

# LOW-RANK TENSOR APPROXIMATION FOR CHEBYSHEV INTERPOLATION IN PARAMETRIC OPTION PRICING\*

KATHRIN GLAU<sup>†</sup>, DANIEL KRESSNER<sup>‡</sup>, AND FRANCESCO STATTI<sup>§</sup>

**Abstract.** Treating high dimensionality is one of the main challenges in the development of computational methods for solving problems arising in finance, where tasks such as pricing, calibration, and risk assessment need to be performed accurately and in real-time. Among the growing literature addressing this problem, Gass et al. [14] propose a complexity reduction technique for parametric option pricing based on Chebyshev interpolation. As the number of parameters increases, however, this method is affected by the curse of dimensionality. In this article, we extend this approach to treat high-dimensional problems: Additionally exploiting low-rank structures allows us to consider parameter spaces of high dimensions. The core of our method is to express the tensorized interpolation in tensor train (TT) format and to develop an efficient way, based on tensor completion, to approximate the interpolation coefficients. We apply the new method to two model problems: American option pricing in the Heston model and European basket option pricing in the multi-dimensional Black-Scholes model. In these examples we treat parameter spaces of dimensions up to 25. The numerical results confirm the low-rank structure of these problems and the effectiveness of our method compared to advanced techniques.

**Key words.** Chebyshev interpolation, parametric option pricing, high-dimensional problem, tensor train format, low-rank tensor approximation, tensor completion

**1. Introduction.** Financial problems are, by their nature, multi- and high-dimensional, because a large number of risk factors contribute to the prices of each financial asset. Moreover, the banking, insurance and hedge fund industry draws on investments in large portfolios. The interdependencies of both the risk factors and the assets make basic computational tasks such as model calibration, pricing, and hedging as well as more global tasks such as uncertainty quantification, risk assessment and capital reserve calculation computationally extremely challenging, see for instance [5].

Automatic and high-speed trading challenge the computational methods in that the results need to be available fast and with minimal storage requirement. Moreover, we observe rising regulatory requirements. On the one hand, more realistic modeling demands more prudent considerations, which leads to rising computational complexity. On the other hand, the availability of requested performance characteristics is expected to be delivered within shorter periods of time. This poses a high challenge for traditional approaches, which typically suffer from low convergence rates in higher dimensions, see for instance [9, 12].

For the reasons explained above, the development of efficient computational methods for high-dimensional problems in finance is an utmost active field of research in both academia and industry. For example, further developments of the Monte Carlo method have been very successfully applied to financial problems; we refer to [35, 16] for the quasi Monte Carlo method and to [15] for the multilevel Monte Carlo method. Besides stochastic integration, deterministic numerical integration has been exploited

---

\*The authors would like to thank Jonas Ballani for helpful discussions on this work.

<sup>†</sup>Queen Mary University of London, Mile End Road, E1 4NS London, United Kingdom, [k.glau@qmul.ac.uk](mailto:k.glau@qmul.ac.uk). This project has received funding from the European Unions Horizon 2020 research and innovation program under grant 665667.

<sup>‡</sup>École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland ([daniel.kressner@epfl.ch](mailto:daniel.kressner@epfl.ch), <http://anchp.epfl.ch>)

<sup>§</sup>École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland ([francesco.statti@epfl.ch](mailto:francesco.statti@epfl.ch), <http://people.epfl.ch/francesco.statti>). Research supported through the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307465-POLYTE.

using sparse grid techniques, see [19, 27, 6]. Also PDE methods have been extended to multivariate problems in finance. For instance using operator splitting methods as in [28], principal component analysis and expansions as in [42], and wavelet compression techniques proposed in [36, 25, 26].

Exploiting the particular structure of a problem, *complexity reduction techniques* exhibit great potential to save run-time and storage capacity while maintaining the required accuracy. In numerical analysis and a large variety of applications, for example in engineering and medicine, complexity reduction techniques have been developed and implemented with great success. For instance the field of reduced basis methods to efficiently solve parametric partial differential equations (PDEs) has experienced a tremendous development over the last decade, see, e.g., [23, 40, 41] and the references therein. Pioneered by [43, 10] the potential of reduced basis methods is also increasingly exploited for problems in finance; see [8, 37, 7] for examples. These methods can be viewed as high-dimensional interpolation methods that are trained in an offline step to solve a specific class of parametric PDEs. In this article we explore direct interpolation of multivariate functions as a *unified approach* to complexity reduction for finance.

Our starting point is the tensorized Chebyshev interpolation of conditional expectations in the parameter and state space, as introduced in [14]. Having observed for a large set of applications that these functions are highly regular, admitting sensitivities of high order or even being analytic, and that the domain of interest can be restricted to a hyperrectangular, Chebyshev interpolation is a promising choice: Its convergence is subexponential for multivariate analytic functions, its implementation is numerically stable, and the coefficients are simply given by a linear transformation of the function values at the nodal points. In this article we exploit this favorable structure further for high dimensionality. In passing, we point out that, while we choose Chebyshev interpolation for the reasons listed in this paragraph, the technique presented in this paper extends to other tensorized interpolation techniques. Also, our approach is applicable beyond option pricing and finance.

The basis of our approach is the following. In an *offline phase*, the price as function of parameters  $\mathbf{p} \in [-1, 1]^d$ ,  $\mathbf{p} \mapsto \text{Price}^{\mathbf{p}}$  is evaluated at selected parameter samples  $\mathbf{p}$  to prepare an approximation by tensorized Chebyshev polynomials  $T_{j_1, \dots, j_d}$  with pre-computed Fourier coefficients  $c_{j_1, \dots, j_d}$ , as follows,

$$(1.1) \quad \text{Price}^{\mathbf{p}} \approx \sum_{j_1=0}^{n_1} \cdots \sum_{j_d=0}^{n_d} c_{j_1, \dots, j_d} T_{j_1, \dots, j_d}(\mathbf{p}).$$

This polynomial approximation of the price function allows us to efficiently approximate the price during an *online phase*, for an arbitrary new set of parameters  $\mathbf{p}$  in the parameter domain. To evaluate the function in the online phase, only the multivariate polynomials on the right-hand side need to be evaluated. However, implementing (1.1) in a straightforward manner exposes the method to the curse of dimensionality in both the offline and the online phase: In the offline phase, the prices need to be evaluated on a tensorized grid of Chebyshev nodes, amounting to  $\mathcal{O}(n^d)$  parameter samples when  $n$  nodes are required for each parameter. This is computationally costly, especially if the underlying pricing method is already computationally demanding. In the online phase alike,  $\mathcal{O}(n^d)$  operations are needed for evaluating the approximating multivariate polynomial. Even for a number as low as  $n = 3$ , corresponding to quadratic polynomials, a problem with  $d = 20$  parameters becomes infeasible.

One approach to breaking the curse of dimensionality that has already proven

effective in a number of areas is to exploit low-rank structures of high-dimensional tensors; see [18, 20, 29] and the references therein. These techniques reduce, sometimes dramatically, memory requirements and the cost of operating with tensors. In the context of parametric PDEs, low-rank tensor structures have been successfully exploited in, e.g., [1, 4, 30, 34, 45]. As option prices are characterized as solutions of parabolic PDEs, this gives hope that low-rank structures can be exploited in finance as well. While Chebyshev interpolation is known to be most effective when applied to a sufficiently regular function, low-rank decompositions can be less sensitive to smoothness properties of the price function. Combining both promises to exploit different features of the price function to reduce complexity. The following questions arise:

*Can we detect low-rank structures for the problem of form (1.1)?* Existing theoretical studies only provide partial answers to this question, either not reflecting the observed effectiveness of low-rank techniques or being limited to rather specific function classes; see [11, 20, 44] for examples. We therefore approach the question from an experimental perspective and analyze examples of different nature and different dimensionality in Section 3. The results clearly indicate an approximate low-rank structure of the tensor  $\mathcal{P}$  containing the prices evaluated at the nodes of the tensorized Chebyshev grid, that is, the entries of  $\mathcal{P}$  are defined as

$$(1.2) \quad \mathcal{P}(k_1, \dots, k_d) = \text{Price}^{q_{k_1}, \dots, q_{k_d}},$$

where the tuple  $q_{k_1, \dots, k_d} := (q_{k_1}, \dots, q_{k_d})$  contains Chebyshev nodes  $q_{k_i} := \cos(\pi \frac{k_i}{n_i})$  for  $k_i = 0, \dots, n_i$  and  $i = 1, \dots, d$ . In the specific case of the interpolation of American option prices in the Heston model in five parameters we can explicitly compare the full tensor  $\mathcal{P}$  with the one resulting from low-rank approximation. We perform this comparison in Section 3.1, which confirms the low-rank structure of  $\mathcal{P}$ . In Section 3.2 we consider prices of basket options in the Black-Scholes model with up to 25 underlyings and interpolate in the initial values of the underlyings. Although the resulting full tensor  $\mathcal{P}$  is too large to be explicitly computed and compared with, we provide a structural analysis in Section 3.2.3 that explains why  $\mathcal{P}$  is expected to exhibit low-rank structure.

*How can we exploit low-rank structures for the problem of form (1.1)?* Expressing the problem in a tensor format reveals that exploiting the tensor structure itself (even without low-rank structure) leads to a considerable efficiency gain in both the offline and the online phase. Next, we explore existing low-rank tensor techniques. In order to efficiently exploit these techniques for problem (1.1), we need to introduce several new components resulting in the new method. We detail these steps below.

In order to construct the interpolation coefficients  $c_{j_1, \dots, j_d}$  in the offline phase, it is first required to compute or approximate all values of the tensor  $\mathcal{P}$ , containing the prices in the tensorized Chebyshev grid. Evaluating  $\mathcal{P}$  explicitly is too costly for larger  $d$ , especially when the underlying pricing procedure is computationally expensive. Instead we only compute part of the entries of  $\mathcal{P}$  and then need to deal with an incomplete tensor. This leads us to the following first step:

1. We start by computing the prices for a small portion of the Chebyshev grid points only. Then, we adapt a *completion algorithm* (in Section 2.3) which allows us to approximate the tensor of prices for the complete Chebyshev grid by fitting tensors of pre-specified low rank to the provided data points. As it is not reasonable to assume a priori knowledge of low-rank structure, the completion procedure needs to be combined with an *adaptive rank and sampling strategy*. Specifically, we repeat the process of adding new samples

and increasing the pre-specified rank until an adequate stopping criterion is fulfilled. This completion algorithm is designed to work with tensors built and stored in tensor train (TT) format.

With the low-rank approximation of the tensor  $\mathcal{P}$  in TT format at hand, we can then approximate efficiently the Fourier coefficients  $c_{j_1, \dots, j_d}$ . It is a special feature of the Chebyshev interpolation that the tensor  $\mathcal{C}$  with entries  $\mathcal{C}(j_1, \dots, j_d) := c_{j_1, \dots, j_d}$  is simply given as a linear transform of the prices evaluated at the nodal points,

$$(1.3) \quad \mathcal{C}(j_1, \dots, j_d) := \left( \prod_{i=1}^d \frac{2^{\mathbb{1}_{n_i > j_i > 0}}}{n_i} \right) \sum_{k_1=0}^{n_1} \dots \sum_{k_d=0}^{n_d} \mathcal{P}(k_1, \dots, k_d) \prod_{i=1}^d \cos\left(j_i \pi \frac{k_i}{n_i}\right),$$

where  $\sum''$  indicates that the first and the last summand are halved. This is computed in the last step of the offline phase:

2. (The computation of the tensor  $\mathcal{C}$ , containing the Fourier coefficients  $c_{j_1, \dots, j_d}$ .) ← The linearity of (1.3) allows us to compute  $\mathcal{C}$  by a sequence of  $d$  tensor-matrix multiplications. Moreover, the particular structure of the involved matrices facilitates the use of the fast Fourier transform, leading to a complexity of  $\mathcal{O}(dnr^2 \log(n))$ , where  $r$  is determined by the ranks of  $\mathcal{P}$ . This step is explained in Section 2.4.2.

Suppose now that, in the online phase, we want to compute the interpolated price (1.1) for a new set of parameter samples. Given the tensor  $\mathcal{C}$  in TT format, the evaluation of (1.1) for a price  $\mathbf{p}$  is performed efficiently as follows:

3. First, each of the Chebyshev polynomials involved in the tensorized Chebyshev basis is evaluated in  $\mathbf{p}$ . It turns out that (1.1) can be viewed as inner product between  $\mathcal{C}$  and a rank-one tensor. Thanks to the TT format, the complexity of computing this inner product is  $\mathcal{O}(dnr^2)$ ; see Section 2.2. As long as  $r$  is reasonably small, this compares favorably with the  $\mathcal{O}(n^d)$  operations needed by the standard approach.

In Section 3, we test the performance of the new method for two different option pricing problems, the interpolation of

- American option prices in the Heston model in  $d = 5$  parameters, and of
- prices of basket options in the Black-Scholes model in up to  $d = 25$  underlyings.

At comparable accuracy, the interpolation in American option prices reveals a promising gain in efficiency when compared to an ADI-based PDE solver. The efficiency gain for the basket option prices is shown in comparison to a Monte Carlo simulation with variance reduction.

**2. TT format and tensor completion for Chebyshev interpolation.** This section describes the methodology proposed in this work. We start with recalling the tensorized Chebyshev interpolation method from [14]. After introducing the TT format [39], we present and extend the tensor completion approach from [45]. Finally, we explain how to combine these algorithms in order to efficiently price parametric options for a large number of parameters.

