SoTT: greedy approximation of a tensor as a sum of Tensor Trains

Virginie Ehrlacher, Maria Fuente-Ruiz, Damiano Lombardi

SOTT: GREEDY APPROXIMATION OF A TENSOR AS A SUM OF TENSOR TRAINS

VIRGINIE EHRLACHER*, MARIA FUENTE RUIZ†, AND DAMIANO LOMBARDI ‡

Abstract. In the present work, a method is proposed in order to compute an approximation of a given tensor as a sum of Tensor Trains (TTs), where the order of the variates and the values of the ranks can vary from one term to the other in an adaptive way. The numerical scheme is based on a greedy algorithm and an adaptation of the TT-SVD method. The proposed approach can also be used in order to compute an approximation of a tensor in a Canonical Polyadic format (CP), as an alternative to standard algorithms like Alternating Linear Squares (ALS) or Alternating Singular Value Decomposition (ASVD) methods. Some numerical experiments are proposed, in which the proposed method is compared to ALS and ASVD methods for the construction of a CP approximation of a given tensor and performs particularly well for high-order tensors. The interest of approximating a tensor as a sum of Tensor Trains is illustrated in several numerical test cases.

Key words. Tensor methods, Canonical Polyadic, Tensor Train.

AMS subject classifications. 65F99, 65D15

1. Introduction.

Machine learning and data mining algorithms are becoming increasingly important in analyzing large volume, multi-relational and multi-modal datasets, which are often conveniently represented as multiway arrays or tensors [5, 19, 20].

The main challenge in dealing with such data is the so called *curse of dimensionality*, that refers to the need of using a number of degrees of freedom exponentially increasing with the dimension [23]. This problem can be alleviated by using various tensor formats, achieved by low-rank tensor approximations, for the compression of the full tensor as described for instance in [18, 4, 7, 11]. The definition of these different tensor formats relies on the well-known separation of variables principle. We refer the reader to [13] and [16] for extensive reviews on tensor theory and extended analysis of tensor decompositions and their numerous applications.

Among the different existing tensor formats, two of them are of specific importance with respect to applications, namely the Canonical Polyadic (CP) and Tensor Train (TT) format. The main advantage of these decompositions is the low memory cost needed to store them. In the case of the CP format, this cost only scales linearly with the order of the tensor, whereas the memory cost for the storage of a full tensor scales exponentially with its order. However, the problem of finding a best approximation of a tensor in CP format may be ill-posed [6] and leads to numerical instabilities. The most classical algorithm in order to compute an approximation of a tensor in the CP format is the so-called Alternating Least Square (ALS) method, which sometimes may be quite slow to converge [2] especially for high-order tensors. Some alternative methods [32, 26] have been proposed in order to obtain more efficient algorithms.

The Tensor Train format is probably one of the most used tensor formats in realistic applications [3, 34, 27], due to a good trade off between optimality and numerical stability. The TT format combines two advantages to take into consideration: on the one hand, it is stable from an algorithmic point of view; on the other, it is computationally affordable provided that the TT ranks of the tensors remain reasonably

*Ecole des Ponts ParisTech & INRIA (Matherials project-team) (virginie.ehrlacher@enpc.fr).
†COMMEDIA, INRIA Paris, France & LJLL, Sorbonne Université (maria.fuente-ruiz@inria.fr).
‡COMMEDIA, INRIA Paris, France & LJLL, Sorbonne Université (damiano.lombardi@inria.fr).

small. The computation of the approximation of a tensor in the TT format is usually done via the so-called TT-SVD algorithm. One of the drawback of the TT-format is that it requires a priori the choice of a particular order in the variables of the tensor, and the quality of the resulting approximation computed by a TT-SVD algorithm strongly depends on this particular choice. Even if the number of entries could be larger than in CP, the main advantage of the TT format is its ability to provide stable quasi-optimal rank reduction, obtained, for instance, by truncated singular value decompositions.

In the literature, hybrid formats combining CP with other methods have been proposed in [25], and described in [12]. Also, CP has been combined with TT in [21], where its was highlighted that the combination of both methods yields interesting improvements. Fast algorithms for the rank truncation in the canonical input tensors with large CP-ranks and large mode size, have been introduced and analyzed in [17]. Some other optimization-based algorithms could be seen in [29].

The main contribution of the present work is a numerical scheme that constructs an approximation of a tensor as a sum of TTs, called the Sum of Tensor Trains (SoTT) scheme, where the order of the variables and the values of the ranks can vary from one term to another and can be adaptively chosen by an algorithm which combines the TT-SVD algorithm together with a greedy procedure (see [30]). The interest of such a procedure is two-fold: (i) it enables to select in an adaptive way the order of the variables in each term so as to obtain favorable compressing rates with respect to pure TT approximations with an a priori prescriber order of variables; (ii) when the values of the ranks of the terms computed are fixed to be equal to one, the procedure provides a new scheme for the computation of a CP approximation of a given tensor, which appears to be more efficient than ALS for high-order tensors. This work is also motivated by applications in quantum chemistry, where approximate ground state electronic wave functions are computed within the so-called DMRG using tensor networks. However, computing pair, triplet or quadruplet densities, and more generally multivariable correlations, for general tensor networks may be quite intricate, whereas it remains quite simple to carry out when the electronic wave function is computed as a sum of TTs. We observe numerically that this algorithm performs well in practice in the sense that it provides a more accurate approximation of a given tensor, at fixed memory storage cost, than a TT-SVD algorithm, in average, when the order of the variables in the TT decomposition is chosen randomly.

We also observed that a particular version of the SoTT algorithm, named CP-TT, which consists in adding pure rank-1 tensor-product at each iteration of the scheme, can be used in order to compute a CP approximation of a given tensor. Such a scheme gives interesting results in comparison with other rank-1 update methods such as ALS for instance, especially when the order of the tensor is high.

The article is structured as follows: some preliminaries about tensors are recalles in Section 2. The SoTT algorithm is presented and discussed in Section 3. The CP-TT version of SoTT is discussed in Section 4 and compared with other numerical methods used for the construction of CP decompositions. Numerical experiments and results illustrating the efficiency of the approach are given in Section 5.

2. Preliminaries.

The aim of this section is to recall some preliminaries before introducing the SoTT algorithm. We begin by introducing some notation together with the well-known Singular Value Decomposition in Section 2.1. We then recall some basic facts about the Canonical Polyadic (CP) and Tensor Train (TT) format in Section 2.2. The classical TT-SVD algorithm is then recalled in Section 2.3.

2.1. Notation and Singular Value Decomposition (SVD). We begin by introducing here some notation which will be used in the sequel. For any $p \in \mathbb{N}^*$, and any family $(D_1, \cdots, D_p)$ such that for all $1 \leq i \leq p$, Ω_i is an open bounded subset of $\mathbb{R}^{p_i}$ for some $p_i \in \mathbb{N}^*$, we call a *tensor* (with a slight language abuse) any real-valued function $F \in L^2(D_1 \times \cdots \times D_p)$. Denoting by $D := D_1 \times \cdots \times D_p$, we denote $\langle \cdot, \cdot \rangle_D$ the scalar product of $L^2(D)$ and by $\| \cdot \|_D$ the associated norm so that for all $U, V \in L^2(D)$,

$$\langle U, V \rangle_D := \int_D UV \quad \text{and} \quad \|U\|_D := \left(\int_D U^2 \right)^{1/2}.$$

For any $u^{(1)} \in L^2(D_1), \cdots, u^{(p)} \in L^2(D_p)$, we denote $u^{(1)} \otimes \cdots \otimes u^{(p)} \in L^2(D)$ the *pure tensor product function* defined by

$$u^{(1)} \otimes \cdots \otimes u^{(p)} : \begin{cases} D = D_1 \times \cdots \times D_p & \to & \mathbb{R} \\ (x_1, \cdots, x_p) & \mapsto & u^{(1)}(x_1) \cdots u^{(d)}(x_p). \end{cases}$$

Moreover, we make use of the following abuse of notation. For any nonempty subset $\mathcal{I} \subset \{1, \cdots, p\}$ such that $\mathcal{I}^c = \{1, \cdots, p\} \backslash \mathcal{I}$ is non-empty, and any $F \in L^2(D_1 \times \cdots D_p)$, we still denote F the function $\widetilde{F} \in L^2\left((\times_{i \in \mathcal{I}} D_i) \times (\times_{j \in \mathcal{I}^c} D_j) \right)$ defined by

$$\widetilde{F} : \begin{cases} (\times_{i \in \mathcal{I}} D_i) \times (\times_{j \in \mathcal{I}^c} D_j) & \to & \mathbb{R} \\ ((x_i)_{i \in \mathcal{I}}, (x_j)_{j \in \mathcal{I}^c}) & \mapsto & F(x_1, \cdots, x_p). \end{cases}$$

For any domain $D = D_x \times D_y$, where D_x and D_y are open subdomains of $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$ for some $d_x, d_y \in \mathbb{N}^*$ respectively, and any $W \in L^2(D)$, it holds that there exists an orthonormal basis $(U_k)_{k \in \mathbb{N}^*}$ of $L^2(D_x)$, an orthonormal basis $(V_k)_{k \in \mathbb{N}^*}$ of $L^2(D_y)$ and a non-increasing sequence $(\sigma_k)_{k \in \mathbb{N}^*}$ of non-negative real numbers which converges to 0 as k goes to ∞, such that

$$W = \sum_{k \in \mathbb{N}^*} \sigma_k U_k \otimes V_k. \tag{2.1}$$

A decomposition of W under the form (2.1) is called a Singular Value Decomposition (SVD) (or Proper Orthogonal Decomposition) of W according to the separation of variables (D_x, D_y). The sequence $(\sigma_k)_{k \in \mathbb{N}^*}$ is known to be unique and is called the sequence of singular values of W associated to the separation of variables (D_x, D_y) of the set D. The orthonormal basis $(U_k)_{k \in \mathbb{N}^*}$ (respectively $(V_k)_{k \in \mathbb{N}^*}$) may not be unique but is called a sequence of left (respectively right) singular vectors of W associated to this partitioning.

