31	38.30	38.30	$[38.300, 38.367]$
20	51.48	51.66	51.71	51.71	51.71	51.71	51.71	$[51.549, 51.803]$
30	59.50	59.68	59.71	59.71	59.72	59.72	59.72	$[59.476, 59.872]$
50	69.58	69.78	69.80	69.81	69.81	69.81	69.81	$[69.560, 69.945]$
100	83.45	83.65	83.67	83.67	83.67	83.66	83.66	$[83.357, 83.862]$
200	97.56	97.69	97.70	97.70	97.70	97.69	97.69	$[97.381, 97.889]$
500	116.45	116.56	116.56	116.56	116.56	116.50	116.52	$[116.210, 116.685]$
750	124.91	124.98	124.99	124.98	nan	nan	nan	
1000	130.96	131.06	131.05	nan	nan	nan	nan	

Table 5: $n = 9$, $T = 3$, $r = 0.05$, $\delta = 0.1$, $\sigma = 0.2$, $\rho = 0$, $S_0^j = 100$, $K = 100$, $\omega_j = 1$, $m = 10^6$, $m_{\text{resim}} = 10^6$ using reordering
From: [AB04; BCJ19]

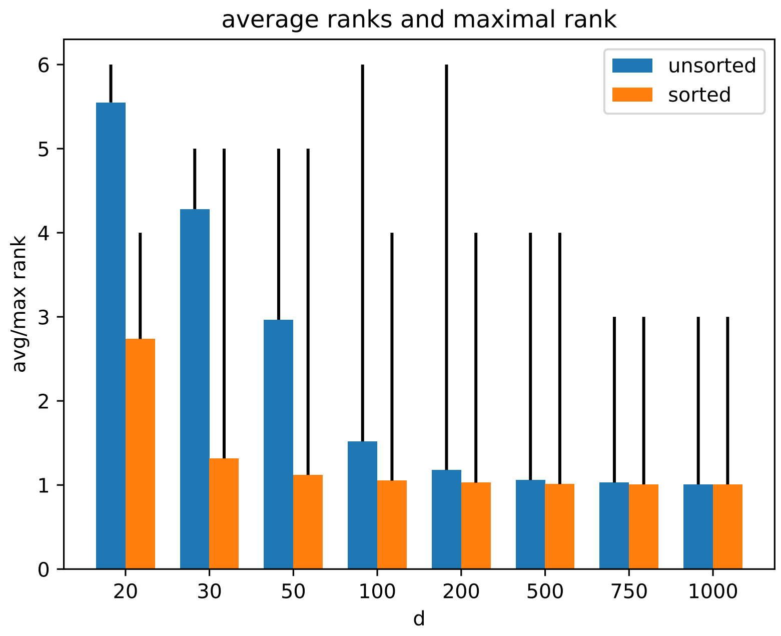


Figure 3: average ranks(blue) and maximal(black) rank for different dimension. The black lines indicate the maximal rank. We use the results with highest values for every dimension and every bar.

Acknowledgements

Christian Bayer gratefully acknowledges support by the DFG cluster of excellence MATH+, project AA4-2. 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 (DAEDALUS)” (GRK 2433) funded by the German Research Foundation (DFG). Philipp Trunschke acknowledges support by the Berlin International Graduate School in Model and Simulation based Research (BIMoS). We thank Max Pfeffer for fruitful discussions.

References

[AB04] Leif Andersen and Mark Broadie. “Primal-dual simulation algorithm for pricing multidimensional American options”. In: *Management Science* 50.9 (2004), pp. 1222–1234.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Dec. 2008. DOI: 10.1515/9781400830244. URL: https://doi.org/10.1515%2F9781400830244.

[AP05] Yves Achdou and Olivier Pironneau. *Computational methods for option pricing*. SIAM, 2005.

[BCD17] Markus Bachmayr, Albert Cohen, and Wolfgang Dahmen. "Parametric PDEs: sparse or low-rank approximations?" In: *IMA Journal of Numerical Analysis* 38.4 (Sept. 2017), pp. 1661–1708. DOI: 10.1093/imanum/drx052. URL: https://doi.org/10.1093%2Fimanum%2Fdrx052.

[BCD18] Markus Bachmayr, Albert Cohen, and Wolfgang Dahmen. "Parametric PDEs: sparse or low-rank approximations?" In: *IMA Journal of Numerical Analysis* 38.4 (2018), pp. 1661–1708.

[BCJ19] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. "Deep optimal stopping". In: *Journal of Machine Learning Research* 20 (2019), p. 74.

[BD16] Markus Bachmayr and Wolfgang Dahmen. "Adaptive low-rank methods: Problems on Sobolev spaces". In: *SIAM Journal on Numerical Analysis* 54.2 (2016), pp. 744–796.