**2.1. Chebyshev interpolation for parametric option pricing.** We consider an option price that depends on a vector of  $d$  parameters  $\mathbf{p}$  contained in  $[-1, 1]^d$ ; general hyperrectangular parameter domains can be addressed by a suitable affine transformation. The basic idea developed in [14] consists of using tensorized Chebyshev interpolation in the parameters (model and payoff parameters) to increase the efficiency

of computing option prices, while maintaining satisfactory accuracy. Writing  $\text{Price}^{\mathbf{P}}$  for the price evaluated in  $\mathbf{p}$ , the Chebyshev interpolation of order  $\bar{\mathbf{n}} := (n_1, \dots, n_d)$  with  $n_i \in \mathbb{N}_0$  is given by

$$(2.1) \quad I_{\bar{\mathbf{n}}}(\text{Price}^{(\cdot)})(\mathbf{p}) = \sum_{j_1=0}^{n_1} \cdots \sum_{j_d=0}^{n_d} c_{j_1, \dots, j_d} T_{j_1, \dots, j_d}(\mathbf{p}).$$

The basis functions  $T_{j_1, \dots, j_d}$  are constructed from Chebyshev polynomials by

$$(2.2) \quad T_{j_1, \dots, j_d}(\mathbf{p}) = \prod_{i=1}^d T_{j_i}(p_i), \quad T_{j_i}(p_i) = \cos(j_i \arccos(p_i)),$$

and the coefficients  $c_{j_1, \dots, j_d}$  are defined as in (1.3). (

$$c_{j_1, \dots, j_d} = \left( \prod_{i=1}^d \frac{2^{\mathbb{1}_{n_i > j_i > 0}}}{n_i} \right) \sum_{k_1=0}^{n_1} \cdots \sum_{k_d=0}^{n_d} \mathcal{P}(k_1, \dots, k_d) \prod_{i=1}^d \cos\left(j_i \pi \frac{k_i}{n_i}\right),$$

where  $\sum''$  indicates that the first and the last summand are halved. ) ← The tensor  $\mathcal{P}$  contains the prices on the tensorized Chebyshev grid and is defined as in (1.2).

$$\mathcal{P}(k_1, \dots, k_d) = \text{Price}^{q_{k_1, \dots, k_d}},$$

where  $q_{k_1, \dots, k_d} := (q_{k_1}, \dots, q_{k_d})$  is defined via Chebyshev nodes  $q_{k_i} := \cos(\pi \frac{k_i}{n_i})$  for  $k_i = 0, \dots, n_i$  and  $i = 1, \dots, d$ . A convergence analysis of the tensorized Chebyshev interpolation in the setting of option pricing is given in [14].

The tensor  $\mathcal{P}$  is of order  $d$  and size  $(n_1 + 1) \times \cdots \times (n_d + 1)$ . The interpolation procedure first requires to compute each entry of this tensor with the reference method. This becomes expensive when the interpolation order and the dimension  $d$  increase. We will use tensor completion to lower this cost.

*Remark 2.1* (Choice of interpolation order). In our numerical experiments, the interpolation order  $\bar{\mathbf{n}}$  is chosen a priori for simplicity. However, this choice can be made adaptively as explained in [22] for the case  $d = 3$  (the extension to general  $d$  is straightforward).

**2.2. TT format.** For recalling the TT format introduced in [39], we consider a general tensor  $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$  of order  $d$ . For each  $\mu = 1, \dots, d - 1$ , the entries of  $\mathcal{X}$  can be rearranged into a matrix

$$X^{<\mu>} \in \mathbb{R}^{(n_1 n_2 \cdots n_\mu) \times (n_{\mu+1} \cdots n_d)},$$

which is called the  $\mu$ th *unfolding* of  $\mathcal{X}$ . For this purpose, the first  $\mu$  indices of  $\mathcal{X}$  are merged into the row index and the last  $n - \mu$  indices into a column index; see [39] for a formal definition. The *TT ranks* of  $\mathcal{X}$  form an integer tuple

$$(2.3) \quad \text{rank}_{\text{TT}}(\mathcal{X}) = (r_0, r_1, \dots, r_d) := (1, \text{rank}(X^{<1>}), \dots, \text{rank}(X^{<d-1>}), 1).$$

Every entry  $\mathcal{X}(i_1, i_2, \dots, i_d)$  can be expressed as a product of  $d$  matrices

$$\mathcal{X}(i_1, i_2, \dots, i_d) = U_1(i_1)U_2(i_2) \cdots U_d(i_d),$$

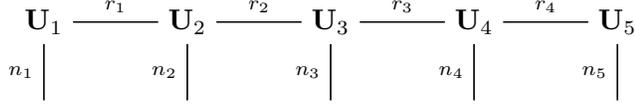


FIG. 2.1. Tensor network diagram of TT decomposition for a tensor of order  $d = 5$ .

with  $U_\mu(i_\mu)$  a matrix of size  $r_{\mu-1} \times r_\mu$ . For each  $\mu = 1, \dots, d$ , one can then collect the  $n_\mu$  matrices  $U_\mu(i_\mu)$ ,  $i_\mu = 1, 2, \dots, n_\mu$  into a third order tensor  $\mathbf{U}_\mu$  of size  $r_{\mu-1} \times n_\mu \times r_\mu$ . These tensors are called *TT cores* and, by construction, we have

$$(2.4) \quad \mathcal{X}(i_1, i_2, \dots, i_d) = \sum_{k_1=1}^{r_1} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \mathbf{U}_1(1, i_1, k_1) \mathbf{U}_2(k_1, i_2, k_2) \cdots \mathbf{U}_d(k_{d-1}, i_d, 1).$$

Figure 2.1 illustrates this so-called *TT decomposition* by a tensor network diagram [38]. Provided that the TT ranks remain moderate, a significant memory reduction is obtained by storing instead of  $\mathcal{X}$  the TT cores: from  $\mathcal{O}(n^d)$  to  $\mathcal{O}(dnr^2)$ , where  $r = \max\{r_0, \dots, r_d\}$  and  $n = \max\{n_1, \dots, n_d\}$ .

Some operations can be effected quite cheaply in the TT format for tensors of low TT ranks. Let us first consider the inner product of two tensors  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times \dots \times n_d}$  defined as

$$(2.5) \quad \langle \mathcal{X}, \mathcal{Y} \rangle = \langle \text{vec}(\mathcal{X}), \text{vec}(\mathcal{Y}) \rangle = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathcal{X}(i_1, \dots, i_d) \mathcal{Y}(i_1, \dots, i_d),$$

where  $\text{vec}(\cdot)$  stacks the entries of a tensor into a long vector. The corresponding tensor network diagram when  $\mathcal{X}$  and  $\mathcal{Y}$  are both in TT decomposition is shown in Figure 2.2. It can be seen that the summations in (2.5) become contractions between the TT cores of  $\mathcal{X}$  and  $\mathcal{Y}$ . By carrying out these contractions of cores from the left to right, the cost of evaluating the inner product reduces from  $\mathcal{O}(n^d)$  to  $\mathcal{O}(dnr^3)$ , where  $r$  denotes the maximum of all involved TT ranks.

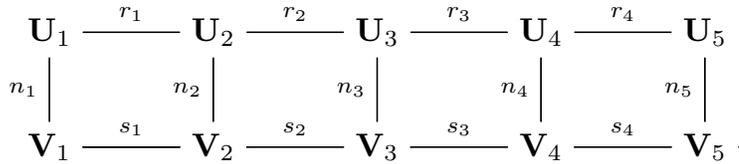


FIG. 2.2. Inner product of two tensors of order  $d = 5$  in TT decomposition.

The *mode- $\mu$  matrix multiplication* between a tensor  $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$  and a matrix  $M \in \mathbb{R}^{m \times n_\mu}$  results in a tensor  $\mathcal{Z} \in \mathbb{R}^{n_1 \times \dots \times n_{\mu-1} \times m \times n_{\mu+1} \times \dots \times n_d}$  defined by

$$\mathcal{Z}(i_1, \dots, i_{\mu-1}, j, i_{\mu+1}, \dots, i_d) = \sum_{i_\mu=1}^{n_\mu} \mathcal{X}(i_1, \dots, i_d) M(j, i_\mu), \quad j = 1, \dots, m.$$

We will denote this operation by  $(\mathcal{Z} = \mathcal{X} \times_k M) \leftarrow \mathcal{Z} = \mathcal{X} \times_\mu M$ . If  $\mathcal{X}$  is in TT decomposition (2.4) then it is straightforward to obtain a TT decomposition for  $\mathcal{Z}$ , by performing a mode-2 matrix multiplication of  $\mathbf{U}_\mu$  with  $M$ . Letting  $c_M$  denote the cost of multiplying  $M$  with a vector, this requires  $(\mathcal{O}(c_M nr)) \leftarrow \mathcal{O}(c_M r^2)$  operations instead of the  $\mathcal{O}(c_M n^{d-1})$  operations needed when  $\mathcal{X}$  is a general tensor.

**2.3. Completion algorithm.** The goal of completion algorithms is to reconstruct a given data set from a small fraction of its entries. As this is clearly an ill-posed task, one needs to additionally impose some regularization, such as smoothness conditions. In this work, we impose low TT ranks on the tensor  $\mathcal{P}$  containing the prices and reconstruct  $\mathcal{P}$  using the completion algorithm proposed in [45].

In the following, we briefly summarize the approach from [45]. Let  $\mathcal{A} \in \mathbb{R}^{n_1 \times \dots \times n_d}$  denote the original data tensor for which only the entries in a (small) *training set*  $(\Omega \subset \{1, n_1\} \times \dots \times \{1, n_d\}) \leftarrow \Omega \subset \{1, \dots, n_1\} \times \dots \times \{1, \dots, n_d\}$  are known. When aiming at fitting a tensor of fixed (low) TT ranks  $\mathbf{r} = (r_0, \dots, r_d)$  to this data, completion takes the form of the constrained optimization problem

$$(2.6) \quad \begin{aligned} \min_{\mathcal{X}} \quad & \|P_\Omega \mathcal{X} - P_\Omega \mathcal{A}\|^2 \\ \text{subject to} \quad & \mathcal{X} \in \mathcal{M}_{\mathbf{r}} := \{\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d} \mid \text{rank}_{\text{TT}} = \mathbf{r}\}, \end{aligned}$$

where  $P_\Omega \mathcal{X}$  denotes the orthogonal projection onto  $\Omega$  and  $\|\cdot\|$  is the norm induced by the inner product (2.5). It is known that  $\mathcal{M}_{\mathbf{r}}$  is a smooth embedded submanifold, which enables one to apply Riemannian optimization techniques to (2.6). Specifically, in [45] it is proposed to employ a Riemannian conjugate gradient (CG) method (see Algorithm 1 in [45]). This method produces iterates that stay on the manifold and, in turn, can be stored and manipulated efficiently in the TT format. One iteration requires  $\mathcal{O}(dnr^3 + d|\Omega|r^2)$  operations, where  $|\Omega|$  denotes the cardinality of  $\Omega$ .

Our stopping criterion of Riemannian CG is designed to attain a level of accuracy warranted by the data and the chosen TT ranks. Following [45], we choose a *test set*  $\Omega_C$  of, say, 100 additional parameter samples not in the training set  $\Omega$ . Letting  $\mathcal{X}_k$  denote the  $k$ th iterate of Riemannian CG algorithm, we measure the errors on the training and the test set:

$$\epsilon_\Omega(\mathcal{X}_k) := \frac{\|P_\Omega \mathcal{A} - P_\Omega \mathcal{X}_k\|}{\|P_\Omega \mathcal{A}\|}, \quad \epsilon_{\Omega_C}(\mathcal{X}_k) := \frac{\|P_{\Omega_C} \mathcal{A} - P_{\Omega_C} \mathcal{X}_k\|}{\|P_{\Omega_C} \mathcal{A}\|}.$$

The algorithm is stopped once these errors stagnate, that is,

$$(2.7) \quad \frac{|\epsilon_\Omega(\mathcal{X}_k) - \epsilon_\Omega(\mathcal{X}_{k+1})|}{|\epsilon_\Omega(\mathcal{X}_k)|} < \delta \quad \text{and} \quad \frac{|\epsilon_{\Omega_C}(\mathcal{X}_k) - \epsilon_{\Omega_C}(\mathcal{X}_{k+1})|}{|\epsilon_{\Omega_C}(\mathcal{X}_k)|} < \delta,$$

holds for some small  $\delta > 0$ .

**2.3.1. Adaptive rank and adaptive sampling strategy.** To set up the optimization problem (2.6), two issues remain to be discussed: The choice of the TT ranks  $\mathbf{r}$  and a suitable training set  $\Omega$ . For our application, these are not known a priori and thus need to be chosen adaptively.

Concerning the choice of TT ranks, we follow the adaptive strategy proposed in [45]. We start by solving (2.6) for the smallest sensible choice of TT ranks,  $\mathbf{r} = (1, \dots, 1)$ . Most likely, this choice will not suffice to obtain satisfactory accuracy and the error on the test set will be relatively large. To decrease it, the obtained solution is used as starting value for Riemannian CG applied again to (2.6), but this time

with the increased TT ranks  $\mathbf{r} = (1, 2, 1, \dots, 1)$  as discussed in [45]. See also [47] for a greedy rank update procedure in the context of matrix completion. The described procedure is repeated by increasing cyclically every TT rank  $r_\mu$ . The overall algorithm stops as soon as increasing any of the TT ranks does not improve the test set error anymore or the maximal possible rank  $r_{\max}$  is reached; see Algorithm 2.1.