Assuming that $\mathcal{N}_x := \#D_x < +\infty$ and $\mathcal{N}_y := \#D_y < +\infty$, the complexity of the computation of the POD decomposition (2.1) scales like

$$\mathcal{O}\left(\max(\mathcal{N}_x, \mathcal{N}_y) \min(\mathcal{N}_x, \mathcal{N}_y)^2 \right). \tag{2.2}$$

2.2. Canonical Polyadic (CP) and Tensor Train (TT) formats. Let $d \in \mathbb{N}^*$, for all $1 \leq i \leq d$. Ω_i are open subsets of $\mathbb{R}^{d_i}$ for some $d_i \in \mathbb{N}^*$, $\Omega := \Omega_1 \times \cdots \times \Omega_d$. Let $F \in L^2(\Omega_1 \times \cdots \times \Omega_d)$, where $\Omega_j \subset \mathbb{R}_{p_j}$ for some $p_j \in \mathbb{N}^*$ for $1 \leq j \leq d$.

The function F is said to belong to the **Canonical Polyadic (CP)** format [14, 18, 8] with rank $r \in \mathbb{N}^*$ if it reads as:

$$F(x_1, x_2, ..., x_d) = \sum_{i=1}^{r} u_i^{(1)}(x_1) u_i^{(2)}(x_2) \cdots u_i^{(d)}(x_d),$$

for some functions $u_i^{(j)} \in L^2(\Omega_j)$ for $1 \leq i \leq r$ and $1 \leq j \leq d$.

The function F is said to belong to the **Tensor Train (TT)** format with ranks $r_1, \dots, r_{d-1} \in \mathbb{N}^*$ [22] if and only if

$$F(x_1, x_2, ..., x_d) = \sum_{i_1=1}^{r_1} ... \sum_{i_{d-1}=1}^{r_{d-1}} u_{i_1}^{(1)}(x_1) u_{i_1,i_2}^{(2)}(x_2) u_{i_2,i_3}^{(3)}(x_3) \cdots u_{i_{d-2},i_{d-1}}^{(d-1)}(x_{d-1}) u_{i_{d-1}}^{(d)}(x_d)$$

with $u_{i_{j-1},i_j}^{(j)} \in L^2(\Omega_j)$ for $1 \leq i_{j-1} \leq r_{j-1}$ and $1 \leq i_j \leq r_j$ for all $1 \leq j \leq d$ (with $r_0 = r_d = 1$).

The main advantage of these decompositions is the low memory cost needed to store them. Indeed, if $\mathcal{N}$ degrees of freedom are used per variable, the storage cost of a general function $F \in L^2(\Omega_1 \times \cdots \times \Omega_d)$ is $\mathcal{O}\left(\mathcal{N}^d\right)$. On the other hand, the storage cost of a CP tensor with rank r reduces to $\mathcal{O}(d\mathcal{N}r)$, which scales linearly in the tensor order d and size $\mathcal{N}$. However, the problem of finding a best approximation of a tensor in CP format may be ill-posed [6] and leads to numerical instabilities. The most classical algorithm in order to compute an approximation of a tensor in the CP format is the so-called Alternating Least Square (ALS) method, which sometimes may be quite slow to converge [2] especially for high-order tensors. Some alternative methods [32, 26] have been proposed in order to obtain more efficient algorithms.

2.3. TT-SVD algorithm. Let now $W \in L^2(\Omega)$. We recall in Algorithm 2.1 the well-known TT-SVD algorithm for computing an approximation of the tensor W with prescribed accuracy $\epsilon > 0$ in a TT format.

3. The Sum of Tensor Trains (SoTT) algorithm.

The aim of this section is to present the Sum of Tensor Trains (SoTT) algorithm we propose in order to greedily construct an approximation of a given tensor as a sum of Tensor Trains (TTs), where the order of the variates and the values of the ranks can be different from one term to another. The algorithm is presented in Section 3.1 and in a more detailed way in Algorithm 3.1. It is proved to converge exponentially fast in finite dimension in Section 3.2. Lastly, a discussion about the complexity of the method is given in Section 3.3.

3.1. Presentation of the SoTT algorithm. In the rest of the article, we denote $\mathcal{S}_d$ the set of permutations of the set $\{1, \cdots, d\}$.

The aim of the SoTT algorithm is to compute, after n iterations, an approximation of the tensor W as a sum of n TTs. At iteration n, the SoTT computes an approximation of W under the form

$$\widetilde{W}^{n-1} + R_1^n(x_{\tau^n(1)}) R_2^n(x_{\tau^n(2)}) \cdots R_d^n(x_{\tau^n(d)}),$$

where $\widetilde{W}^{n-1}$ is the approximation obtained after $n-1$ iterations of the algorithm, where $\tau_n \in \mathcal{S}_d$ is a well-chosen permutation of the variables, and for all $1 \leq j \leq d$, $R_j^n \in L^2\left(\Omega_{\tau^n(j)}, \mathbb{R}^{K_{j-1}^n \times K_j^n}\right)$, where $K_0^n = 1$ and K_d^n. The aim of the n^{th} iteration is to choose the permutation τ_n and the values of the ranks $(K_j^n)_{1 \leq j \leq d-1}$ in an appropriate way, which follows a greedy procedure.

The idea behind the SoTT procedure is the following: the order of the variables is chosen so that it enables to obtain an interesting trade-off between accuracy and memory storage. For instance, $\tau_n(1)$ is chosen as follows. Let us denote by $W^{n-1} :=$

Procedure 2.1 TT-SVD algorithm

Input: $\epsilon > 0$, $W \in L^2(\Omega)$

Output: $K_1, \cdots, K_{d-1} \in \mathbb{N}^*$ TT-ranks, $R_1 \in L^2\left(\Omega_1, \mathbb{R}^{1\times K_1}\right)$, $R_d \in L^2\left(\Omega_d, \mathbb{R}^{K_{d-1}\times 1}\right)$ and for all $i = 2, \cdots, d-1$, $R_i \in L^2\left(\Omega_i, \mathbb{R}^{K_{i-1}\times K_i}\right)$ so that the Tensor Train $\widetilde{W} \in L^2(\Omega)$ defined by

$$\widetilde{W}(x_1, \cdots, x_d) := R_1(x_1)R_2(x_2)\cdots R_d(x_d) \quad \forall (x_1, \cdots, x_d) \in \Omega,$$

satisfies $\|W - \widetilde{W}\|^2_{L^2(\Omega)} \leq \epsilon^2$.

Define $K_0 := 1$, $D_0 = \{1\}$ and define $\overline{W}_0 \in L^2(D_0 \times \Omega)$ such that $\overline{W}_0(1,\cdot) = W$, $\mathcal{I}_0 := \{1, \cdots, d\}$, $\widehat{\Omega}_0 := \Omega$.

for $j = 1, \cdots, d-1$ **do**

Since $D_{j-1}\times\widehat{\Omega}_{j-1} = (D_{j-1}\times\Omega_j)\times\widehat{\Omega}_j$ with $\widehat{\Omega}_j = \Omega_{j+1}\times\cdots\times\Omega_d$, compute the SVD decomposition of $\overline{W}_{j-1}$ according to the separation of variables $(D_{j-1}\times\Omega_j, \widehat{\Omega}_j)$ so that

$$\overline{W}_{j-1} = \sum_{k\in\mathbb{N}^*} \sigma_{j,k} U_{j,k} \otimes V_{j,k}.$$

Select $K_j \in \mathbb{N}^*$ such that $K_j = \inf\left\{K \in \mathbb{N}^*, \quad \sum_{k\geq K} |\sigma_{j,k}|^2 \leq \frac{\epsilon^2}{d-1}\right\}.$

Define $D_j := \{1, \cdots, K_j\}$ and define $\overline{W}_j \in L^2\left(D_j \times \widehat{\Omega}_j\right)$ by

$$\overline{W}_j(k_j, y_j) = \sigma_{j,k_j} V_{j,k_j}(y_j)$$

for all $(k_j, y_j) \in D_j \times \widehat{\Omega}_j$.