[BFG16] Christian Bayer, Peter Friz, and Jim Gatheral. "Pricing under rough volatility". In: *Quantitative Finance* 16.6 (2016), pp. 887–904.

[BS18] Denis Belomestny and John Schoenmakers. *Advanced Simulation-Based Methods for Optimal Stopping and Control: With Applications in Finance*. Springer, 2018.

[BSU16] Markus Bachmayr, Reinhold Schneider, and André Uschmajew. "Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations". In: *Foundations of Computational Mathematics* 16.6 (2016), pp. 1423–1472.

[BTW20] Christian Bayer, Raúl Tempone, and Sören Wolfers. "Pricing American options by exercise rate optimization". In: *Quantitative Finance* 20.11 (2020), pp. 1749–1760.

[DKK19] Sergey Dolgov, Dante Kalise, and Karl Kunisch. "Tensor decompositions for high-dimensional Hamilton-Jacobi-Bellman equations". In: *arXiv preprint arXiv:1908.01533* (2019).

[Eig+19] Martin Eigel et al. "Variational Monte Carlo – bridging concepts of machine learning and high-dimensional partial differential equations". In: *Advances in Computational Mathematics* 45.5-6 (2019), pp. 2503–2532.

[Eig+20] Martin Eigel et al. “Adaptive stochastic Galerkin FEM for lognormal coefficients in hierarchical tensor representations”. In: *Numerische Mathematik* 145.3 (June 2020), pp. 655–692. DOI: 10.1007/s00211-020-01123-1. URL: https://doi.org/10.1007%2Fs00211-020-01123-1.

[EPS17] Martin Eigel, Max Pfeffer, and Reinhold Schneider. “Adaptive stochastic Galerkin FEM with hierarchical tensor representations”. In: *Numerische Mathematik* 136.3 (2017), pp. 765–803.

[Fac+20] Konstantin Fackeldey et al. “Approximative Policy Iteration for Exit Time Feedback Control Problems driven by Stochastic Differential Equations using Tensor Train format”. In: *arXiv preprint arXiv:2010.04465* (2020).

[Gaß+18] Maximilian Gaß et al. “Chebyshev interpolation for parametric option pricing”. In: *Finance and Stochastics* 22.3 (2018), pp. 701–731.

[GK19] Lars Grasedyck and Sebastian Krämer. “Stable ALS approximation in the TT-format for rank-adaptive tensor completion”. In: *Numerische Mathematik* 143.4 (Aug. 2019), pp. 855–904. DOI: 10.1007/s00211-019-01072-4. URL: https://doi.org/10.1007%2Fs00211-019-01072-4.

[GKS20] Kathrin Glau, Daniel Kressner, and Francesco Statti. “Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing”. In: *SIAM Journal on Financial Mathematics* 11.3 (2020), pp. 897–927.

[Gla13] Paul Glasserman. *Monte Carlo methods in financial engineering.* Vol. 53. Springer Science & Business Media, 2013.

[Hac12] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-28027-6. URL: https://doi.org/10.1007%2F978-3-642-28027-6.

[HK09] Wolfgang Hackbusch and Stefan Kühn. “A New Scheme for the Tensor Representation”. English. In: *Journal of Fourier Analysis and Applications* 15.5 (2009), pp. 706–722. ISSN: 1069-5869. DOI: 10.1007/s00041-009-9094-9. URL: http://dx.doi.org/10.1007/s00041-009-9094-9.

[HRS11] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. “On manifolds of tensors of fixed TT-rank”. In: *Numerische Mathematik* 120.4 (Sept. 2011), pp. 701–731. DOI: 10.1007/s00211-011-0419-7. URL: https://doi.org/10.1007%2Fs00211-011-0419-7.

[HRS12a] S. Holtz, T. Rohwedder, and R. Schneider. “The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format”. In: *SIAM J. Sci. Comput.* 34.2 (2012), A683–A713. DOI: 10.1137/100818893. eprint: https://doi.org/10.1137/100818893. URL: https://doi.org/10.1137/100818893.

[HRS12b] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. "On manifolds of tensors of fixed TT-rank". In: *Numerische Mathematik* 120.4 (2012), pp. 701–731.

[HRS12c] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. "The alternating linear scheme for tensor optimization in the tensor train format". In: *SIAM J. Sci. Comput.* 34.2 (2012), A683–A713. ISSN: 1064-8275. URL: https://doi.org/10.1137/100818893.

[HS14] Wolfgang Hackbusch and Reinhold Schneider. "Tensor spaces and hierarchical tensor representations". In: *Extraction of Quantifiable Information from Complex Systems*. Springer, 2014, pp. 237–261.

[KSV14] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. "Low-rank tensor completion by Riemannian optimization". In: *BIT* 54.2 (2014), pp. 447–468. ISSN: 0006-3835. URL: https://doi.org/10.1007/s10543-013-0455-z.