---

**Algorithm 2.1** Adaptive rank strategy

---

**Input:** Data on sampling/test sets  $\Omega, \Omega_C$ , max rank  $r_{\max}$ , acceptance parameter  $\rho \geq 0$

**Output:** Completed tensor  $\mathcal{X}$  with adaptively chosen TT ranks  $\mathbf{r}$ ,  $r_\mu \leq r_{\max}$ .

```

1:  $\mathcal{X}$  random tensor having TT ranks  $\mathbf{r} = (1, \dots, 1)$ 
2:  $\mathcal{X} \leftarrow$  result of Riemannian CG (see Section 2.3) using starting guess  $\mathcal{X}$ 
3: locked = 0
4:  $\mu = 1$ 
5: while locked <  $d - 1$  &  $\max_\nu r_\nu < r_{\max}$  do
6:    $\mathcal{X}_{\text{new}} \leftarrow$  increase  $\mu$ th rank of  $\mathcal{X}$  to  $(r_0, \dots, r_{\mu-1}, r_\mu + 1, r_{\mu+1}, \dots, r_d)$ 
7:    $\mathcal{X}_{\text{new}} \leftarrow$  result of Riemannian CG using starting guess  $\mathcal{X}_{\text{new}}$ 
8:   if  $(\epsilon_{\Omega_C}(\mathcal{X}_{\text{new}}) - \epsilon_{\Omega_C}(\mathcal{X})) > -\rho$  then
9:     locked  $\leftarrow$  locked + 1   % revert step
10:  else
11:    locked  $\leftarrow$  0,  $\mathcal{X} \leftarrow \mathcal{X}_{\text{new}}$    % accept step
12:  end if
13:   $\mu \leftarrow 1 + (\mu \bmod d - 1)$ 
14: end while

```

---

For the adaptive choice of the sampling set  $\Omega$ , which has not been addressed in [45], we present two different strategies. The core idea is to gradually increase the size of  $\Omega$  in order to improve the approximation of the tensor. Both strategies are also combined with Algorithm 2.1 and they differ only in the measurement of the error.

The steps of the first adaptive sampling strategy are as follows.

1. Start with a sample set  $\Omega$  of small size and a test set  $\Omega_C$  of a certain prescribed size  $|\Omega_C|$ . Run Algorithm 2.1.
2. Measure the relative error on the test set  $\Omega_C$  and stop if the stopping criterion is satisfied. If not satisfied, add the test set  $\Omega_C$  to the sample set  $\Omega$  and create a new test set of size  $|\Omega_C|$ . In our applications, this corresponds to computing new option prices on the Chebyshev grid using the reference method.
3. Run again Algorithm 2.1 from line 2 to the end, by using a rank  $\mathbf{r} = (1, \dots, 1)$  approximation of the result from the previous step as initial guess for the CG algorithm.
4. Repeat 1-3 until a maximal sampling percentage is reached or an a priori chosen stopping criterion is satisfied.

The pseudo-code in Algorithm 2.2 summarizes this first strategy.

The second adaptive sampling strategy that we propose is designed in a similar way. The only difference is that the error is measured on an a priori defined fixed set  $\Gamma$  and not on  $\Omega_C$ , which changes at each step. Therefore, this strategy follows the same steps as the first one, with the only difference that in Step 2 we measure the error on the set  $\Gamma$ , which has been previously defined. The algorithm summarizing this second strategy can be obtained by replacing line 3 and line 12 in Algorithm 2.2

---

**Algorithm 2.2** Adaptive sampling strategy 1

---

**Input:** Initial sampled data  $P_\Omega \mathcal{A}$ , maximal rank  $r_{\max}$  for rank adaptivity, maximal allowed size percentage  $p$  (of  $\Omega$ )

**Output:** Completed tensor  $\mathcal{X}$  of TT ranks  $\mathbf{r}$ ,  $r_\mu \leq r_{\max}$

- 1: Create test set  $\Omega_C^{\text{new}}$  such that  $\Omega \cap \Omega_C^{\text{new}} = \emptyset$
  - 2: Run Algorithm 2.1 with  $\Omega, \Omega_C^{\text{new}}$  and get completed tensor  $\mathcal{X}_c$ .
  - 3:  $\text{err}_{\text{new}} \leftarrow \epsilon_{\Omega_C^{\text{new}}}(\mathcal{X}_c)$
  - 4:
  - 5: **while**  $|\Omega|/\text{size}(\mathcal{A}) < p$  **do**
  - 6:    $\text{err}_{\text{old}} \leftarrow \text{err}_{\text{new}}$
  - 7:    $\tilde{\mathcal{X}} \leftarrow \text{rank}(1, \dots, 1)$  approximation of  $\mathcal{X}_c$
  - 8:    $\Omega_C^{\text{old}} \leftarrow \Omega_C^{\text{new}}$
  - 9:   Create new test set  $\Omega_C^{\text{new}}$  such that  $\Omega_C^{\text{new}} \cap \Omega_C^{\text{old}} = \emptyset$
  - 10:    $\Omega \leftarrow \Omega \cup \Omega_C^{\text{old}}$
  - 11:   Run Algorithm 2.1 (from line 2 to end) with  $\Omega, \Omega_C^{\text{new}}$  and  $\tilde{\mathcal{X}}$  as starting guess. Get completed tensor  $\mathcal{X}_c$  out of it.
  - 12:    $\text{err}_{\text{new}} \leftarrow \epsilon_{\Omega_C^{\text{new}}}(\mathcal{X}_c)$
  - 13:   **if** stopping criterion satisfied **then**
  - 14:     Break
  - 15:   **end if**
  - 16: **end while**
  - 17:
  - 18:  $\mathcal{X} \leftarrow \mathcal{X}_c$
- 

with

$$\text{err}_{\text{new}} \leftarrow \epsilon_\Gamma(\mathcal{X}_c).$$

The stopping criterion of line 13 can be also defined in different ways. We choose to stop the algorithm if one of the following criteria is satisfied:

1. if  $\text{err}_{\text{new}} < \text{tol}$ , where  $\text{tol}$  is a prescribed tolerance;
2. if  $|\text{err}_{\text{new}} - \text{err}_{\text{old}}| < \text{tol}'$  where  $\text{tol}'$  is a prescribed tolerance;
3. if  $\exists \mu$  such that  $r_\mu(\mathcal{X}_c) == r_{\max}$ .

The first criterion allows us to stop as soon as the error goes below a certain level, the second stops the algorithm whenever the error stagnates and the last one when the TT rank has reached the maximal allowed rank at least in one mode  $\mu$ .

We test the new adaptive sampling strategies on a problem with known solution in the next section.

**2.3.2. Numerical test for adaptive sampling strategies.** We consider the problem of Chapter 5.4.2 in [45] and we apply our adaptive sampling strategies to it in order to compare them and to investigate their advantages and disadvantages. We expect a similar performance of both strategies in terms of accuracy and compression. In this numerical example, as well as in the rest of the paper, we choose  $\|\cdot\|$  to be the 2-norm and  $\delta = 10^{-4}$  in (2.7). The problem consists of discretizing the function

$$f : [0, 1]^4 \rightarrow \mathbb{R}, \quad f(\mathbf{x}) = \exp(-\|\mathbf{x}\|)$$

using  $n = 20$  equally spaced discretization points on  $[0, 1]$  in each mode. We aim at reconstructing the tensor  $\mathcal{X} \in \mathbb{R}^{20 \times 20 \times 20 \times 20}$  containing the function values in the grid

and defined as

$$\mathcal{X}(i_1, i_2, i_3, i_4) := f(\mathbf{x}_{i_1, i_2, i_3, i_4}),$$

where

$$\mathbf{x}_{i_1, i_2, i_3, i_4} := \left( \frac{(i_1 - 1)}{19}, \frac{(i_2 - 1)}{19}, \frac{(i_3 - 1)}{19}, \frac{(i_4 - 1)}{19} \right),$$

for  $i_1, i_2, i_3, i_4 = 1, \dots, 20$ . In Algorithm 2.2 we set the maximum rank to  $\mathbf{r}_{\max} = (1, 7, 7, 7, 1)$  and we start with an initial sampling set  $\Omega$  satisfying  $|\Omega|/n^4 = 0.01$ . Moreover, we set the acceptance parameter  $\rho$  of Algorithm 2.1 to  $\rho = 10^{-4}$ . In order to analyze the behavior of the error, we do not impose any stopping criterion, but we let our adaptive sampling strategies run until  $|\Omega|/\text{size}(\mathcal{A}) > 0.25$ . The size of each  $\Omega_C$  is set to 2000 and  $|\Gamma| = 3000$  for the second strategy. Figures 2.3 and 2.4 show the results for the two different strategies.

First, we observe that both strategies eventually reach the same accuracy and the same final TT ranks, which makes both of them valid. We observe an oscillatory behavior in Figure 2.3. This non-smooth decay can be expected since in each step the error is measured on a different test set  $\Omega_C$ . We observe that the amplitude of the oscillations becomes smaller as  $|\Omega|$  increases. This indicates an error stagnation over the whole tensor which cannot be improved by enlarging  $(\Omega_C) \leftarrow \Omega$  further. On the other hand, the error in the second strategy behaves almost monotonically and stagnates much earlier than in the previous case. This is due to the fact that we measure it on the fixed set  $\Gamma$ . In practice, the earlier error stagnation of the second strategy is preferable as it triggers the stopping criterion 2. However, the second strategy has the disadvantage of the initial additional cost of evaluating the tensor in the set  $\Gamma$ . In our numerical experiments in Section 3 we choose the first strategy, which turned out to be more favorable since the stopping criterion 1 was triggered.

Lastly, we compared the completed tensors resulting from running both adaptive sampling strategies with the best TT decomposition of  $\mathcal{X}$ . The latter was obtained by first explicitly constructing the tensor  $\mathcal{X}$  and then by applying the TT-SVD to it (we applied [39, Algorithm 1] with prescribed accuracy  $\epsilon = 10^{-14}$ ). The TT ranks of the best TT decomposition are  $(1, 17, 19, 17, 1)$ , i.e.  $\mathcal{X}$  has a low-rank structure. For the first adaptive sampling strategy, the relative error was  $1.69 \cdot 10^{-4}$ , while for the second one we obtained a relative error of  $1.75 \cdot 10^{-4}$ . Both results are consistent with the relative errors shown in Figures 2.3 and 2.4. In particular, this confirms that the error measurements on the test sets are representative for the error on the full tensor, supporting the effectiveness of the adaptive sampling strategies.

**2.4. Combined methodology.** We are now in the position to combine the concepts and the algorithms in order to develop an efficient procedure for high-dimensional tensorized Chebyshev interpolation.

We would like to price options that depend on a vector  $\mathbf{p} = (p_1, \dots, p_d)$  of  $d$  varying parameters. It is reasonable to assume that every combination of parameters  $\mathbf{p}$  belongs to a compact hyper-rectangular  $[p_1, \bar{p}_1] \times [p_2, \bar{p}_2] \times \dots \times [p_d, \bar{p}_d]$ . For example, if time-to-maturity  $T$  belongs to the set of varying parameters, we can assume that  $T \in [0.05, 2]$ ; similarly for the other payoff or model parameters. The combined methodology consists of two phases: offline phase and online phase, as already introduced in [14].

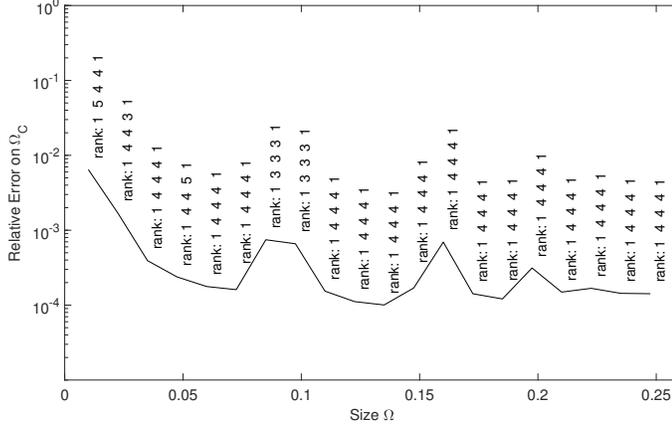


FIG. 2.3. Relative error on varying test sets  $\Omega_C$  for different sampling set sizes in adaptive sampling strategy 1.

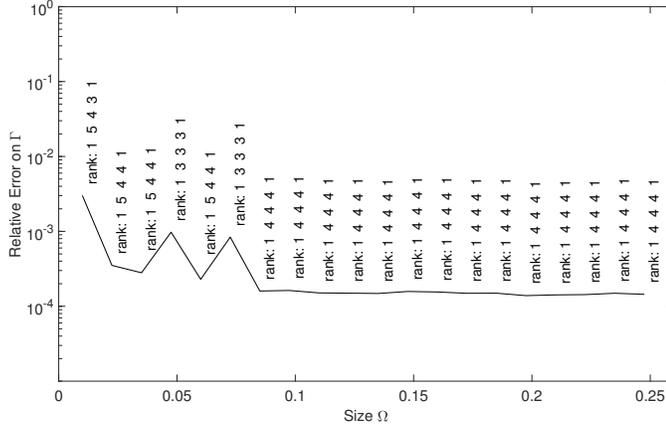


FIG. 2.4. Relative error on set  $\Gamma$  for different sampling set sizes in adaptive sampling strategy 2.