Define $R_j \in L^2\left(\Omega_j, \mathbb{R}^{K_{j-1}\times K_j}\right)$ as

$$R_j(x_j) = \left(U_{j,k_j}(k_{j-1}, x_j)\right)_{\substack{1\leq k_{j-1}\leq K_{j-1}\\ 1\leq k_j\leq K_j}}$$

for all $x_j \in \Omega_j$.

end for

Define $R_d \in L^2\left(\Omega_d, \mathbb{R}^{K_{d-1}\times 1}\right)$ by

$$R_d(x_d) = \left(\sigma_{d-1,k_{d-1}} V_{d-1,k_{d-1}}(x_d)\right)_{1\leq k_{d-1}\leq K_{d-1}}.$$

$W - \widetilde{W}^{n-1}$ and let us denote $\overline{W}_0^n := W^{n-1}$. The POD decomposition of $\overline{W}_0^n$ is computed with respect to all the partitioning of the variables of the form $\Omega = \Omega_i \times (\times_{1\leq j\neq i\leq d}\Omega_j)$ for all $i \in \{1, \cdots, d\} = \mathcal{I}_0^n$. Denoting by $\left(\sigma_{1,k}^{i,n}\right)_{k\in\mathbb{N}^*}$ the sequence of singular values associated to the i^{th} partitioning of the variables, for all $r \in \mathbb{N}^*$, one can compute

$$L_{i,1}^n(r) = \sum_{k=1}^r |\sigma_{1,k}^{i,n}|^2 - \beta_{i,1}^n r$$

where $\beta_{i,1}^n$ is a positive scalar which definition is discussed below. The function $L_{i,1}$ is defined so that it reads as the sum of two terms: on one hand, $\sum_{k=1}^r |\sigma_{1,k}^{i,n}|^2$ gives the ℓ^2 norm of the rank-r truncated POD of $\overline{W}_0^n$ and increases with r; on the other hand, $\beta_{i,1}^n r$ is a term which reflects the memory need related to the storage of a rank-r truncated POD of $\overline{W}_0^n$. The maximizer $r_{i,1}^n \in \mathbb{N}^*$ so that

$$r_{i,1}^n \in \operatorname*{argmax}_{r\in\mathbb{N}^*} L_{i,1}^n(r)$$

Procedure 3.1 SoTT algorithm

Input: $\epsilon > 0$, $W \in L^2(\Omega)$

Output: $N \in \mathbb{N}^*$, for all $1 \leq n \leq N$, $\tau^n \in \mathcal{S}_d$, $K_1^n, \cdots, K_{d-1}^n \in \mathbb{N}^*$ TT-ranks, $R_1^n \in L^2\left(\Omega_{\tau^n(1)}, \mathbb{R}^{1\times K_1^n}\right)$, $R_d^n \in L^2\left(\Omega_{\tau^n(d)}, \mathbb{R}^{K_{d-1}^n \times 1}\right)$ and for all $i = 2, \cdots, d-1$, $R_i^n \in L^2\left(\Omega_{\tau^n(i)}, \mathbb{R}^{K_{i-1}^n \times K_i^n}\right)$ so that the sum of Tensor Trains $\widetilde{W} \in L^2(\Omega)$ defined by

$$\widetilde{W}(x_1, \cdots, x_d) := \sum_{n=1}^{N} R_1^n(x_{\tau^n(1)}) R_2^n(x_{\tau^n(2)}) \cdots R_d^n(x_{\tau^n(d)}) \quad \forall (x_1, \cdots, x_d) \in \Omega,$$

satisfies $\|W - \widetilde{W}\|^2_{L^2(\Omega)} \leq \epsilon^2$.

Set $W^0 = W$, $n = 1$.

while $\|W^{n-1}\|^2_{L^2(\Omega)} > \epsilon$ **do**

  Define $K_0^n := 1$, $D_0^n = \{1\}$ and define $\overline{W}_0^n \in L^2(D_0 \times \Omega)$ such that $\overline{W}_0^n(1, \cdot) = W^{n-1}$, $\mathcal{I}_0^n := \{1, \cdots, d\}$.

  for $j = 1, \cdots, d-1$ **do**

    For all $i \in \mathcal{I}_{j-1}^n$, since $D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i = \left(D_{j-1}^n \times \Omega_i\right) \times \bigtimes_{i' \in \mathcal{I}_{j-1}^n \setminus \{i\}} \Omega_{i'}$, compute the SVD decomposition of $\overline{W}_{j-1}^n$ according to the separation of variables $(D_{j-1}^n \times \Omega_i, \bigtimes_{i' \in \mathcal{I}_{j-1}^n \setminus \{i'\}} \Omega_{i'})$ so that

$$\overline{W}_{j-1}^n = \sum_{k \in \mathbb{N}^*} \sigma_{j,k}^{i,n} U_{j,k}^{i,n} \otimes V_{j,k}^{i,n}.$$

    Select $i_j^n \in \mathcal{I}_{j-1}^n$ and $\overline{K}_j^n \in \mathbb{N}^*$ so that

$$\left(i_j^n, \overline{K}_j^n\right) \in \underset{i \in \mathcal{I}_{n-1}, r \in \mathbb{N}^*}{\operatorname{argmax}} \sum_{k=1}^{r} |\sigma_{j,k}^{i,n}|^2 - \beta_{i,j}^n r,$$

    where for all $i \in \mathcal{I}_{j-1}^n$, $\beta_{i,j}^n > 0$ is chosen according to (3.1).

    Define $\tau^n(j) = i_j^n$.

    Select $K_j^n \in \mathbb{N}^*$ such that $K_j^n = \min\left(\overline{K}_j^n, \inf\left\{K \in \mathbb{N}^*, \quad \sum_{k \geq K} \left|\sigma_{j,k}^{\tau^n(j),n}\right|^2 \leq \frac{\epsilon^2}{d-1}\right\}\right)$.

    Define $\mathcal{I}_j^n := \mathcal{I}_{j-1}^n \setminus \{\tau^n(j)\}$ so that $\#I_j^n = d - j$.

    Define $D_j^n := \left\{1, \cdots, K_j^n\right\}$ and define $\overline{W}_j^n \in L^2\left(D_j^n \times \bigtimes_{i \in \mathcal{I}_j^n} \Omega_i\right)$ by

$$\overline{W}_j^n(k_j, y_{\tau^n(j)}) = \sigma_{j,k_j}^{\tau^n(j),n} V_{j,k_j}^{\tau^n(j),n}(y_{\tau^n(j)})$$

    for all $(k_j, y_{\tau^n(j)}) \in D_j^n \times \bigtimes_{i \in \mathcal{I}_j^n} \Omega_i$.

    Define $R_j^n \in L^2\left(\Omega_{\tau_n(j)}, \mathbb{R}^{K_{j-1}^n \times K_j^n}\right)$ as

$$R_j^n(x_{\tau^n(j)}) = \left(U_{j,k_j}^{\tau_n(j),n}(k_{j-1}, x_{\tau^n(j)})\right)_{1 \leq k_j \leq K_j^n, 1 \leq k_{j-1} \leq K_{j-1}^n}$$

    for all $x_{\tau^n(j)} \in \Omega_{\tau^n(j)}$.

  end for

  Since $\#I_{d-1}^n = 1$, let $i_d^n \in \{1, \cdots, n\}$ such that $I_{d-1}^n = \{i_d^n\}$. Define $\tau^n(d) = i_d^n$.

  Define $R_d^n \in L^2\left(\Omega_{\tau^n(d)}, \mathbb{R}^{K_{d-1}^n \times 1}\right)$ by

$$R_d^n(x_{\tau^n(d)}) = \left(\sigma_{d-1,k_{d-1}}^{\tau^n(d-1),n} V_{d-1,k_{d-1}}^{\tau^n(d-1),n}(x_{\tau^n(d)})\right)_{1 \leq k_{d-1} \leq K_{d-1}^n}.$$

  Compute $W^n(x_1, \cdots, x_d) = W^{n-1}(x_1, \cdots, x_d) - R_1^n(x_{\tau^n(1)}) R_2^n(x_{\tau^n(2)}) \cdots R_d^n(x_{\tau^n(d)})$ for all $(x_1, \cdots, x_d) \in \Omega$.