[Kut17] Benjamin Kutschan. "Tangent cones to TT varieties". In: *arXiv preprint arXiv:1705.10152* (2017).

[Lel18] Jérôme Lelong. "Dual Pricing of American Options by Wiener Chaos Expansion". In: *SIAM Journal on Financial Mathematics* 9.2 (Jan. 2018), pp. 493–519. DOI: 10.1137/16m1102161. URL: https://doi.org/10.1137%2F16m1102161.

[Lel19] Jérôme Lelong. "Pricing path-dependent Bermudan options using Wiener chaos expansion: an embarrassingly parallel approach". In: *arXiv preprint arXiv:1901.05672* (2019).

[LS01] Francis A Longstaff and Eduardo S Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The review of financial studies* 14.1 (2001), pp. 113–147.

[Lud20] Mike Ludkovski. *mlOSP: Towards a Unified Implementation of Regression Monte Carlo Algorithms*. 2020. arXiv: 2012.00729 [q-fin.CP].

[Nou17] Anthony Nouy. "Low-rank methods for high-dimensional approximation and model order reduction". In: *Model reduction and approximation, P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, eds., SIAM, Philadelphia, PA* (2017), pp. 171–226.

[NW06] Jorge Nocedal and Stephen J. Wright. "Conjugate Gradient Methods". In: *Springer Series in Operations Research and Financial Engineering*. Springer New York, 2006, pp. 101–134. DOI: 10.1007/978-0-387-40065-5_5. URL: https://doi.org/10.1007%2F978-0-387-40065-5_5.

[Ose11] I. Oseledets. "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317. DOI: 10.1137/090752286. eprint: http://dx.doi.org/10.1137/090752286. URL: http://dx.doi.org/10.1137/090752286.

[Ose13] Ivan V Oseledets. “Constructive representation of functions in low-rank tensor formats”. In: *Constructive Approximation* 37.1 (2013), pp. 1–18.

[OSS19] Mathias Oster, Leon Sallandt, and Reinhold Schneider. “Approximating the Stationary Hamilton-Jacobi-Bellman Equation by Hierarchical Tensor Products”. In: *arXiv preprint arXiv:1911.00279* (2019).

[OT09] I. V. Oseledets and E. E. Tyrtyshnikov. “Breaking the curse of dimensionality, or how to use SVD in many dimensions”. In: *SIAM J. Sci. Comput.* 31.5 (2009), pp. 3744–3759. ISSN: 1064-8275. URL: https://doi.org/10.1137/090748330.

[OT10] Ivan V. Oseledets and Eugene Tyrtyshnikov. “TT-cross approximation for multidimensional arrays”. In: *Linear Algebra and its Applications* 432.1 (2010), pp. 70–88.

[Rog02] L. C. G. Rogers. “Monte Carlo valuation of American options”. In: *Mathematical Finance* 12.3 (July 2002), pp. 271–286. DOI: 10.1111/1467-9965.02010. URL: https://doi.org/10.1111%2F1467-9965.02010.

[RSS17] Holger Rauhut, Reinhold Schneider, and Željka Stojanac. “Low rank tensor recovery via iterative hard thresholding”. In: *Linear Algebra and its Applications* 523 (2017), pp. 220–262.

[Ste16] Michael Maximilian Steinlechner. “Riemannian Optimization for Solving High-Dimensional Problems with Low-Rank Tensor Structure”. en. In: (2016). DOI: 10.5075/EPFL-THESIS-6958. URL: http://infoscience.epfl.ch/record/217938.

[UV20] A. Uschmajew and B. Vandereycken. “Geometric methods on low-rank matrix and tensor manifolds”. In: *Variational methods for nonlinear geometric data and applications*. Ed. by P. Grohs, M. Holler, and A. Weinmann. Springer, 2020. DOI: 10.1007/978-3-030-31351-7_9.

[Vid03] Guifré Vidal. “Efficient Classical Simulation of Slightly Entangled Quantum Computations”. In: *Phys. Rev. Lett.* 91 (14 Oct. 2003), p. 147902. DOI: 10.1103/PhysRevLett.91.147902. URL: https://link.aps.org/doi/10.1103/PhysRevLett.91.147902.

[Wol19] Alexander Sebastian Johannes Wolf Wolf. “Low rank tensor decompositions for high dimensional data approximation, recovery and prediction”. en. In: (2019). DOI: 10.14279/DEPOSITONCE-8109. URL: https://depositonce.tu-berlin.de/handle/11303/8986.

[WS05] Xiaoqun Wang and Ian H Sloan. “Why are high-dimensional finance problems often of low effective dimension?” In: *SIAM Journal on Scientific Computing* 27.1 (2005), pp. 159–183.