**2.4.1. Offline phase - Computation of  $\mathcal{P}$ .** The offline phase starts by performing following operations:

1. Fix an interpolation order  $\bar{\mathbf{n}} = (n_1, \dots, n_d)$  and compute the entries of the tensor  $\mathcal{P}$  (as defined in (1.2)) from an a priori chosen subset  $\Omega$  of Chebyshev nodes, using the reference pricing technique.
2. Apply tensor completion with adaptive sampling strategy (Algorithm 2.2) in order to get a low-rank approximation of the tensor  $\mathcal{P}$  in TT format.

For simplicity, we denote the obtained low-rank approximation of  $\mathcal{P}$  again by  $\mathcal{P}^1$ . In the last step of the offline phase we construct the interpolation coefficients, defined in (1.3). We denote the tensor of coefficients by  $\mathcal{C} \in \mathbb{R}^{(n_1+1) \times (n_2+1) \times \dots \times (n_d+1)}$ . Its entries are therefore given by (adjusting the ordering according the Sections 2.1 and

<sup>1</sup>Note that in Sections 1 and 2.1 we let the indices of the entries of  $\mathcal{P}$  and  $\mathcal{C}$  start from 0. From now on, we let them start from 1 in order to be consistent with the new notation introduced in Section 2.2

2.2)

$$(2.8) \quad \mathcal{C}(i_1, i_2, \dots, i_d) = c_{i_1-1, i_2-1, \dots, i_d-1},$$

for  $i_j = 1, \dots, n_j + 1$  and  $j = 1, \dots, d$ . The tensor  $\mathcal{C}$  can be efficiently computed in TT format, as explained in the following subsection.

**2.4.2. Offline phase - Efficient computation of  $\mathcal{C}$ .** In order to explain the algorithm we first consider the simple case  $d = 1$ . In this case  $\mathcal{P}$  and  $\mathcal{C}$  are in  $\mathbb{R}^{(n_1+1) \times 1}$ , where  $n_1$  is the chosen interpolation order. The entries of  $\mathcal{C}$  are given by

$$\mathcal{C}(j+1) = \frac{2^{\mathbb{1}_{n_1 > j > 0}}}{n_1} \sum_{k=0}^{n_1} \mathcal{P}(k+1) \cos\left(j\pi \frac{k}{n_1}\right), \quad j = 0, \dots, n_1,$$

so that the whole vector  $\mathcal{C}$  can be computed via the matrix-vector multiplication

$$(2.9) \quad \mathcal{C} = \frac{2}{n_1} \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \dots & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \cos\left(\frac{\pi}{n_1}\right) & \dots & \cos\left(\frac{\pi(n_1-1)}{n_1}\right) & \frac{1}{2} \cos(\pi) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2} & \cos\left(\frac{\pi(n_1-1)}{n_1}\right) & \dots & \cos\left(\frac{\pi(n_1-1)^2}{n_1}\right) & \frac{1}{2} \cos(\pi(n_1-1)) \\ \frac{1}{4} & \frac{1}{2} \cos(\pi) & \dots & \frac{1}{2} \cos(\pi(n_1-1)) & \frac{1}{4} \cos(\pi n_1) \end{pmatrix} \mathcal{P},$$

and we denote by  $F_{n_1} \in \mathbb{R}^{(n_1+1) \times (n_1+1)}$  the matrix multiplying  $\mathcal{P}$  in (2.9).

For a general dimension  $d > 1$ , the same reasoning can be applied and the tensor  $\mathcal{C}$  of interpolation coefficients can be computed by sub-sequentially multiplying  $\mathcal{P}$  with  $F_{n_i}$  ( $i = 1, \dots, d$ ) via the mode- $\mu$  multiplication, defined in Section 2.2. The final procedure for an efficient computation of  $\mathcal{C}$  is given in Algorithm 2.3.

---

**Algorithm 2.3** Efficient computation of  $\mathcal{C}$

---

**Input:** Tensor  $\mathcal{P}$  in TT format containing option prices in the Chebyshev grid

**Output:** Tensor  $\mathcal{C}$  as defined in (1.3), in TT format

- 1: Compute  $F_{n_1}$  as in (2.9).
  - 2:  $\mathcal{C} \leftarrow \mathcal{P} \times_1 F_{n_1}$
  - 3: **for**  $m = 2, \dots, d$  **do**
  - 4:   Compute  $F_{n_m}$
  - 5:    $\mathcal{C} \leftarrow \mathcal{C} \times_m F_{n_m}$ .
  - 6: **end for**
- 

Note that if  $n_1 = \dots = n_d =: n$  (as for example in our numerical experiments in Section 3), Algorithm 2.3 can be further simplified by computing the matrix  $F_n$  only once. The particular structure of the matrices  $F_{n_i}$  allows us to apply a Fast-Fourier-Transform based algorithm which computes each mode multiplication in  $\mathcal{O}(r^2 n \log(n))$  (instead of  $\mathcal{O}(r^2 n^2)$  as mentioned in Section 3). Therefore, the total complexity for computing  $\mathcal{C}$  is  $\mathcal{O}(dnr^2 \log(n))$ .

The offline phase can be finally completed by performing the step

3. Construct the tensor  $\mathcal{C}$  as explained in Algorithm 2.3.

**2.4.3. Online phase.** Once we have stored  $\mathcal{C}$  in TT format, we can use it to compute every option price via interpolation during the online phase. For any particular choice of parameters  $\mathbf{p}$ , we first perform the step

4. Evaluate the Chebyshev tensor basis (2.2) in  $\mathbf{p}$ .

This step returns a tensor  $\mathcal{T}_{\mathbf{p}} \in \mathbb{R}^{(n_1+1) \times (n_2+1) \times \dots \times (n_d+1)}$  of TT rank  $(1, \dots, 1)$ , that we store in TT format. The interpolated price, defined in (2.1), can now be rewritten as the inner product

$$(2.10) \quad I_{\bar{\mathbf{n}}}(\text{Price}^{(\cdot)})(\mathbf{p}) = \langle \mathcal{C}, \mathcal{T}_{\mathbf{p}} \rangle.$$

The final step of our combined methodology is then defined as

5. Compute the interpolated price (2.10) in TT format as in (2.5).

If we consider a fixed interpolation order  $n$  in each dimension and if the TT ranks of  $\mathcal{P}$  and  $\mathcal{C}$  are approximately  $r$ , then the total cost for performing both Step 3 and Step 5 is given by  $\mathcal{O}(dnr^2 + dnr^2 \log(n))$ . These two steps are represented via a tensor network diagram in Figure 2.5 (for  $d = 5$ ), where we denoted by  $\mathbf{P}_i$  the core tensors of  $\mathcal{P}$  and by  $\mathbf{T}_i$  the ones of  $\mathcal{T}_{\mathbf{p}}$ .

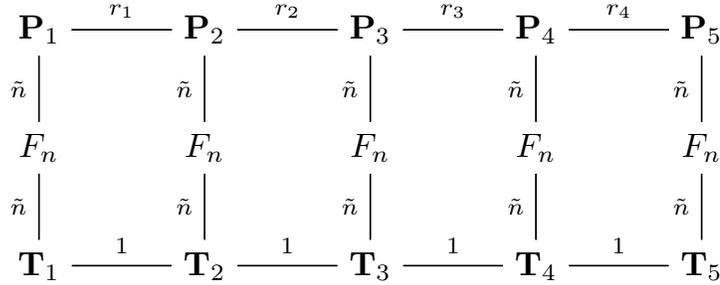


FIG. 2.5. Tensor network diagram representing the whole interpolation procedure as in (2.1) and (1.3), for  $d = 5$  in TT format. Note that  $\tilde{n} := n + 1$ .

Finally, we summarize our complete methodology in Algorithm 2.4.

---

**Algorithm 2.4** Combined methodology for Chebyshev interpolation in parametric option pricing

---

**Input:** Interpolation order  $\bar{\mathbf{n}}$ , subset  $\Omega$  of total Chebyshev points, set  $\Pi$  of parameters  $\mathbf{p}$  for which we want to compute option prices

**Output:** Interpolated option prices for parameters  $\mathbf{p} \in \Pi$

- 1: % Offline phase
  - 2: Compute option prices using reference method in the subset  $\Omega$  of Chebyshev points
  - 3: Construct  $\mathcal{P}$  using tensor completion in TT format (Algorithm 2.2)
  - 4: Construct tensor  $\mathcal{C}$  as in Section 2.4.2
  - 5:
  - 6: % Computation of option prices - Online phase
  - 7: **for**  $\mathbf{p} \in \Pi$  **do**
  - 8:   Evaluate the Chebyshev tensor basis  $\mathcal{T}_{\mathbf{p}}$
  - 9:   Compute interpolated price (2.10)
  - 10: **end for**
- 

In the next section we see how this combined methodology performs on concrete examples.

**3. Financial applications and numerical experiments.** Putting the new approach to test, we implement the method described in Section 2 for two different types of applications. In the first one, we tackle computational intense option pricing methods in a parametric model. We treat option prices as functions in the parameter space which consists of model and option parameters. We then approximate the price function by Chebyshev interpolation in the parameter space. This approach has been successfully tested in cases where the parameter space is low-dimensional. In various applications, several varying parameters are of interest. If the interpolation is even efficient in the full parameter space, it is indeed a new pricing methodology. Here, we combine Chebyshev interpolation and low-rank approximation to cope with higher dimensionality in the parameter space. Already for pricing single asset options, it is promising to tackle medium and high-dimensional parameters spaces in this approach. As a generic example, we choose to approximate American put option prices in the Heston model with the varying parameters  $K, \rho, \sigma, \kappa$  and  $\theta$ . It turns out that the computational complexity reduces significantly in this case.

As second type of application we examine the interpolation of basket option prices in the  $d$ -variate Black-Scholes model as function of the initial stock prices. This is a prototypical example for the computation of generalized conditional moments of high-dimensional Markov processes.

All algorithms have been implemented in MATLAB and run on a standard laptop (Intel Core i7, 2 cores, 256kB/4MB L2/L3 cache). In order to deal with tensors, we used the toolboxes [39] by Oseledets and [2, 3], while for the completion algorithm we used the TT completion toolbox described in [32, 33, 45, 46]. Note that in this toolbox the most expensive steps have been implemented in C using the Mex-function capabilities of MATLAB.

**3.1. Pricing American options in Heston's model.** We consider pricing single asset American put options in the Heston model. As introduced by Heston in [24], the price dynamics of the financial asset under the risk neutral measure are given by

$$dS_t = rS_t dt + \sqrt{v_t} S_t dW_t^1,$$

where the square of the volatility  $v_t$  is modeled by the square root process

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t}dW_t^2.$$

Here, the two Brownian motions  $W^1$  and  $W^2$  are correlated with correlation parameter  $\rho$ , mean-reversion rate  $\kappa > 0$ , long-term mean  $\theta > 0$ , volatility of the variance  $\sigma > 0$  and, finally, fixed and deterministic continuously compounding interest rate  $r$ .

The price of an American option at time  $t < T$ , maturing at  $T$ , with initial underlying price  $s \geq 0$  and initial volatility  $v \geq 0$  is given by (

$$(3.1) \quad \text{Price} = \sup_{t < \tau < T} \mathbb{E}[e^{-r\tau} f(S_\tau) | S_t = s, v_t = v],$$

)←

$$(3.2) \quad \text{Price} = \sup_{t < \tau < T} \mathbb{E}[e^{-r(\tau-t)} f(S_\tau) | S_t = s, v_t = v],$$

where the sup is taken over all stopping times  $\tau$  in  $[t, T]$ . Here,  $f$  denotes the payoff function of the European put option, i.e.

$$f(x) = (K - x)^+,$$

where  $K$  denotes the strike price.

It is well-known (see e.g. [13]) that the price (3.2) of the American option satisfies the following partial differential complementarity problem (PDCP):

$$(3.3) \quad \begin{cases} \partial_t \text{Price} \geq \mathcal{G}\text{Price} \\ \text{Price} \geq f \\ (\text{Price} - f)(\partial_t \text{Price} - \mathcal{G}\text{Price}) = 0, \end{cases}$$

where  $\mathcal{G}$  is the infinitesimal generator of  $(s, v)$  in the Heston model, defined as

$$\mathcal{G}g(s, v) = \frac{1}{2}s^2v\partial_{ss}^2g + \rho\sigma sv\partial_{sv}^2g + \frac{1}{2}\sigma^2v\partial_{vv}^2g + rs\partial_sg + \kappa(\theta - v)\partial_vg - rg.$$

The problem (3.3) has been well studied in the literature and different pricing algorithms have been developed so far. In our example we consider, as reference method for our combined methodology, the pricing algorithm explained in [21]. More precisely, the authors propose different schemes for the time discretization and we consider the Hundsdorfer Verwer - Ikonen Toivanen (HV-IT) scheme, explained at page 219 of [21].