  $n = n + 1$

end while

$N = n - 1$

is a value of rank which enables to obtain a reasonable trade-off between the accuracy of the truncated POD and its memory storage. Then, $\tau_n(1)$ is chosen as the optimum index $i \in \mathcal{I}_0^n$ such that

$$\tau_n(1) = \underset{i\in\mathcal{I}_0^n}{\operatorname{argmax}}\, L_{i,1}^n(r_{i,1}^n) = \underset{i\in\mathcal{I}_0^n}{\operatorname{argmax}} \sup_{r\in\mathbb{N}^*} L_{i,1}^n(r),$$

and gives the index of the first variable in the TT computed at the n^{th} iteration of the SoTT. A preliminary value of the rank $\overline{K}_1^n$ is then chosen so that $\overline{K}_1^n = r_{\tau_n(1),1}^n$.

An additional step is used at line 9 for the definition of the final value of the rank K_1^n which ensures that if

$$\sum_{k\geq \overline{K}_1^n} \left|\sigma_{1,k}^{\tau^n(1),n}\right|^2 \leq \frac{\epsilon^2}{d-1},$$

then K_1^n is the lowest possible rank which guarantees that

$$\sum_{k\geq K_1^n} \left|\sigma_{1,k}^{\tau^n(1),n}\right|^2 \leq \frac{\epsilon^2}{d-1}.$$

To select the values $\tau_n(2), \cdots, \tau_n(d)$ in order to choose the complete order of the variables entering the definition of the n^{th} TT, one uses a similar iterative procedure applied to the $d-1$-order tensor $\overline{W}_1^n$ which reads as the projection of the tensor $\overline{W}_0^n$ onto the K_1^n first POD modes obtained from the $\tau_n(1)^{th}$ partitioning of variables.

We observe that the choice of the values of $\beta_{i,j}^n > 0$ at Step 8 of the SoTT algorithm is critical for its efficiency. In practice, in the case where for all $1 \leq i \leq d$, $\#\Omega_i = \mathcal{N}_i < +\infty$ (i.e. when the tensor is defined on a discrete domain), we make the following choice:

$$\beta_{i,j}^n = \frac{\mathcal{N}_i + \Pi_{i'\in\mathcal{I}_j^n\setminus\{i\}}\mathcal{N}_{i'}}{\Pi_{i'\in\mathcal{I}_j^n}\mathcal{N}_{i'}} \sum_{k=1}^{+\infty} |\sigma_{j,k}^{i,n}|^2 = \frac{\mathcal{N}_i + \Pi_{i'\in\mathcal{I}_j^n\setminus\{i\}}\mathcal{N}_{i'}}{\Pi_{i'\in\mathcal{I}_j^n}\mathcal{N}_{i'}} \|\overline{W}_{j-1}^n\|_{\ell^2}^2. \tag{3.1}$$

Let us point out that the function $\mathbb{N} \ni r \mapsto L_{i,j}^n(r) := \sum_{k=1}^r |\sigma_{j,k}^{i,n}|^2 - \beta_{i,j}^n r$ is concave. Then, it holds that $L_{i,j}^n(0) = 0$, and there exists at least one $r_{i,j}^n \in \left\{0, \cdots, \min\left(\mathcal{N}_i, \Pi_{i'\in\mathcal{I}_j^n\setminus\{i\}}\mathcal{N}_{i'}\right)\right\}$ so that

$$r_{i,j}^n \in \underset{r\in\left\{0,\cdots,\min\left(\mathcal{N}_i,\Pi_{i'\in\mathcal{I}_j^n\setminus\{i\}}\mathcal{N}_{i'}\right)\right\}}{\operatorname{argmax}} L_{i,j}^n(r),$$

and

$$L_{i,j}^n(r_{i,j}^n) \geq 0.$$

3.2. Exponential convergence of the SoTT algorithm in finite dimension. The aim of this section is to prove that the SoTT algorithm converges exponentially fast with the number of iterations in finite dimension.

PROPOSITION 3.1. *Let us assume that there exists for all* $1 \leq i \leq d$, *a* $\#\Omega_i < +\infty$. *Then, there exists* $0 < \alpha < 1$ *such that for all* $n \in \mathbb{N}^*$,

$$\|W^n\|_{L^2(\Omega)}^2 \leq \alpha^n \|W\|_{L^2(\Omega)}^2, \tag{3.2}$$

with

$$\alpha \leq 1 - \frac{1}{\mathcal{N}^{2(\lceil d/2\rceil!)}},$$

where $\mathcal{N} := \max_{1\leq i\leq d} \#\Omega_i$.

We observe in practice that the upper bound on the convergence rate of the SoTT algorithm given by Proposition 3.1 is very pessimistic. We refer the reader to Section 5 for numerical results which illustrate this fact.

Proof. Let us begin by proving that for all $n \in \mathbb{N}^*$,

$$\|W^n\|_{L^2}^2 \leq \left(1 - \frac{1}{\mathcal{N}^{2(\lceil d/2 \rceil !)}}\right) \|W^{n-1}\|_{L^2}^2 . \tag{3.3}$$

Indeed, for all $n \in \mathbb{N}^*$, let us denote

$$U^n(x_1, \cdots, x_d) := R_1^n(x_{\tau^n(1)}) R_2^n(x_{\tau^n(2)}) \cdots R_d^n(x_{\tau^n(d)}), \quad \forall (x_1, \cdots, x_d) \in \Omega_1 \times \cdots \times \Omega_d.$$

Note that, by construction and definition of the SoTT algorithm, $\left\langle W^{n-1} - U^n, U^n \right\rangle_{L^2(\Omega)} = 0$, so that

$$\|W^n\|_{L^2}^2 = \left\|W^{n-1} - U^n\right\|_{L^2}^2 = \left\|W^{n-1}\right\|_{L^2}^2 - \|U^n\|_{L^2}^2 . \tag{3.4}$$

By definition of the algorithm, it holds that for all $1 \leq j \leq d-1$,

$$K_j^n \leq \min\left(\mathcal{N} K_{j-1}^n, \mathcal{N}^{d-j}\right),$$

where $K_0^n = 1$. Thus, by induction, we obtain that for all $1 \leq j \leq d-1$,

$$K_j^n \leq \min\left(\mathcal{N}^j, \mathcal{N}^{d-j}\right).$$

As a consequence, for all $n \in \mathbb{N}^*$, and all $1 \leq j \leq d-1$, we obtain that for all $i \in \mathcal{I}_{j-1}^n$,

$$\# D_{j-1}^n \times \Omega_i \leq \mathcal{N} \min\left(\mathcal{N}^{j-1}, \mathcal{N}^{d+1-j}\right) = \min\left(\mathcal{N}^j, \mathcal{N}^{d+2-j}\right) \qquad \text{and } \# \bigtimes_{i' \in \mathcal{I}_{j-1}^n \setminus \{i\}} \Omega_{i'} \leq \mathcal{N}^{d-j}.$$

Thus, for all $i \in \mathcal{I}_{j-1}^n$,

$$\min\left(\# D_{j-1}^n \times \Omega_i, \# \bigtimes_{i' \in \mathcal{I}_{j-1}^n \setminus \{i\}} \Omega_{i'}\right) \leq \min\left(\mathcal{N}^j, \mathcal{N}^{d-j}\right). \tag{3.5}$$

As a consequence, for all $i \in \mathcal{I}_{j-1}^n$, it holds that

$$\left|\sigma_{j,1}^{i,n}\right|^2 \geq \frac{\left\|\overline{W}_{j-1}^n\right\|_{L^2}^2}{\min\left(\mathcal{N}^j, \mathcal{N}^{d-j}\right)}.$$

Moreover, denoting by $\widehat{W}_j^n := \sum_{k_j=1}^{K_j^n} \sigma_{j,k_j}^{\tau^n(j),n} U_{j,k_j}^{\tau^n(j),n} \otimes V_{j,k_j}^{\tau^n(j),n}$, it holds that

$$\left\|\overline{W}_{j-1}^n - \widehat{W}_j^n\right\|_{L^2\left(D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i\right)}^2 \leq \left|\sigma_{j,1}^{i,n}\right|^2 .$$

Thus, using the fact that $\overline{W}_{j-1}^n - \widehat{W}_j^n$ is orthogonal to $\widehat{W}_j^n$, we obtain that

$$\begin{aligned}
\left\|\widehat{W}_j^n\right\|_{L^2\left(D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i\right)}^2 &= \left\|\overline{W}_{j-1}^n\right\|_{L^2\left(D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i\right)}^2 - \left\|\overline{W}_{j-1}^n - \widehat{W}_j^n\right\|_{L^2\left(D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i\right)}^2 \\
&\geq \left\|\overline{W}_{j-1}^n\right\|_{L^2\left(D_{j-1}^n \times \bigtimes_{i \in \mathcal{I}_{j-1}^n} \Omega_i\right)}^2 \left(1 - \frac{1}{\min\left(\mathcal{N}^j, \mathcal{N}^{d-j}\right)}\right).
\end{aligned}$$

Lastly, using the fact that $\left(U_{j,k_j}^{\tau^n(j),n}\right)_{1\le k_j\le K_n^j}$ is an orthonormal family of $L^2\left(\Omega_{\tau^n(j)}\right)$ and $\left(V_{j,k_j}^{\tau^n(j),n}\right)_{1\le k_j\le K_n^j}$ is an orthonormal family of $L^2\left(\bigtimes_{i\in\mathcal{I}_j^n}\Omega_i\right)$, it holds that

$$\left\|\widehat{W}_j^n\right\|^2_{L^2\left(D_{j-1}^n\times\bigtimes_{i\in\mathcal{I}_{j-1}^n}\Omega_i\right)} = \left\|\overline{W}_j^n\right\|^2_{L^2\left(D_j^n\times\bigtimes_{i\in\mathcal{I}_j^n}\Omega_i\right)} = \sum_{k_j=1}^{K_j^n}\left|\sigma_{j,k_j}^{\tau^n(j),n}\right|^2.$$

As a consequence, we obtain that for all $n\in\mathbb{N}^*$ and for all $1\le j\le d-1$,

$$\left\|\overline{W}_j^n\right\|^2_{L^2\left(D_j^n\times\bigtimes_{i\in\mathcal{I}_j^n}\Omega_i\right)} \ge \left\|\overline{W}_{j-1}^n\right\|^2_{L^2\left(D_{j-1}^n\times\bigtimes_{i\in\mathcal{I}_{j-1}^n}\Omega_i\right)}.$$

In addition, it can easily be checked that $\|U^n\|_{L^2(\Omega)^2} = \left\|\overline{W}_{d-1}^n\right\|^2_{L^2\left(D_{d-1}^n\times\Omega_{\tau_n(d)}\right)}$. Thus, by induction over $1\le j\le d-1$, we obtain that for all $n\in\mathbb{N}^*$,

$$\begin{aligned}\|U^n\|_{L^2(\Omega)^2} &\ge \left\|\overline{W}_0^n\right\|^2_{L^2(\Omega)}\frac{1}{\Pi_{1\le j\le d-1}\min(\mathcal{N}^j,\mathcal{N}^{d-j})}\\ &= \left\|W^{n-1}\right\|^2_{L^2(\Omega)}\frac{1}{\Pi_{1\le j\le d-1}\min(\mathcal{N}^j,\mathcal{N}^{d-j})}\\ &\ge \left\|W^{n-1}\right\|^2_{L^2(\Omega)}\frac{1}{\mathcal{N}^{2(\lceil d/2\rceil !)}}.\end{aligned}$$

Collecting this estimate with (3.4), we obtain (3.3). Thus, by induction, we easily obtain the desired result (3.2). □

3.3. Complexity estimate of the SoTT algorithm. The aim of this section is to provide some estimates on the complexity of the computational cost of the SoTT algorithm. Let us assume here that there exists $\mathcal{N}\in\mathbb{N}^*$ such that $\#\Omega_i\le\mathcal{N}$ for all $1\le i\le d$.

We detail the computational cost of each iteration $n\in\mathbb{N}^*$. The computational cost is concentrated in the computation of the different POD decompositions of the tensor $\overline{W}_{j-1}^n$ for each $1\le j\le d-1$ (l.7 of SoTT algorithm), which can be estimated using (2.2). We consider two different cases.

3.3.1. Case 1: Unbounded ranks. Let us begin by giving a very pessimistic bound in the case where no upper bound on the ranks K_j^n is imposed for all $1\le j\le d$. From (3.5), it holds that the computational cost of each POD decomposition scales like

$$\mathcal{O}\left(\max\left(\mathcal{N}^j,\mathcal{N}^{d-j}\right)\min\left(\mathcal{N}^j,\mathcal{N}^{d-j}\right)^2\right).$$

Thus, the total computational cost of the POD decompositions of one iteration of the SoTT algorithm is of the order of

$$\mathcal{O}\left(\sum_{j=1}^{d-1}(d-j+1)\max\left(\mathcal{N}^j,\mathcal{N}^{d-j}\right)\min\left(\mathcal{N}^j,\mathcal{N}^{d-j}\right)^2\right)\approx\mathcal{O}\left(d^2\mathcal{N}^{3\lceil d/2\rceil}\right).$$

3.3.2. Case 2: Bounded ranks. Now, let us assume that there exists $R \in \mathbb{N}^*$ such that $R < \mathcal{N}$ and such that for all $1 \leq j \leq d-1$, $K_j^n \leq R$. Then, for $j = 1$, the computational cost of each POD decomposition scales like

$$\mathcal{N}^{d+1}.$$

Besides, for $2 \leq j \leq d-2$, the computational cost of each POD decomposition scales like

$$R^2 \mathcal{N}^{d-j+1}.$$

Lastly, for $j = d-1$, there is only one POD decomposition to compute, the cost of which scales like

$$R\mathcal{N}^3.$$

Thus, the total cost of the POD decompositions of one SoTT iteration scales like

$$\mathcal{O}\left(d\mathcal{N}^{d+1} + \sum_{j=2}^{d-2} (d-j+1) R^2 \mathcal{N}^{d-j+1} + R\mathcal{N}^3 \right) \approx \mathcal{O}\left(d\mathcal{N}^{d+1} + dR^2\mathcal{N}^{d-1} + R\mathcal{N}^3 \right).$$

Remark. The computational cost of the SoTT algorithm is in general larger than the computational cost of the TT-SVD, as we do not fix a priori the order of the variables. In the first step of the SoTT iteration, the cost is similar to the one of the HOSVD method, in which we compute the POD for all the unfoldings. There is, however, a difference in terms of the memory used. In the practical implementation of SoTT, we do not need to store in memory the POD decomposition of all the unfolding, just the best one. In addition, we do not need to store the potentially dense core tensor. In the case in which we particularize SoTT by fixing the rank (an example to rank 1 is proposed in the forthcoming section), the computational cost of the POD can be reduced.

4. CP-TT: fixed-rank SoTT algorithm with rank 1.

We make here a focus on a particular variant of the SoTT algorithm where all the ranks K_j^n are a priori chosen to be fixed and equal to 1 for all $1 \leq j \leq d$ and all iterations $n \in \mathbb{N}^*$. We refer the reader to [35] for a review on the stability properties of rank-1 tensor decompositions. As an output, the SoTT algorithm then computes an approximation of the tensor W in a CP format and we call it hereafter the CP-TT algorithm. More precisely, for all $1 \leq j \leq d$ and $n \in \mathbb{N}^*$ and where Step 8 of the algorithm is replaced by the following step: select $i_j^n \in \mathcal{I}_{j-1}^n$ so that

$$i_j^n \in \underset{i \in \mathcal{I}_{n-1}}{\operatorname{argmax}} |\sigma_{j,1}^{i,n}|^2$$

and where Step 10 is not performed.

As an output, after n iterations of the CP-TT algorithm, the method greedily produces an approximation of the tensor W under the CP format

$$W \approx \sum_{k=1}^{n} R_1^k(x_{\tau_k(1)}) \cdots R_d^k(x_{\tau_k(d)})$$

where for all $1 \leq k \leq n$ and all $1 \leq i \leq d$, $R_i^k \in L^2(\Omega_{\tau_k(i)})$.

We make a specific focus on this particular case because we numerically observed that this algorithm possesses interesting stability and approximation properties in comparison to other more classical numerical methods like ALS or ASVD for the computation of a CP approximation of a tensor, especially when the order of the tensor d is high. For the sake of comparison, we recall the ALS and ASVD algorithm in Algorithm 4.1 and Algorithm 4.2 respectively. The convergence properties of the ALS algorithm have been abundantly studied. We refer the reader for more details to the following series of works [31, 28, 9, 33, 24]. The ASVD method is proposed in [10].

Procedure 4.1 ALS algorithm

Input: $\epsilon > 0$, $W \in L^2(\Omega)$

Output: $N \in \mathbb{N}^*$, for all $1 \leq n \leq N$ and all $1 \leq i \leq d$, $R_i^n \in L^2(\Omega_i)$ so that the CP tensor $\widetilde{W} \in L^2(\Omega)$ defined by

$$\widetilde{W}(x_1, \cdots, x_d) := \sum_{n=1}^{N} R_1^n(x_1) R_2^n(x_2) \cdots R_d^n(x_d) \quad \forall (x_1, \cdots, x_d) \in \Omega,$$

satisfies $\|W - \widetilde{W}\|^2_{L^2(\Omega)} \leq \epsilon$.