Solving the discretized PDCP yields an approximate price for all values of  $S_0, v_0$  and  $T$  in each grid point of the pre-specified domain. For many applications we would like to have the solution at hand for other parameters (as well). In calibration, for instance, we observe  $S_0$  and  $r$ , and one could estimate  $v_0$  from historical stock price data. Then the calibration problem reduces to fitting the parameters  $(K, \rho, \sigma, \kappa, \theta)$  to the observed option price data. To do so one needs to solve an optimization problem where prices need to be computed for large sets of parameters  $(K, \rho, \sigma, \kappa, \theta, T)$ . Since the price for different maturities can be obtained by rescaling  $\kappa$  and  $\sigma$ , effectively we need the prices for combinations of the parameters  $K, \rho, \sigma, \kappa$  and  $\theta$ . This motivates the following set up, where we fix the model and payoff parameters (

$$S_0 = 2, \quad v_0 = 0.0175, \quad r = 0.1, \quad T = 0.25,$$

)←

$$S_0 = 1, \quad v_0 = 0.1, \quad r = 0.01, \quad T = 0.25,$$

and we let vary the five parameters (

$$(K, \rho, \sigma, \kappa, \theta) \in [2; 4] \times [-1; 1] \times [0.2; 0.5] \times [1; 2] \times [0.05; 0.2]$$

)←

$$(K, \rho, \sigma, \kappa, \theta) \in [0.9; 2] \times [-1; 1] \times [0.2; 0.5] \times [1; 2] \times [0.05; 0.2]$$

in their corresponding domain.

In order to compute the reference prices we consider 50 equidistant spatial grid points in both directions  $s$  and  $v$  with  $s_{\min} = 0, s_{\max} = 5, v_{\min} = 0, v_{\max} = 1$ , 40 time steps and the Crank-Nicholson time stepping scheme.

We start by performing the offline phase of Algorithm 2.4. We consider an interpolation order  $n_1 = \dots = n_5 =: n = 10$  in each direction and we construct the tensor  $\mathcal{P}$  by tensor completion as explained in Section 2.3. We apply the first adaptive sampling strategy as in Algorithm 2.2. We choose the completion parameters as

$$\rho = 0, \quad \text{tol} = 10^{-3}, \quad \text{tol}' = 10^{-8}, \quad r_{\max} = 10, \quad |\Omega| = 805, \quad |\Omega_C| = 805, \quad p = 0.2.$$

For this particular example, we were also able to explicitly construct the full tensor (in more than 1 hour and 40 minutes!). In Table 3.2 we show the size of the final set  $\Omega$  (first column), the relative error of the completed tensor on the last  $\Omega_c^{new}$  (second column), the relative error between the obtained completed tensor and the full one (third column), the runtime of the completion, Algorithm 2.2, in seconds (fourth column), the TT-rank of  $\mathcal{P}$  (fifth column), the storage needed to save  $\mathcal{P}$  in TT format, denoted by  $\text{store}(\text{TT})$  and measured in bytes (sixth column) and finally, the storage needed to save the full tensor, denoted by  $\text{store}(\text{full})$  and again measured in bytes. MATLAB requires 8 bytes to store a floating-point number of type double, which gives us the formula  $\text{store}(\text{full}) = 8 \cdot (n+1)^d$  for the storage of the full tensor and  $\text{store}(\text{TT}) = 8 \cdot (n+1)(r_1 r_2 + \dots + r_{d-2} r_{d-1}) + 8 \cdot (n+1)(r_1 + r_{d-1})$  for the storage of the tensor in TT format, see [39].

Table 3.2 shows that a sample set of (5%) $\leftarrow$ 1.5% is sufficient for the algorithm to reach the prescribed accuracy. Furthermore, the relative error of the completed tensor  $\mathcal{P}$  in the 2-norm over the last test sample parameter space  $\Omega_c^{new}$  and the relative error over the full tensor, i.e. over all Chebyshev nodes, is only in the 6th digit. This is one order of magnitude smaller than the relative error on the full  $\mathcal{P}$ . This is a good indication that the approach can be extended to more complex cases, where the computation of the full tensor  $\mathcal{P}$  is not feasible any more (see Section 3.2). The completion time was about (6) $\leftarrow$ 1.5 minutes. Finally, the rank properties together with its storage reduction of a factor of 115 confirm the low-rank structure of the problem. For constructing the tensor  $\mathcal{C}$  (last step of the offline phase) we

TABLE 3.1

Completion results on  $\mathcal{P}$  for the parametric American put option pricing problem in the Heston model.

final $ \Omega $	rel err on last $\Omega_c^{new}$	rel err on full $\mathcal{P}$	completion time (s)
8050 (5 %)	$2.56 \cdot 10^{-5}$	$2.75 \cdot 10^{-5}$	366.12
rank <sub>TT</sub> ( $\mathcal{P}$ )    store(TT) (bytes)    store(full) (bytes)			
(1, 5, 8, 6, 5, 1)                    11264                    1288408			

TABLE 3.2

Completion results on  $\mathcal{P}$  for the parametric American put option pricing problem in the Heston model.

final $ \Omega $	rel err on last $\Omega_c^{new}$	rel err on full $\mathcal{P}$	completion time (s)
2415 (1.5 %)	$1.97 \cdot 10^{-5}$	$2.09 \cdot 10^{-5}$	96.79
rank <sub>TT</sub> ( $\mathcal{P}$ )    store(TT) (bytes)    store(full) (bytes)			
(1, 8, 6, 6, 5, 1)                    11176                    1288408			

applied Algorithm 2.3 and the computation time was (0.0037) $\leftarrow$ 0.0086 seconds, which is negligible compared to the completion time. Hence, almost all the computation time in the offline phase is spent in the construction of the tensor  $\mathcal{P}$ .

Next, we compute American put option prices for the online phase in both ways using our methodology and the reference algorithm. We compute 243 prices with random model parameters uniformly drawn from the reference set  $([2; 4] \times [-1; 1] \times [0.2; 0.5] \times [1; 2] \times [0.05; 0.2]) \leftarrow [0.9; 2] \times [-1; 1] \times [0.2; 0.5] \times [1; 2] \times [0.05; 0.2]$ . We

measure the maximal absolute error over the computed options prices, i.e. we report the quantity

$$\max(|P_{\text{Int}} - P_{\text{Ref}}|),$$

where  $P_{\text{Int}}$  is a vector containing all interpolated prices for the different choices of model parameters; analogously is  $P_{\text{Ref}}$  for the reference method. In Table 3.4 we also report the computation time for computing one single option price for both methods. One can notice that the online phase of the interpolation compared to the reference method accelerates the procedure by a factor of (75)←94. The accuracy of the reference method is reported in part C of Figure 1 in [21] for one specific parameter set to be of the order  $10^{-3}$  in the maximum norm. The interpolation error is one order smaller, making the new procedure at least as accurate as the reference method. Therefore, we can conclude that the methodology strongly outperforms the reference method in the online phase while keeping the same accuracy.

We would like to emphasize that this approach can be further extended to an interpolation in the full set of parameters  $(S_0, v_0, r, T, K, \rho, \sigma, \kappa, \theta)$ . Since then the offline phase needs to be performed only once, this would result in a new pricing method. Here, in the offline phase one could explore the fact that the PDCP solver returns the price for all  $(S_0, v_0, T)$  in the grid to make the sampling steps more efficient. This opens up an interesting topic for future research.

TABLE 3.3

*Results on American put option pricing via combined methodology and reference method.*

time reference method (s)	time interpolation (s)	max abs error
$3.65 \cdot 10^{-2}$	$4.89 \cdot 10^{-4}$	$1.95 \cdot 10^{-4}$

TABLE 3.4

*Results on American put option pricing via combined methodology and reference method.*

time reference method (s)	time interpolation (s)	max abs error
$2.72 \cdot 10^{-2}$	$2.88 \cdot 10^{-4}$	$3.03 \cdot 10^{-4}$

**3.2. Basket options in multivariate Black-Scholes model.** In the  $d$ -variate Black-Scholes model with  $d$  assets  $S^1, \dots, S^d$ , the risk neutral dynamics are given by

$$(3.4) \quad dS_t^i = rS_t^i dt + \sigma_i dW_t^i,$$

where  $r$  is a fixed deterministic interest rate,  $(\sigma_1, \dots, \sigma_d)$  is the vector of volatilities and  $(W^1, \dots, W^d)$  is a vector of correlated Brownian motions with correlation matrix  $\Sigma$ . The solution to (3.4) is given by

$$S_t^i = S_0^i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)t + \sigma_i W_t^i\right).$$

In this section we apply the new methodology in order to price basket options with payoff function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  defined as

$$f(\mathbf{x}) := \left(\sum_{n=1}^d w_n x_n - K\right)^+,$$

where  $K$  is the strike and  $(w_1, \dots, w_d)$  is a vector of weights satisfying  $\sum_{n=1}^d w_n = 1$ . The risk neutral price at time  $t = 0$  of the basket option with maturity  $T$  is, as usual, given by

$$(3.5) \quad \text{Price} = e^{-rT} \mathbb{E}[f(\mathbf{S}_T)].$$

From now on, we consider the parameters  $r, \sigma_i$  ( $i = 1, \dots, d$ ) and the correlation matrix  $\Sigma$  to be fixed, and we let the vector  $\mathbf{S}_0 \in \mathbb{R}^d$  of initial asset prices be the varying parameter. The reference pricing algorithm will be of Monte Carlo (MC) type combined with a variance reduction technique. In particular, we use the control variates method presented in [17], where the control variate is given by

$$Y =: \left( \exp \left( \sum_{i=1}^d \omega_i \log(S_T^i) \right) - K \right)^+.$$

Since the only varying parameter is the vector of initial asset prices, it is very convenient to split the Monte Carlo simulation in two parts in order to make the completion more efficient. More precisely, in a pre-computation phase (Algorithm 3.1) we simulate a certain number of realizations (e.g.  $10^4$ ) of

$$\exp \left( \left( r - \frac{\sigma_i^2}{2} \right) T + \sigma_i W_T^i \right), \quad \text{for } i = 1, \dots, d,$$

and in a second moment we multiply the vector  $\mathbf{S}_0$  (for all required parameter combinations) with all the realizations and we compute the Monte Carlo price by applying the chosen variance reduction technique (Algorithm 3.2). In order to generate the correlated random variables  $W_T^i$ , we use the Cholesky factorization of the correlation matrix, which is then multiplied by a vector of independently generated standard normal distributed random variates. Note that  $\circ$  in Algorithm 3.2 represents the Hadamard (component-wise) product between vectors.

---

**Algorithm 3.1** Simulation of correlated geometric Brownian motions

---

**Input:** Model and payoff parameters  $\sigma, \Sigma, T, r$ ; number of simulations NumberSim.

**Output:** Matrix  $M \in \mathbb{R}^{\text{NumberSim} \times d}$  containing simulated random variables.

- 1:  $L \leftarrow$  Cholesky factor of  $\Sigma$
  - 2:  $M \leftarrow$  zeros(NumberSim,  $d$ )
  - 3: **for** iSim = 1: NumberSim **do**
  - 4:    $\epsilon \leftarrow$  Generate a vector of  $d$  independent standard normal variates
  - 5:    $x \leftarrow L\epsilon$
  - 6:   **for** iStock = 1:  $d$  **do**
  - 7:      $M(\text{iSim}, \text{iStock}) \leftarrow \exp \left( \left( r - \frac{\sigma(\text{iStock})^2}{2} \right) T + \sigma(\text{iStock}) x(\text{iStock}) \sqrt{T} \right)$
  - 8:   **end for**
  - 9: **end for**
- 

Algorithm 3.1 is executed at the beginning of the whole procedure and Algorithm 3.2 whenever needed in later stages. The advantage of splitting the MC algorithm is twofold. Firstly, it supports a considerable gain in efficiency in the performance of the completion algorithm: When we adaptively increment the sampling set  $\Omega$  (which consists of sampling Chebyshev nodes in  $\mathbf{S}_0$ ) in Algorithm 2.2, we need to compute new prices in the Chebyshev grid, which can be done by using Algorithm 3.2

---

**Algorithm 3.2** Computation of basket options using MC with control variate technique

---

**Input:** Matrix  $M$  from Algorithm 3.1,  $\mathbf{S}_0$ , strike  $K$ , vector of weights  $\omega$ ,  $r$ ,  $T$

**Output:** Basket option price (3.5)

```

1: payoff  $\leftarrow$  zeros(NumberSim, 1)
2: control  $\leftarrow$  zeros(NumberSim, 1)
3: for iSim = 1: NumberSim do
4:    $R \leftarrow$  iSim-th row of  $M$ 
5:    $S \leftarrow S_0 \circ R^T$ 
6:   payoff(iSim)  $\leftarrow$   $(\sum_{i=1}^d \omega_i S_i - K)^+$ 
7:   control(iSim)  $\leftarrow$   $(\exp(\sum_{i=1}^d \omega_i \log(S_i)) - K)^+$ 
8: end for
9: Compute mean  $\mu_Y$  of  $Y$  as explained in [17]
10: sum  $\leftarrow$  payoff - (control -  $\mu_Y$ )
11: Compute mean  $\mu$  of sum
12: Price  $\leftarrow$   $\exp(-rT)\mu$ 

```

---

only. The second advantage regards the analysis of the methodology and the completion accuracy: Since we use the same set of simulations for every Chebyshev price, the MC simulation does not introduce any further error to the completion. Moreover, we will see in Section 3.2.3 that this splitting procedure allows for a qualitative analysis of the rank structure of  $\mathcal{P}$ .

Next, we perform numerical experiments for different settings of model parameters, first for uncorrelated then for correlated assets.

**3.2.1. Basket options of uncorrelated assets.** In this example we consider the special case of uncorrelated assets. We investigate the performance of the proposed method for two different interpolation orders  $n_1 = \dots = n_d =: n = 4$  and  $n_1 = \dots = n_d =: n = 6$ . We apply the combined methodology (Algorithm 2.4) to portfolios consisting of  $d \in \{5, 10, 15, 20, 25\}$  assets. The set of fixed parameters is given by

$$T = 0.25, \quad K = 1, \quad r = 0, \quad \sigma_i = 0.2 \quad \forall i, \quad \Sigma = I_d, \quad \omega_i = \frac{1}{d} \quad \forall i,$$

where  $I_d$  denotes the  $d \times d$  identity matrix. We let  $\mathbf{S}_0$  vary in the hyper-rectangular

$$[1; 1.5]^d,$$

so that we consider ITM options and ATM options as well.