Set $W^0 = W$, $n = 1$.

while $\|W^{n-1}\|^2_{L^2(\Omega)} > \epsilon$ **do**

  For all $1 \leq i \leq d$, select randomly $R_i^{n,0} \in L^2(\Omega_i)$ and set $\eta := \epsilon$ and $m = 1$.

  while $\eta > \frac{1}{10}\epsilon$ **do**

    for $j = 1, \cdots, d$ **do**

      Compute $R_j^{n,m} \in L^2(\Omega_j)$ solution to

$$R_j^{n,m} \in \underset{R_j \in L^2(\Omega_j)}{\operatorname{argmin}} \left\| W_{n-1} - R_1^{n,m} \otimes \cdots \otimes R_{j-1}^{n,m} \otimes R_j \otimes R_{j+1}^{n,m-1} \otimes \cdots \otimes R_d^{n,m-1} \right\|^2_{L^2(\Omega)}$$

    end for

    Compute $\eta := \left\| R_1^{n,m} \otimes \cdots \otimes R_d^{n,m} - R_1^{n,m-1} \otimes \cdots \otimes R_d^{n,m-1} \right\|^2_{L^2}$. Set $m := m + 1$.

  end while

  Define $R_i^n = R_i^{n,m-1}$ for all $1 \leq i \leq d$.

  Compute $W_n(x_1, \cdots, x_d) = W_{n-1}(x_1, \cdots, x_d) - R_1^n(x_1) R_2^n(x_2) \cdots R_d^n(x_d)$ for all $(x_1, \cdots, x_d) \in \Omega$.

  $n = n + 1$

end while

$N = n - 1$

For the presentation of the ASVD algorithm, we need to introduce some additional notation. We denote $\mathcal{J} := \{\{i, j\}, \ 1 \leq i < j \leq d\}$ be the set of all possible pairs of indices between 1 and d. An ordering of the elements of $\mathcal{J}$ is chosen so that

$$\mathcal{J} = (J_l)_{1 \leq l \leq L}$$

where $L = |\mathcal{J}|$.

The closest method to CP-TT we found in the literature is the so called TTr1, proposed in [1]. In this, we apply the TT-SVD method and we fix the rank to one, at every stage of the algorithm. By proceeding this way, we compute all the possible rank-1 terms, and we order them according to the singular values of the POD decompositions. The orthogonality properties make it possible to truncate the so obtained CP decomposition in order to fullfil a prescribed accuracy. The main differences with respect to CP-TT are: first, in CP-TT we do not fix a priori the order of the variables whereas we fix it in TTr1. Second, in CP-TT we proceed in a greedy way, by computing one term at a time, whereas in TTr1 we compute all the terms and then we truncate. The CP-TT has, henceforth, a computational cost per term which is larger, but needs globally less storage, which seems beneficial to compress higher order tensors.

5. Numerical Experiments. In this section, several numerical experiments are proposed. In the first part, we compare several rank-1 update methods (ALS, ASVD,

Procedure 4.2 ASVD algorithm

Input: $\epsilon > 0$, $W \in L^2(\Omega)$

Output: $N \in \mathbb{N}^*$, for all $1 \leq n \leq N$ and all $1 \leq i \leq d$, $R_i^n \in L^2(\Omega_i)$ so that the CP tensor $\widetilde{W} \in L^2(\Omega)$ defined by

$$\widetilde{W}(x_1, \cdots, x_d) := \sum_{n=1}^{N} R_1^n(x_1 R_2^n(x_2) \cdots R_d^n(x_d) \quad \forall (x_1, \cdots, x_d) \in \Omega,$$

satisfies $\|W - \widetilde{W}\|^2_{L^2(\Omega)} \leq \epsilon$.

Set $W^0 = W$, $n = 1$.

while $\|W^{n-1}\|^2_{L^2(\Omega)} > \epsilon$ **do**

  For all $1 \leq i \leq d$, select randomly $R_i^{n,0} \in L^2(\Omega_i)$ and set $\eta := \epsilon$ and $m = 1$.

  while $\eta > \frac{1}{10}\epsilon$ **do**

    Set $R_i^{n,m} = R_i^{n,m-1}$ for all $1 \leq i \leq d$

    for $l = 1, \cdots, L$ **do**

      Let $1 \leq i_l < j_l \leq d$ so that $J_l = (i_l, j_l)$.

      Compute $U_l^{n,m} \in L^2(\Omega_{i_l} \times \Omega_{j_l})$ solution to

$$U_l^{n,m} \in \underset{U_l \in L^2(\Omega_{i_l} \times \Omega_{j_l})}{\operatorname{argmin}} \left\| W_{n-1} - U_l \otimes \bigotimes_{i \in \{1, \cdots, d\} \setminus J_l} R_i^{n,m} \right\|^2_{L^2(\Omega)}.$$

      Compute $\left(R_{i_l}^{n,m}, R_{j_l}^{n,m}\right) \in L^2(\Omega_{i_l}) \times L^2(\Omega_{j_l})$ solution to

$$\left(R_{i_l}^{n,m}, R_{j_l}^{n,m}\right) \in \underset{\left(R_{i_l}, R_{j_l}\right) \in L^2(\Omega_{i_l}) \times L^2(\Omega_{j_l})}{\operatorname{argmin}} \|U_l^{n,m} - R_{i_l} \otimes R_{j_l}\|^2_{L^2(\Omega_{i_l} \times \Omega_{j_l})}.$$

    end for

    Compute $\eta := \left\| R_1^{n,m} \otimes \cdots \otimes R_d^{n,m} - R_1^{n,m-1} \otimes \cdots \otimes R_d^{n,m-1} \right\|^2_{L^2}$.

  end while

  Define $R_i^n = R_i^{n,m}$ for all $1 \leq i \leq d$.

  Compute $W_n(x_1, \cdots, x_d) = W_{n-1}(x_1, \cdots, x_d) - R_1^n(x_1) R_2^n(x_2) \cdots R_d^n(x_d)$ for all $(x_1, \cdots, x_d) \in \Omega$.

  $n = n + 1$

end while

$N = n - 1$

TTr1 and CP-TT) on random functions belonging to certain classes of regularity. Then, we illustrate the efficiency of the SoTT algorithm with respect to standard TT-SVD algorithms on a particular test case which consists in compressing the solution of a parametric Partial Differential Equation.

5.1. Comparison between CP-TT and other rank-one update methods. The aim of this section is to compare the efficiency of the CP-TT algorithm for the computation of approximations of a tensor in a CP format with ALS, ASVD and TTr1.

Let $(x_1, \ldots, x_d) \in \Omega = [0, 1]^d$. Let $(k_1, ..., k_d) \in \mathbb{N}^d$ be the wave numbers. The function to be compressed is assumed to be given in a Tucker format :

$$W(x_1, \ldots, x_d) = \sum_{k_1=1}^{l_1} \sum_{k_2=1}^{l_2} \ldots \sum_{k_d=1}^{l_d} a_{k_1 \ldots k_d} \sin(\pi k_1 x_1) \times \ldots \times \sin(\pi k_d x_d) \tag{5.1}$$

The values of $l_1, \cdots, l_d \in \mathbb{N}^*$ and of the coefficients $(a_{k_1 \ldots k_d})_{1 \leq k_1 \leq l_1, \cdots, 1 \leq k_d \leq l_d}$ are randomly chosen as follows.

First, the values of $(l_i)_{1 \leq i \leq d}$ are chosen to be a family of independent random integers uniformly distributed between 1 and 6. Second, let $\beta > 0$. Let $(\alpha_{k_1 \ldots k_d})_{1 \leq k_1 \leq l_1, \ldots, 1 \leq k_d \leq l_d}$ be a family of independent random variables uniformly distributed in $[-1, 1]$. For all $1 \leq k_1 \leq l_1$, ..., $1 \leq k_d \leq l_d$, the value $a_{k_1 \dots k_d}$ is then defined as:

$$a_{k_1 \dots k_d} = \frac{\alpha_{k_1 \dots k_d}}{\left(\sqrt{k_1^2 + \dots + k_d^2}\right)^{\beta}}.$$

For different random samples and different values of the coefficient β, we obtain different families of functions W given by (5.1). Let us remark that more detailed rank-r CP approximation of the orthogonal Tucker tensor could be seen in [15], [17].

We are testing how the four methods behave for the compression of 32 different functions generated by the random procedure described above for values of d ranging from 4 to 16. Let us point out that ALS and ASVD are both fixed point based methods, in contrast to CP-TT and TTr1, and the tolerance for the fixed point procedure has been set as $\eta = 1.0 \times 10^{-4}$. The maximum number of iterations of the method $it_{max} = 100$. A uniform discretization grid of Ω with 25 degrees of freedom per direction is used for the discretization of W.

5.1.1. Results for functions with $\beta = \frac{d}{2} + 0.1$. We begin by presenting here some numerical tests obtained with functions generated with $\beta = \frac{d}{2} + 0.1$.

We first present numerical experiments comparing CP-TT, TTr1, ALS and ASVD in cases where $d = 4$ in Figure 1. Note that the memory required by the TTr1 method has prevented us from being able to carry out the method for higher values of d. Hence, for higher values of d, we only compare CP-TT with ALS and ASVD methods in Figure 2 for $d = 12$ and $d = 16$. The mean and standard deviation of the L^2 norm of the difference between the exact function W and its approximation given by any method is plotted as a function of the rank of the approximation.

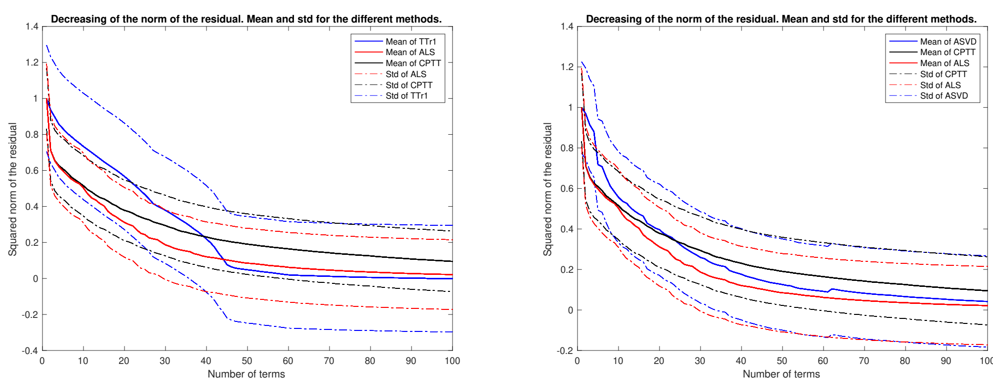
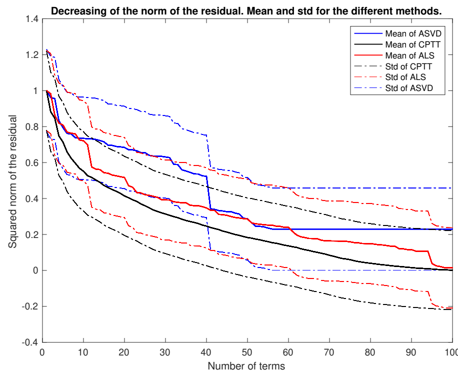


FIG. 1. *Case $d = 4$ and $\beta = \frac{d}{2} + 0.1$. Left: mean and standard deviation of the L^2 norm of the difference between the exact function W and its approximation given by ALS (red), TTr1 (blue) and CP-TT (black) as a function of the number of terms. Right: mean and standard deviation of the L^2 norm of the difference between the exact function W and its approximation given by ALS (red), ASVD (blue) and CP-TT (black) as a function of the number of terms.*

Note that in the case $d = 4$ (Figure 1), ALS outperforms ASVD and CP-TT. The ALS also outperforms TTr1 for small values of the rank. However, in cases where $d = 12$ and $d = 16$ (Figure 2), CP-TT has a better numerical behavior when considering the decay of the norm of the residual with respect to the number of terms. In particular, the compression rate is better on average and the norm decay of the error with respect to the rank of the approximation is less subject to statistical noise.



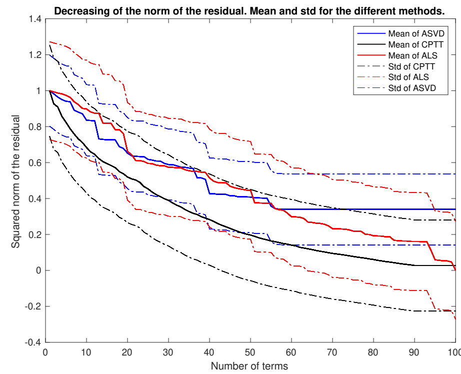


FIG. 2. *Case* $\beta = \frac{d}{2} + 0.1$. *Mean and standard deviation of the* L^2 *norm of the difference between the exact function* W *and its approximation given by ALS (red), ASVD (blue) and CP-TT (black) as a function of the number of terms. Left: case* $d = 12$. *Right: case* $d = 16$.

Table 1 summarizes the numerical results obtained with the CP-TT, ALS and ASVD methods. In particular, the mean and the standard deviation of the error (on the 32 random functions) are reported for ranks equal to $25, 50, 75$ and tensor orders $d = 4, 6, 8, 10, 12, 14, 16$. We see here again that for low-order tensors (here when $d = 4$) ALS has better performances, whereas for higher order tensors CP-TT outperforms the other methods both in terms of mean and standard deviation.

5.1.2. Results for functions with $\beta = \frac{d}{2} + 1.1$**.** Similar numerical tests have been obtained here in the case where the value of the parameter β is chosen to be equal to $\frac{d}{2} + 1.1$.

Figure 3 and Figure 4 are the counterparts of Figure 1 and Figure 2 respectively.

The results obtained on $H^1(\Omega)$ functions are equivalent to the ones shown for $L^2(\Omega)$ functions, showing that the decrease in the error norm with the approximation rank is quite regular in CP-TT and behaves in a quite stable way also for higher order tensors.

Table 2 is the counterpart of Table 1 for $\beta = \frac{d}{2} + 1.1$.

Conclusions on this second test case are similar to the ones obtained in Section 5.1.1. ALS seems to outperform all other rank-1 update methods in the case where $d = 4$, whereas CP-TT seems to outperform the other methods for higer values of d.

5.1.3. Comparison of the norm of the residual with respect to computational time. It is clear that one iteration of CP-TT is in general more costly in terms of computational time than one ALS iteration. As a consequence, even if the norm of the residual given by the CP-TT algorithm seems to decrease faster as a function of the number of terms in the approximation than with any other rank-1 update methods for high values of d, it is legitimate to compare the norm of the residual given by any method with respect to the computational time needed to compute the corresponding approximations.

This is the aim of Figure 5, where the norm of the residual is plotted for CP-TT, ALS and ASVD as a function of the computational time for different values of β

		Mean			Std		
Dimension (d)	Rank (r)	ALS	CPTT	ASVD	ALS	CPTT	ASVD
4	25	**0.2942**	0.3826	0.3118	0.0702	0.0850	0.0843
	50	**0.1082**	0.2433	0.1257	0.0326	0.0568	0.0664
	75	**0.0508**	0.1681	0.0689	0.0180	0.0408	0.0666
6	25	0.4479	**0.3771**	0.4806	0.1099	0.0826	0.1074
	50	0.2705	**0.1982**	0.2883	0.0752	0.0485	0.0675
	75	0.1232	**0.0806**	0.1369	0.0325	0.0252	0.0368
8	25	0.5341	**0.3707**	0.5532	0.1183	0.0592	0.1238
	50	0.3060	**0.1909**	0.3415	0.0722	0.0341	0.0932
	75	0.1592	**0.0682**	0.1807	0.0435	0.0160	0.0625
10	25	0.5023	**0.3598**	0.5451	0.0879	0.0643	0.1055
	50	0.3191	**0.1826**	0.3797	0.0643	0.0342	0.0774
	75	0.1714	**0.0655**	0.2792	0.0453	0.0162	0.1265
12	25	0.5170	**0.3246**	0.5639	0.1117	0.0576	0.1250
	50	0.3249	**0.1623**	0.4206	0.0824	0.0286	0.1579
	75	0.1543	**0.0579**	0.3498	0.0369	0.0113	0.2057
14	25	0.4443	**0.2336**	0.4783	0.1712	0.1064	0.1585
	50	0.2407	**0.1004**	0.3307	0.0937	0.0588	0.1737
	75	0.1411	**0.0321**	0.2230	0.0541	0.0235	0.1821
16	25	0.5529	**0.3160**	0.6150	0.1305	0.0818	0.1656
	50	0.3487	**0.1448**	0.4424	0.0849	0.0389	0.1942
	75	0.1946	**0.0616**	0.3678	0.0905	0.0289	0.2354

TABLE 1

Mean and standard deviation of the norm of the residual for 32 random functions in the case where $\beta = \frac{d}{2} + 0.1$.