For each value of  $d$ , we start by performing Algorithm 3.1 with NumberSim =  $10^3$  for  $n = 4$  and with NumberSim =  $10^4$  for  $n = 6$ . In a second moment we construct the tensor  $\mathcal{P}$  by applying the tensor completion with the adaptive sampling strategy of Algorithm 2.2 (first strategy). Table 3.5 shows the completion parameters for each value of  $d$  and each interpolation order. The results of the tensor completions are displayed in Table 3.6. As in the previous subsection, we report the final size of the set  $\Omega$ , the relative error measured on the last set  $\Omega_C^{new}$ , the completion time and the memory needed to store both the obtained tensor in TT format and the full tensor. For the TT ranks of the completed tensor, we do not report the full tuple  $(r_0, \dots, r_d)$  (see Definition (2.3)) but only the quantity  $\max_{\mu \in \{0, \dots, d\}} r_\mu$ .

It is interesting to analyze the size of the finally obtained set  $\Omega$  in Algorithm 2.2 for different values of  $d$  and  $n$  (different sizes of  $\mathcal{P}$ ). Figure 3.1 shows a plot of  $|\Omega|$

TABLE 3.5  
Completion parameters for constructing  $\mathcal{P}$ . Case of uncorrelated assets.

	$d$	$\rho$	$tol$	$tol'$	$r_{\max}$	initial $ \Omega $	$ \Omega_C $	$p$
$n = 4$	5	0	$10^{-2}$	$10^{-8}$	5	31	31	$10^{-1}$
	10	0	$10^{-2}$	$10^{-8}$	5	78	78	$10^{-2}$
	15	0	$10^{-2}$	$10^{-8}$	5	214	214	$10^{-5}$
	20	0	$10^{-2}$	$10^{-8}$	5	763	763	$10^{-8}$
	25	0	$10^{-2}$	$10^{-8}$	5	2086	2086	$10^{-11}$
$n = 6$	5	0	$10^{-3}$	$10^{-8}$	7	17	17	$10^{-1}$
	10	0	$10^{-3}$	$10^{-8}$	7	282	141	$10^{-3}$
	15	0	$10^{-3}$	$10^{-8}$	7	475	475	$10^{-6}$
	20	0	$10^{-3}$	$10^{-8}$	7	798	798	$10^{-10}$
	25	0	$10^{-3}$	$10^{-8}$	7	1341	1341	$10^{-15}$

TABLE 3.6  
Completion results on  $\mathcal{P}$  for the basket option pricing problem in the Black and Scholes model. Case of uncorrelated assets.

	$d$	final $ \Omega $	rel err on last $\Omega_C^{new}$	completion time (s)
$n = 4$	5	124	$3.42 \cdot 10^{-3}$	9.90
	10	546	$2.54 \cdot 10^{-6}$	67.44
	15	1712	$3.55 \cdot 10^{-8}$	171.14
	20	2289	$5.03 \cdot 10^{-8}$	193.90
	25	4172	$3.96 \cdot 10^{-9}$	226.38
$n = 6$	5	204	$2.40 \cdot 10^{-4}$	52.55
	10	987	$1.20 \cdot 10^{-6}$	198.27
	15	1900	$2.28 \cdot 10^{-7}$	429.39
	20	3192	$2.97 \cdot 10^{-7}$	732.49
	25	4023	$1.35 \cdot 10^{-7}$	999.25

	$d$	max $r_\mu$ reached	store(TT) (bytes)	store(full) (bytes)
$n = 4$	5	5	2080	$2.50 \cdot 10^4$
	10	4	3440	$7.81 \cdot 10^7$
	15	4	5840	$2.44 \cdot 10^{11}$
	20	4	5800	$7.63 \cdot 10^{14}$
	25	4	10920	$2.38 \cdot 10^{18}$
$n = 6$	5	4	2688	$1.34 \cdot 10^5$
	10	6	9912	$2.26 \cdot 10^9$
	15	6	13720	$3.80 \cdot 10^{13}$
	20	5	11536	$6.38 \cdot 10^{17}$
	25	4	12600	$1.07 \cdot 10^{22}$

(final) against  $d$  for the two chosen interpolation orders. The graphical representation clearly suggests that the number of sampled entries, i.e.  $|\Omega|$ , required for the chosen tolerance  $tol = 10^{-2}$  for a fixed interpolation order  $n = 4$  and  $tol = 10^{-3}$  for a fixed  $n = 6$  is roughly of  $\mathcal{O}(d^2)$ , whereas the size of the full tensor is  $n^d$ . On the practical side, this means that by the completion algorithm we can reduce the complexity of

the first step of the offline phase from an exponential growth down to a quadratic growth in the dimensionality. The exponential growth typically is referred to as *curse of dimensionality*. The reduction in absolute numbers is already tremendous for  $d = 5$  and  $n = 4$ , where we observe  $|\Omega| = 124$  and the full tensor size equals  $(n+1)^d = 3125$ . The compression is dramatic for  $n = 6$  and  $d = 25$ , namely the numbers of required entries shrinks by a factor of more than  $3 \times 10^{17}$ .

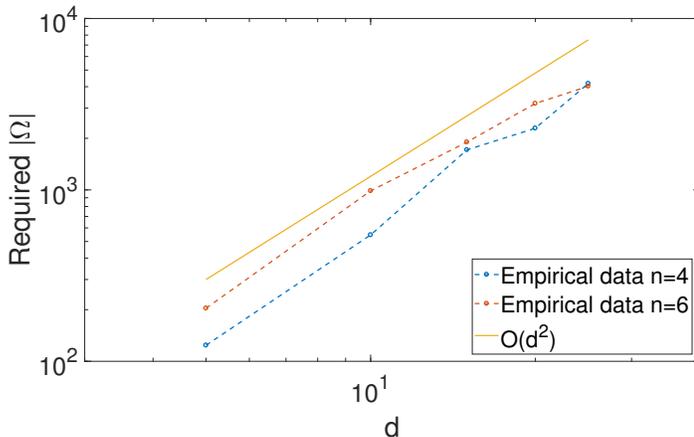


FIG. 3.1. Required size of  $\Omega$  for the completion to go below  $\text{tol} = 10^{-2}$  for  $n = 4$  and  $\text{tol} = 10^{-3}$  for  $n = 6$ . Case of uncorrelated assets.

As in the previous numerical example, the computation time to build the tensor  $\mathcal{C}$  of interpolation coefficients is negligible in the offline phase. Indeed, for all choices of  $d$  and  $n$  it is less than 0.01 seconds, for instance 0.0045 seconds for  $n = 4$  and  $d = 5$ , and 0.0095 seconds for  $n = 6$  and  $d = 25$ .

We now perform the online phase of Algorithm 2.4 in order to see how efficient becomes pricing basket options in the new setting. We start by computing 100 basket option prices via Chebyshev interpolation (combined methodology), choosing random initial asset prices  $\mathbf{S}_0$  in the reference hypercube  $[1; 1.5]^d$ . We then compare the obtained prices with reference prices computed by applying the reference method (Monte Carlo with control variates) with  $10^4$  new simulations for  $n = 4$  and  $10^5$  new simulations for  $n = 6$ . In particular, we measure again the maximal absolute error over all computed prices

$$\max(|P_{\text{Int}} - P_{\text{Ref}}|),$$

where  $P_{\text{Int}}$  is a vector containing all 100 interpolated prices for the different choices of  $\mathbf{S}_0$ ; analogously is  $P_{\text{Ref}}$  for the reference method. The errors together with the computational times are shown in Table 3.7. Note that we report again the computational time to compute one single option price.

One can see that the online phase of the new procedure compared to the MC reference method accelerates the computation of a factor between 200 and 400 for  $n = 4$  and of a factor between 2000 and 4000 for  $n = 6$ . Note that the difference in the acceleration between the two chosen interpolation orders is given by the different numbers of simulations chosen in the MC reference method ( $10^4$  for  $n = 4$  and  $10^5$  for  $n = 6$ ). Therefore, for both interpolation orders and for all choices of  $d$ , the

TABLE 3.7

Basket option prices computed via Chebyshev interpolation (combined methodology) versus MC reference method with  $10^4$  simulations for  $n = 4$  and  $10^5$  simulations for  $n = 6$ . Case of uncorrelated assets.

	$d$	time reference method (s)	time interpolation (s)	max abs error
$n = 4$	5	0.18	$0.45 \cdot 10^{-3}$	$3.75 \cdot 10^{-3}$
	10	0.19	$0.64 \cdot 10^{-3}$	$5.21 \cdot 10^{-4}$
	15	0.20	$0.73 \cdot 10^{-3}$	$4.38 \cdot 10^{-4}$
	20	0.20	$1.09 \cdot 10^{-3}$	$3.16 \cdot 10^{-4}$
	25	0.21	$0.97 \cdot 10^{-3}$	$2.08 \cdot 10^{-4}$
$n = 6$	5	1.84	$0.40 \cdot 10^{-3}$	$5.20 \cdot 10^{-4}$
	10	1.91	$0.61 \cdot 10^{-3}$	$1.42 \cdot 10^{-4}$
	15	1.99	$0.78 \cdot 10^{-3}$	$1.02 \cdot 10^{-4}$
	20	2.04	$0.93 \cdot 10^{-3}$	$1.01 \cdot 10^{-4}$
	25	2.10	$1.04 \cdot 10^{-3}$	$9.36 \cdot 10^{-5}$

acceleration is dramatic. In order to judge the accuracy of our method we have computed the 95% confidence interval of the reference method, which results to be of a size between  $10^{-4}$  and  $5 \cdot 10^{-4}$  for all choices of  $\mathbf{S}_0$  and  $d$  or  $n$ . This, together with the last column of Table 3.7, leads us to the conclusion that the new method is as accurate as the reference MC algorithm.

Finally, in Figure 3.2 we show the gain in efficiency of the new method when computing basket option prices for  $d = 25$  and both choices of interpolation orders. In particular, on the x-axis we consider a possible number of computed prices and on the y-axis we present

1. the computational time of the reference MC method,
2. the computational time of the new combined methodology (offline phase + online phase ),

required to compute the corresponding amount of prices.

The plots in Figure 3.2 show that after an initial investment the computational time grows very slowly in the number of computed prices for the new method. This is due to the fact that the online phase in Algorithm 4 is very cheap, as shown in the numerical experiments. This proves that the method is useful whenever one can split the task in a pre-computational phase during idle times and a run-time phase where execution is required to be fast. Moreover, it will outperform the reference methods if a large number of prices needs to be computed. The first plot in Figure 3.2 indicates that for the case  $n = 4$  it is convenient to use the reference MC method if we want to compute up to 1000 option prices. For the case  $n = 6$  the break-even point is already reached with 500 prices.

**3.2.2. Basket options of correlated assets.** In this second numerical experiment we repeat the test of the previous subsection but, this time, we consider correlated assets. In particular, we choose again the interpolation orders  $n = 4$ ,  $n = 6$  and the other parameters are given by

$$T = 0.25, \quad K = 1, \quad r = 0, \quad \sigma_i = 0.2 \quad \forall i, \quad \Sigma = R_d, \quad \omega_i = \frac{1}{d} \quad \forall i,$$

where  $R_d$  denotes a random correlation matrix. The free parameters  $S_0^i$ ,  $i = 1, \dots, d$  are again contained in  $[1;1.5]$ . We perform the offline phase by considering again

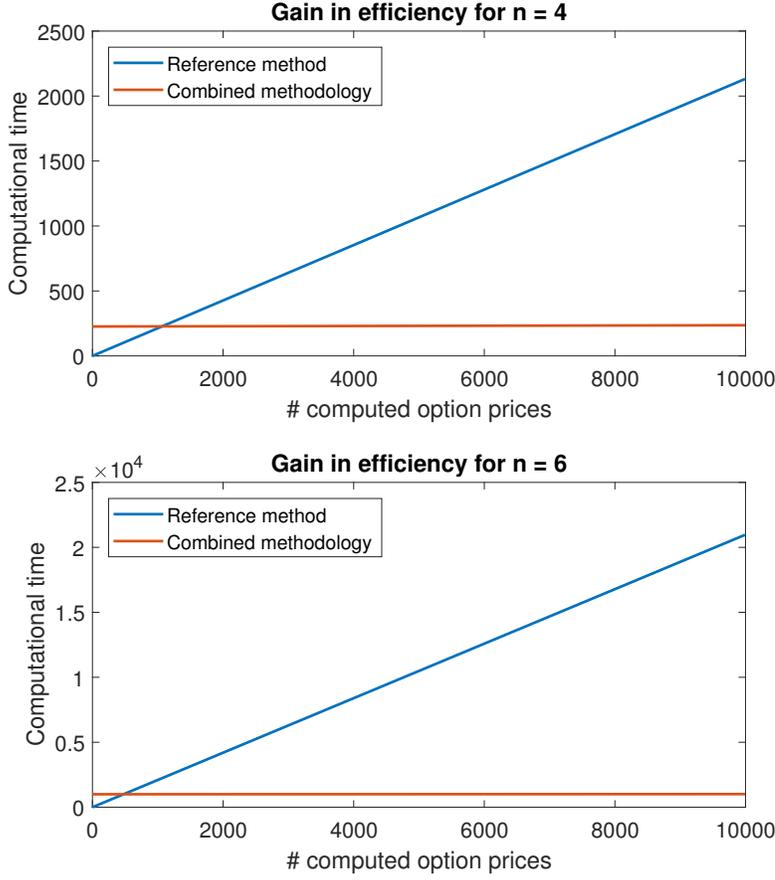


FIG. 3.2. Computational time for computing basket option prices. Comparison MC versus combined methodology for  $n = 4$  and  $n = 6$ . Case of uncorrelated assets.

the set of completion parameters listed in Table 3.5. The obtained results of the completion are now in Table 3.8 and Figure 3.3 shows the required size of  $\Omega$  to go below the tolerance  $tol = 10^{-2}$  for  $n = 4$  and  $tol = 10^{-3}$  for  $n = 6$ . We notice that the completion results are similar to the case of uncorrelated assets and that  $|\Omega|$  scales again like  $\mathcal{O}(d^2)$ . The computational time to construct  $\mathcal{C}$  was again measured to be less than 0.01 seconds for all choices of  $d$  and  $n$ .

The online phase is performed similarly to the previous chapter, in particular we compute again 100 prices using the new method and the reference one. The MC parameters are set as before and the results are shown in Table 3.9. The performance of the new method in terms of accuracy and computational efficiency is similar to the one observed in the case of uncorrelated assets. To summarize, the new methodology achieves a very good performance for uncorrelated as well as for correlated assets.

**3.2.3. Rank structure of  $\mathcal{P}$ .** In this section we qualitatively analyze the rank structure of the tensor  $\mathcal{P}$ . For simplicity, we perform this analysis for the standard Monte Carlo approach (without any variance reduction technique). Assume that we

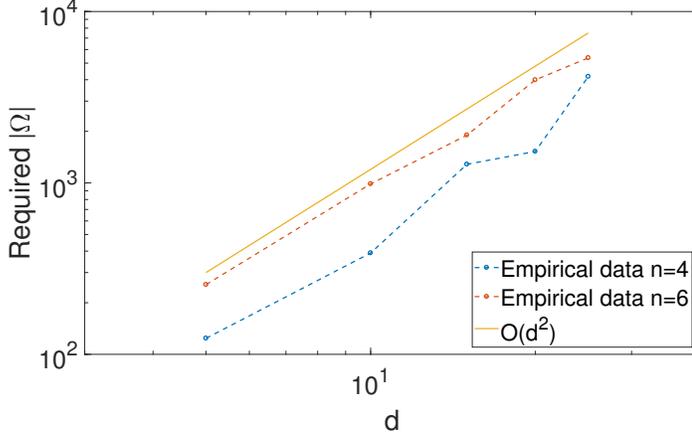


FIG. 3.3. Required size of  $\Omega$  for the completion to go below  $\text{tol} = 10^{-2}$  for  $n = 4$  and  $\text{tol} = 10^{-3}$  for  $n = 6$ . Case of correlated assets..

have already simulated the realizations of the correlated geometric Brownian motions stored in the matrix  $M$  (Algorithm 3.1). Then, the price in the point  $\mathbf{S}_0$  is given by the function

$$p : D \rightarrow \mathbb{R},$$

$$p(\mathbf{S}_0) := \frac{e^{-rT}}{N_S} \sum_{n=1}^{N_S} [w^T(\mathbf{S}_0 \circ M(n, :)^T) - K]^+,$$

where  $D$  is the hyper-rectangular domain for the interpolation,  $M(n, :)$  is the  $n$ -th row of  $M$  and  $N_S$  is the number of Monte Carlo simulations. This expression can be rewritten in the form

$$p(\mathbf{S}_0) = \frac{e^{-rT}}{N_S} \sum_{n=1}^{N_S} \left( \sum_{i=1}^d \alpha_i(n) S_0^i - K \right)^+,$$

where the  $\alpha_i(n)$ 's are coefficients multiplying  $S_0^i$  depending on the  $n$ -th simulation and on the  $i$ -th weight  $\omega_i$ . The function  $p$  is piecewise affine in the variables  $S_0^i$ .

To explore the rank structure of  $\mathcal{P}$  let us consider the case of a single Monte Carlo simulation  $N_S = 1$ . Then  $p$  is of the form

$$p(\mathbf{S}_0) = e^{-rT} \left( \sum_{i=1}^d \alpha_i S_0^i - K \right)^+.$$

Now we analyze three different cases. First, consider the case where the price is positive for any  $\mathbf{S}_0$  in the hyper-rectangular  $D$ . Here,  $p$  is affine. This implies that the TT ranks are bounded by  $d$ . This follows from the fact that the CP rank (rank of the Canonical Polyadic Decomposition, see [31]) of  $\mathcal{P}$ , which is an upper bound for each  $r_\mu$  in the TT ranks (see [20]), is equal to  $d$ . Second, if we observe a vanishing price for all  $\mathbf{S}_0$  in the hyper-rectangular, then  $\mathcal{P}$  is the zero-tensor, which has rank 0. These two cases obviously yield a low-rank structure of  $\mathcal{P}$ , a favorable case for the new combined methodology.

TABLE 3.8

Completion results on  $\mathcal{P}$  for the basket option pricing problem in the Black and Scholes model. Case of correlated assets.

	$d$	final $ \Omega $	rel err on last $\Omega_C^{new}$	completion time (s)
$n = 4$	5	124	$1.86 \cdot 10^{-3}$	8.95
	10	390	$2.19 \cdot 10^{-4}$	65.73
	15	1284	$1.72 \cdot 10^{-7}$	118.73
	20	1526	$2.49 \cdot 10^{-8}$	168.20
	25	4172	$7.52 \cdot 10^{-9}$	215.44
$n = 6$	5	255	$4.40 \cdot 10^{-4}$	66.54
	10	987	$2.06 \cdot 10^{-4}$	200.15
	15	1900	$1.79 \cdot 10^{-7}$	432.58
	20	3990	$1.82 \cdot 10^{-8}$	852.13
	25	5364	$2.88 \cdot 10^{-7}$	1335.76

	$d$	max $r_\mu$ reached	store(TT) (bytes)	store(full) (bytes)
$n = 4$	5	5	2320	$2.50 \cdot 10^4$
	10	3	2440	$7.81 \cdot 10^7$
	15	5	8960	$2.44 \cdot 10^{11}$
	20	4	7040	$7.63 \cdot 10^{14}$
	25	3	8040	$2.38 \cdot 10^{18}$
$n = 6$	5	5	2520	$1.34 \cdot 10^5$
	10	5	6832	$2.26 \cdot 10^9$
	15	4	6664	$3.80 \cdot 10^{13}$
	20	4	12040	$6.38 \cdot 10^{17}$
	25	4	8960	$1.07 \cdot 10^{22}$

In the third case where  $p$  is only piecewise affine the situation is more complex and to gain an intuition we consider the case  $d = 2$ , where  $p$  is of the form

$$p(S_0^1, S_0^2) = e^{-rT}(\alpha_1 S_0^1 + \alpha_2 S_0^2 - K)^+,$$

on a squared domain  $D$ . Now, define the set

$$L := \{(S_0^1, S_0^2) \in D \mid \alpha_1 S_0^1 + \alpha_2 S_0^2 - K = 0\}.$$

When  $L$  intersects the domain  $D$  it cuts it in two regions. Only if  $\alpha_1, \alpha_2$  and  $K$  are of a specific form that leads  $L$  to be the diagonal of  $D$ , the rank of  $\mathcal{P}$  is almost full. In the Monte Carlo simulation context, this special case is very unlikely. In all other cases,  $\mathcal{P}$  exhibits a lower rank structure. In particular, we expect the rank to be the lower the more the sizes of the two regions differ.

In order to visualize these findings we consider three different pairs  $(\alpha_1, \alpha_2)$  together with  $r = 0$ ,  $K = 1$  and evaluate the corresponding  $p$  on the discretized  $D = [1; 1.5]^2$  using 50 equidistant points in each direction. Figure 3.4 shows the sparsity pattern and the rank of the obtained matrices  $\mathcal{P}$ .

This qualitative explanation indicates that the rank structure of  $\mathcal{P}$  depends on  $D$ . We expect the rank to be lower for domains  $D$  with an asymmetry with respect to the strike  $K$ . Next we construct  $\mathcal{P}$  as in the experiments of Section 3.2 for  $K = 1$ ,

TABLE 3.9

Basket option prices computed via Chebyshev interpolation (combined methodology) versus MC reference method with  $10^4$  simulations for  $n = 4$  and  $10^5$  simulations for  $n = 6$ . Case of correlated assets.

	$d$	time reference method (s)	time interpolation (s)	max abs error
$n = 4$	5	0.18	$0.50 \cdot 10^{-3}$	$1.39 \cdot 10^{-3}$
	10	0.20	$0.56 \cdot 10^{-3}$	$4.82 \cdot 10^{-4}$
	15	0.20	$0.70 \cdot 10^{-3}$	$2.82 \cdot 10^{-4}$
	20	0.23	$0.91 \cdot 10^{-3}$	$2.93 \cdot 10^{-4}$
	25	0.23	$1 \cdot 10^{-3}$	$4.30 \cdot 10^{-4}$
$n = 6$	5	1.85	$0.38 \cdot 10^{-3}$	$3.55 \cdot 10^{-4}$
	10	1.90	$0.57 \cdot 10^{-3}$	$5.58 \cdot 10^{-4}$
	15	1.99	$0.74 \cdot 10^{-3}$	$1.39 \cdot 10^{-4}$
	20	2.06	$0.90 \cdot 10^{-3}$	$1.41 \cdot 10^{-4}$
	25	2.15	$0.96 \cdot 10^{-3}$	$9.28 \cdot 10^{-5}$

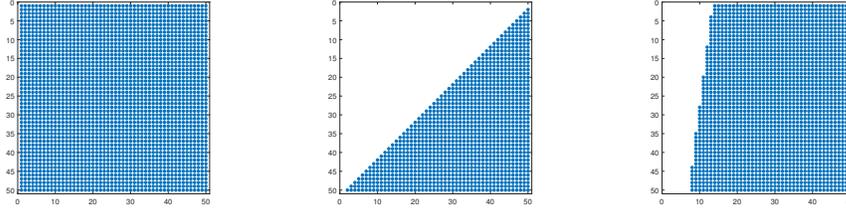


FIG. 3.4. Sparsity patterns and ranks for evaluated  $p$  on  $D = [1; 1.5]^2$  for different values of  $(\alpha_1, \alpha_2)$ . Left:  $(\alpha_1, \alpha_2) = (0.9, 0.8)$  and rank = 2. Center:  $(\alpha_1, \alpha_2) = (0.4, 0.4)$  and rank = 49. Right:  $(\alpha_1, \alpha_2) = (0.1, 0.8)$  and rank = 8.

$d = 2$  and different interpolation orders  $n$  for both  $D = [0.5; 1.5]^2$  and  $D = [1; 1.5]^2$ . In particular, we first construct the matrix  $M$  via Algorithm 3.1 with  $10^5$  simulations and subsequently compute  $\mathcal{P}$  using Algorithm 3.2. In Figure 3.5 we display the decay of the singular values for all treated cases. As expected, the decay is faster for  $D = [1; 1.5]^2$ . However, also for  $D = [0.5; 1.5]^2$  the decay of the singular values is reasonably fast. This implies that the new methodology would be still beneficial in this case.

**4. Summary and future work.** We have presented a unified approach to efficiently compute parametric option prices. The starting point of our methodology was the Chebyshev interpolation technique developed in [14], which we briefly summarized in Section 2.1. We refined both the offline and the online phase to treat high-dimensional problems with parameter spaces up to dimension 25. We have exploited the low-rank structure of the tensors involved in the interpolation procedure, which have been stored in TT format (summarized in Section 2.2). In particular, we have developed a completion technique (explained in Section 2.3) which allows us to construct the tensor  $\mathcal{P}$ , containing the option prices in the Chebyshev tensor grid. All ingredients have been efficiently assembled to finally build a combined methodology, explained in Section 2.4.

In the second part of the paper, Section 3, we have tested our approach in two different concrete option pricing settings: We have treated the American option pricing

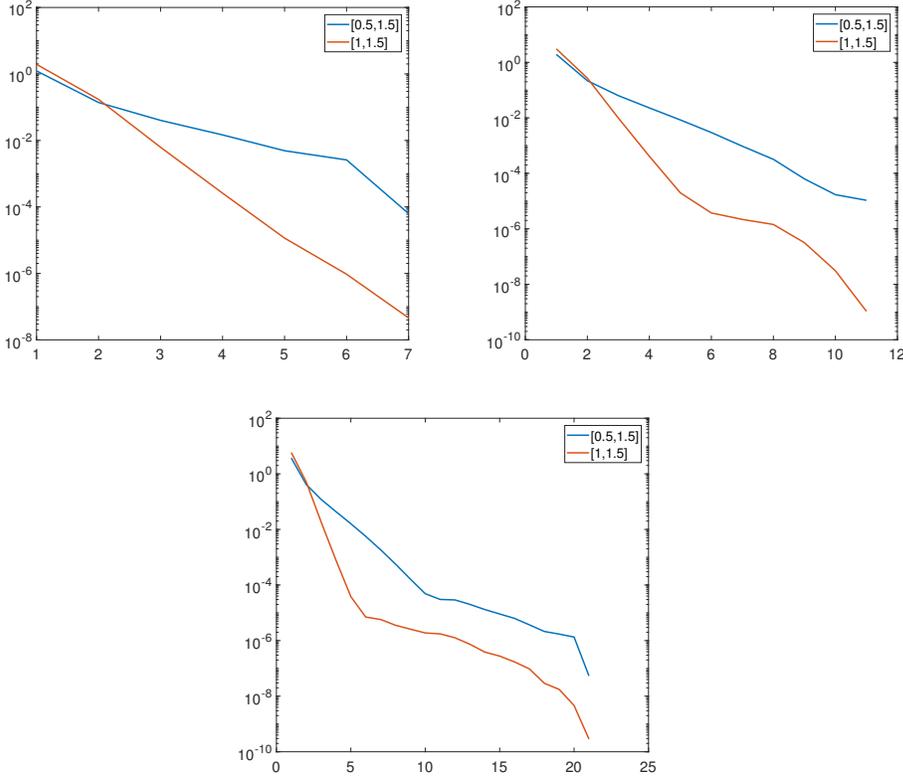


FIG. 3.5. Singular value decay of the matrix  $\mathcal{P}$  for sampling intervals  $[0, 5; 1, 5]$  and  $[1; 1.5]$  for different interpolation orders: Top left:  $n = 6$ , Top right:  $n = 10$ , Bottom:  $n = 20$ .

problem in the Heston model (Section 3.1) and the European basket option pricing problem in the  $d$ -dimensional Black and Scholes model (Section 3.2). Both examples show that our approach allows for a substantial gain in efficiency, while maintaining very accurate results, whose precision is comparable to the one of the considered reference methods. For instance, the interpolation of American option prices in 5 parameters accelerates the procedure by a factor of  $(75) \leftarrow 94$ , when compared to the FD reference method [21]. For basket option pricing with 25 underlyings the efficiency gain reaches factors up to 4000. See Tables 3.4, 3.7, 3.9 and Figure 3.2 for further results. Finally, for both examples we qualitatively investigated the rank structure of  $\mathcal{P}$ , which confirmed that our initial low-rank assumption was indeed reasonable. For instance, for the American put, we obtain a compression factor of 115 of the completed tensor  $\mathcal{P}$  with respect to the full one, with a relative error in the 5th digit only, see Table 3.2. For the basket option the full tensor containing prices in the Chebyshev grid is too large to be computed, however in Section 3.2.3 it is qualitatively explained why  $\mathcal{P}$  is expected to have a low-rank structure. This is also confirmed by the compression rates observed in Tables 3.6 and 3.8 that go up to  $3 \times 10^{17}$ .

Seen the promising performance of this new approach and considering the fact that this methodology can be easily tailored to different problem settings, we expect it to be applicable in several domains in finance. For instance, pricing, calibration

and sensitivity analysis in equity markets, fixed income and credit, and parameter uncertainty quantification are some of the possible domains of application.

#### REFERENCES

- [1] M. BACHMAYR AND A. COHEN, *Kolmogorov widths and low-rank approximations of parametric elliptic PDEs*, Math. Comp., 86 (2017), pp. 701–724, <https://doi.org/10.1090/mcom/3132>.
- [2] B. W. BADER AND T. G. KOLDA, *Algorithm 862: MATLAB tensor classes for fast algorithm prototyping*, ACM Transactions on Mathematical Software, 32 (2006), pp. 635–653, <https://doi.org/10.1145/1186785.1186794>.
- [3] B. W. BADER, T. G. KOLDA, ET AL., *Matlab tensor toolbox version 2.6*. Available online, February 2015, <http://www.sandia.gov/~tgkolda/TensorToolbox/>.
- [4] J. BALLANI AND L. GASEDYCK, *Hierarchical tensor approximation of output quantities of parameter-dependent PDEs*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 852–872, <https://doi.org/10.1137/140960980>.
- [5] D. BARRERA, S. CRÉPEY, B. DIALLO, G. FORT, E. GOBET, AND U. STAZHYNSKI, *Stochastic approximation schemes for economic capital and risk margin computations*. Forthcoming in ESAIM: Proceedings and Surveys, <https://math.maths.univ-evry.fr/crepey/papers/SA-EC-RM.pdf>, 2019.
- [6] C. BAYER, M. SIEBENMORGEN, AND R. TEMPONE, *Smoothing the payoff for efficient computation of basket option prices*, Quant. Finance, 18 (2018), pp. 491–505, <https://doi.org/10.1080/14697688.2017.1308003>.
- [7] O. BURKOVSKA, K. GLAU, M. MAHLSTEDT, AND B. WOHLMUTH, *Complexity reduction for calibration of American options*. Forthcoming in J. Comput. Finance, <https://arxiv.org/abs/1611.06452>, 2017.
- [8] O. BURKOVSKA, B. HAASDONK, J. SALOMON, AND B. WOHLMUTH, *Reduced basis methods for pricing options with the Black-Scholes and Heston models*, SIAM J. Financial Math., 6 (2015), pp. 685–712, <https://doi.org/10.1137/140981216>.
- [9] L. CAPRIOTTI, Y. JIANG, AND A. MACRINA, *AAD and least-square Monte Carlo: fast Bermudan-style options and XVA Greeks*, Algorithmic Finance, 6 (2017), pp. 35–49, <https://doi.org/10.3233/af-170201>.
- [10] R. CONT, N. LANTOS, AND O. PIRONNEAU, *A reduced basis for option pricing*, SIAM J. Financial Math., 2 (2011), pp. 287–316, <https://doi.org/10.1137/10079851X>.
- [11] W. DAHMEN, R. DEVORE, L. GASEDYCK, AND E. SÜLI, *Tensor-sparsity of solutions to high-dimensional elliptic partial differential equations*, Found. Comput. Math., 16 (2016), pp. 813–874, <https://doi.org/10.1007/s10208-015-9265-9>.
- [12] M. A. H. DEMPSTER, J. KANNIAINEN, J. KEANE, AND E. VYNCKIER, *High-Performance Computing in Finance: Problems, Methods, and Solutions*, Chapman & Hall/CRC, 1st ed., 2018.
- [13] D. J. DUFFY, *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*, Wiley Finance Series, John Wiley & Sons, Ltd., Chichester, 2006, <https://doi.org/10.1002/9781118673447>.
- [14] M. GASS, K. GLAU, M. MAHLSTEDT, AND M. MAIR, *Chebyshev interpolation for parametric option pricing*, Finance Stoch., 22 (2018), pp. 701–731, <https://doi.org/10.1007/s00780-018-0361-y>.
- [15] M. B. GILES, *Multilevel Monte Carlo methods*, Acta Numer., 24 (2015), pp. 259–328, <https://doi.org/10.1017/S096249291500001X>.
- [16] M. B. GILES AND Y. XIA, *Multilevel Monte Carlo for exponential Lévy models*, Finance Stoch., 21 (2017), pp. 995–1026, <https://doi.org/10.1007/s00780-017-0341-7>.
- [17] P. GLASSERMAN, *Monte Carlo methods in financial engineering*, vol. 53 of Applications of Mathematics, Springer-Verlag, New York, 2004. Stochastic Modelling and Applied Probability.
- [18] L. GASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78, <https://doi.org/10.1002/gamm.201310004>.
- [19] M. GRIEBEL AND M. HOLTZ, *Dimension-wise integration of high-dimensional functions with applications to finance*, J. Complexity, 26 (2010), pp. 455–489, <https://doi.org/10.1016/j.jco.2010.06.001>.
- [20] W. HACKBUSCH, *Tensor spaces and numerical tensor calculus*, vol. 42 of Springer Series in Computational Mathematics, Springer, Heidelberg, 2012, <https://doi.org/10.1007/978-3-642-28027-6>.

- [21] T. HAENTJENS AND K. J. IN'T HOUT, *ADI schemes for pricing American options under the Heston model*, Appl. Math. Finance, 22 (2015), pp. 207–237, <https://doi.org/10.1080/1350486X.2015.1009129>.
- [22] B. HASHEMI AND L. N. TREFETHEN, *Chebfun in three dimensions*, SIAM J. Sci. Comput., 39 (2017), pp. C341–C363, <https://doi.org/10.1137/16M1083803>.
- [23] J. S. HESTHAVEN, G. ROZZA, AND B. STAMM, *Certified reduced basis methods for parametrized partial differential equations*, SpringerBriefs in Mathematics, Springer, Cham; BCAM Basque Center for Applied Mathematics, Bilbao, 2016, <https://doi.org/10.1007/978-3-319-22470-1>. BCAM SpringerBriefs.
- [24] S. L. HESTON, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, Review of Financial Studies, 6 (1993), pp. 327–343.
- [25] N. HILBER, N. REICH, C. SCHWAB, AND C. WINTER, *Numerical methods for Lévy processes*, Finance Stoch., 13 (2009), pp. 471–500, <https://doi.org/10.1007/s00780-009-0100-5>.
- [26] N. HILBER, O. REICHMANN, C. SCHWAB, AND C. WINTER, *Computational Methods for Quantitative Finance*, Springer Finance, Springer, Heidelberg, 2013, <https://doi.org/10.1007/978-3-642-35401-4>. Finite element methods for derivative pricing.
- [27] M. HOLTZ, *Sparse grid quadrature in high dimensions with applications in finance and insurance*, vol. 77 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, 2011, <https://doi.org/10.1007/978-3-642-16004-2>.
- [28] K. IN'T HOUT AND J. TOIVANEN, *Application of operator splitting methods in finance*, Sci. Comput., Springer, Cham, 2016, pp. 541–575.
- [29] B. N. KHOROMSKIJ, *Tensor numerical methods in scientific computing*, vol. 19 of Radon Series on Computational and Applied Mathematics, De Gruyter, Berlin, 2018.
- [30] B. N. KHOROMSKIJ AND C. SCHWAB, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput., 33 (2011), pp. 364–385, <https://doi.org/10.1137/100785715>.
- [31] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500, <https://doi.org/10.1137/07070111X>.
- [32] D. KRESSNER, M. STEINLECHNER, AND B. VANDEREYCKEN, *Low-rank tensor completion by Riemannian optimization*, BIT, 54 (2014), pp. 447–468, <https://doi.org/10.1007/s10543-013-0455-z>.
- [33] D. KRESSNER, M. STEINLECHNER, AND B. VANDEREYCKEN, *Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure*, SIAM J. Sci. Comput., 38 (2016), pp. A2018–A2044, <https://doi.org/10.1137/15M1032909>.
- [34] D. KRESSNER AND C. TOBLER, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316, <https://doi.org/10.1137/100799010>.
- [35] P. L'ECUYER, *Quasi-Monte Carlo methods with applications in finance*, Finance Stoch., 13 (2009), pp. 307–349, <https://doi.org/10.1007/s00780-009-0095-y>.
- [36] A.-M. MATACHE, P.-A. NITSCHKE, AND C. SCHWAB, *Wavelet Galerkin pricing of American options on Lévy driven assets*, Quant. Finance, 5 (2005), pp. 403–424, <https://doi.org/10.1080/14697680500244478>.
- [37] A. MAYERHOFER AND K. URBAN, *A reduced basis method for parabolic partial differential equations with parameter functions and application to option pricing*, J. Comput. Finance, 20 (2017), pp. 71–106.
- [38] R. ORÚS, *A practical introduction to tensor networks: matrix product states and projected entangled pair states*, Ann. Physics, 349 (2014), pp. 117–158, <https://doi.org/10.1016/j.aop.2014.06.013>.
- [39] I. V. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, <https://doi.org/10.1137/090752286>.
- [40] A. T. PATERA AND G. ROZZA, *Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations*, Tech. Report Version 1.0, MIT 2006–2007, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering, Massachusetts Institute of Technology, 2006, [http://augustine.mit.edu/methodology/bookParts/Patera\\_Rozza\\_bookPartI.BV1.pdf](http://augustine.mit.edu/methodology/bookParts/Patera_Rozza_bookPartI.BV1.pdf).
- [41] A. QUARTERONI, A. MANZONI, AND F. NEGRI, *Reduced basis methods for partial differential equations*, vol. 92 of Unitext, Springer, Cham, 2016, <https://doi.org/10.1007/978-3-319-15431-2>. An introduction, La Matematica per il 3+2.
- [42] C. REISINGER AND R. WISSMANN, *Numerical valuation of derivatives in high-dimensional settings via PDE expansions*, J. Comput. Finance, 18 (2015), pp. 95–127.
- [43] E. W. SACHS AND M. SCHU, *Reduced order models (POD) for calibration problems in finance*, in Numerical mathematics and advanced applications, Springer, Berlin, 2008, pp. 735–742.

- [44] R. SCHNEIDER AND A. USCHMAJEW, *Approximation rates for the hierarchical tensor format in periodic Sobolev spaces*, J. Complexity, 30 (2014), pp. 56–71, <https://doi.org/10.1016/j.jco.2013.10.001>.
- [45] M. STEINLECHNER, *Riemannian optimization for high-dimensional tensor completion*, SIAM J. Sci. Comput., 38 (2016), pp. S461–S484, <https://doi.org/10.1137/15M1010506>.
- [46] M. STEINLECHNER, *Riemannian Optimization for Solving High-Dimensional Problems with Low-Rank Tensor Structure*, PhD thesis, École polytechnique fédérale de Lausanne, 2016.
- [47] A. USCHMAJEW AND B. VANDEREYCKEN, *Greedy rank updates combined with Riemannian descent methods for low-rank optimization*, in Proceedings of the 2015 International Conference on Sampling Theory and Applications (SampTA), IEEE, 2015, pp. 420–424.