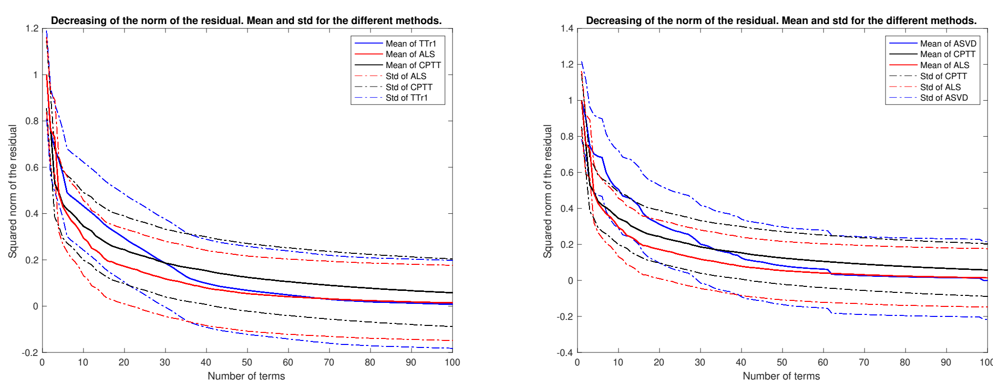


FIG. 3. *Case* $d = 4$ *and* $\beta = \frac{d}{2} + 1.1$. *Left: mean and standard deviation of the* L^2 *norm of the difference between the exact function* W *and its approximation given by ALS (red), TTr1 (blue) and CP-TT (black) as a function of the number of terms. Right: mean and standard deviation of the* L^2 *norm of the difference between the exact function* W *and its approximation given by ALS (red), ASVD (blue) and CP-TT (black) as a function of the number of terms.*

different values of d.

We observe in these tests that, in terms of mean of the decay of the norm of

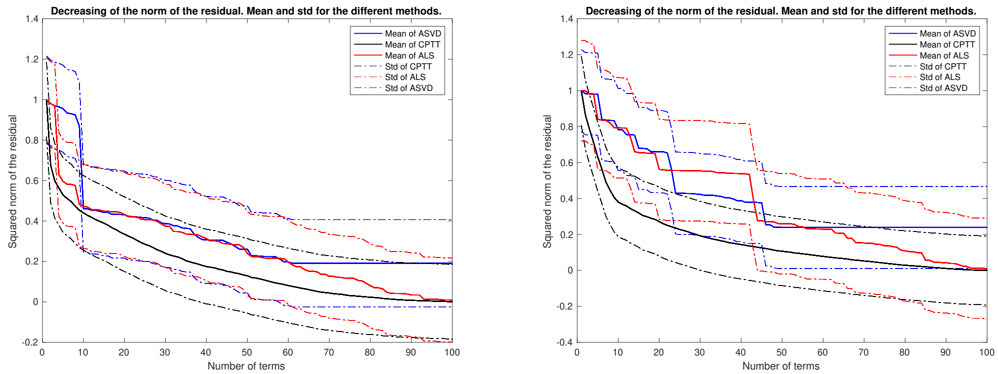


FIG. 4. *Case* $\beta = \frac{d}{2} + 1.1$. *Mean and standard deviation of the* L^2 *norm of the difference between the exact function* W *and its approximation given by ALS (red), ASVD (blue) and CP-TT (black) as a function of the number of terms. Left: case* $d = 12$. *Right: case* $d = 16$.

the residual as a function of the computational time, the three methods perform similarly. However, we observe that CP-TT has a lower stochastic variability than ALS and ASVD.

5.2. SoTT for the compression of the solution of a parametric reaction diffusion equation. The aim of this section is to illustrate the numerical behavior of the SoTT algorithm where the ranks are not fixed a priori but chosen according to Algorithm 3.1.

We consider here a fourth-order a tensor obtained by solving numerically a 1D-1D parametric Fischer-Kolmogorov-Petrovsky-Piskunov (FKPP) equation. Let $\Omega_1 := [0,1]$ be the space domain, and $\Omega_2 := [0, 0.25]$ be the time domain. Let $\alpha \in \Omega_3 := [25, 100]$ be the reaction coefficient, and $\beta \in \Omega_4 := [0.25, 0.75]$ be a parameter defining the initial condition. The equation reads: for all $(\alpha, \beta) \in \Omega_3 \times \Omega_4$, find $u_{\alpha,\beta} : \Omega_1 \times \Omega_2 \ni (x,t) \mapsto u_{\alpha,\beta}(x,t) \in \mathbb{R}$ solution to

$$
\left\{
\begin{array}{lll}
\partial_t u_{\alpha,\beta} & = \partial_x^2 u_{\alpha,\beta} + \alpha u_{\alpha,\beta}(1 - u_{\alpha,\beta}), & \forall (x,t) \in \Omega_1 \times \Omega_2 \\
u_{\alpha,\beta}(0,t) & = u_{\alpha,\beta}(1,t) = 0, & \forall t \in \Omega_2, \\
u_{\alpha,\beta}(x,0) & = \exp(-200(x-\beta)^2), & \forall x \in \Omega_1.
\end{array}
\right. \tag{5.2}
$$

We then define, for all $(x_1, x_2, x_3, x_4) \in \Omega_1 \times \Omega_2 \times \Omega_3 \times \Omega_4$,

$$
W(x_1, x_2, x_3, x_4) := u_{x_3, x_4}(x_1, x_2).
$$

Equation 5.2 is discretized and solved by means of a classical centred finite difference scheme. Examples of the space-time portrait of the solution for different values of the parameters are shown in Fig.6.

We consider uniform discretization grids of Ω_1, Ω_2, Ω_3 and Ω_4 of size $\mathcal{N}_1 = 100$, $\mathcal{N}_2 = 50$, $\mathcal{N}_3 = 10$ and $\mathcal{N}_4 = 10$ respectively.

In Figure 7, the memory of the computed approximation (i.e. the number of stored double precision numbers) is plotted as a function of the residual norm, for the TT-SVD approximations corresponding to all the possible 24 choices of permutations of the variable indices and the SoTT approximation. These approximations are

		Mean			Std		
Dimension (d)	Rank (r)	ALS	CPTT	ASVD	ALS	CPTT	ASVD
4	25	**0.1722**	0.2261	0.1759	0.0643	0.2261	0.1759
	50	**0.0572**	0.1382	0.0590	0.0220	0.1382	0.0232
	75	**0.0252**	0.0948	0.0262	0.0103	0.0948	0.0110
6	25	0.3741	**0.2938**	0.4171	0.1158	0.0942	0.1341
	50	0.2037	**0.1507**	0.2281	0.0655	0.0523	0.0791
	75	0.0851	**0.0579**	0.1045	0.0334	0.0233	0.0493
8	25	0.3676	**0.2560**	0.3977	0.1361	0.0905	0.1517
	50	0.2136	**0.1229**	0.2413	0.0807	0.0451	0.1023
	75	0.1046	**0.0455**	0.1145	0.0437	0.0195	0.0631
10	25	0.4574	**0.3737**	0.4753	0.1235	0.1548	0.1817
	50	**0.2613**	0.3483	0.3193	0.0809	0.1825	0.1648
	75	**0.1168**	0.3332	0.2352	0.0628	0.2034	0.1865
12	25	0.4634	**0.2505**	0.5182	0.1681	0.0842	0.2116
	50	0.2889	**0.1141**	0.3922	0.1421	0.0384	0.2170
	75	0.1278	**0.0382**	0.3144	0.0671	0.0126	0.2502
14	25	0.5943	**0.2169**	0.4386	0.2043	0.1262	0.2014
	50	0.2841	**0.0779**	0.3132	0.1277	0.0686	0.1915
	75	0.1422	**0.0244**	0.2021	0.0814	0.0227	0.2192
16	25	0.4598	**0.2460**	0.5543	0.1496	0.0726	0.1603
	50	0.2861	**0.1108**	0.3936	0.1268	0.0348	0.2022
	75	0.1395	**0.0438**	0.3181	0.0552	0.0153	0.2477

TABLE 2

Mean and standard deviation of the norm of the residual for 32 random functions in the case where $\beta = \frac{d}{2} + 1.1$.

computed in all cases for several residual tolerances, ranging from 10^{-2} to $5 \cdot 10^{-4}$. Remark here that the results obtained by the TT-SVD algorithm heavily depend on the order of the variables chosen. The difference in memory between the best and the worst TT-SVD is roughly one order of magnitude, for all the tolerances tested.

We observe in this test case that the SoTT method produces a sub-optimal compression with respect to the best TT-SVD compression. However, it performs better than the average TT-SVD and in general better than the canonical order $1, 2, 3, 4$. The first term computed is a rank-1 update, for the second term the TT ranks are $[5, 5, 4]$, for the third $[7, 7, 5]$, and in general we observe that the order of the variables change.

In Figure 8, we compare the performance of SoTT with CP-TT, its particularization to rank-1 updates. More precisely, the logarithm of the memory is plotted as a function of the logarithm of the residual norm, for 5 iterations of SoTT and approximately 360 iterations of CP-TT. We observe in this test case that the performance of SoTT is better than the one of CP-TT.

6. Conclusions and perspectives. In the present work, we proposed a method to compress a given tensor as a sum of Tensor Trains (SoTT). Neither the order of the variables nor the ranks are fixed a priori. Instead, they are the result of an optimization step. A particular instance of this method, consisting in fixing the ranks equal to one in all the steps of the algorithm, produces a CP approximation of a given tensor. A proof of convergence is proposed in the general case of the SoTT

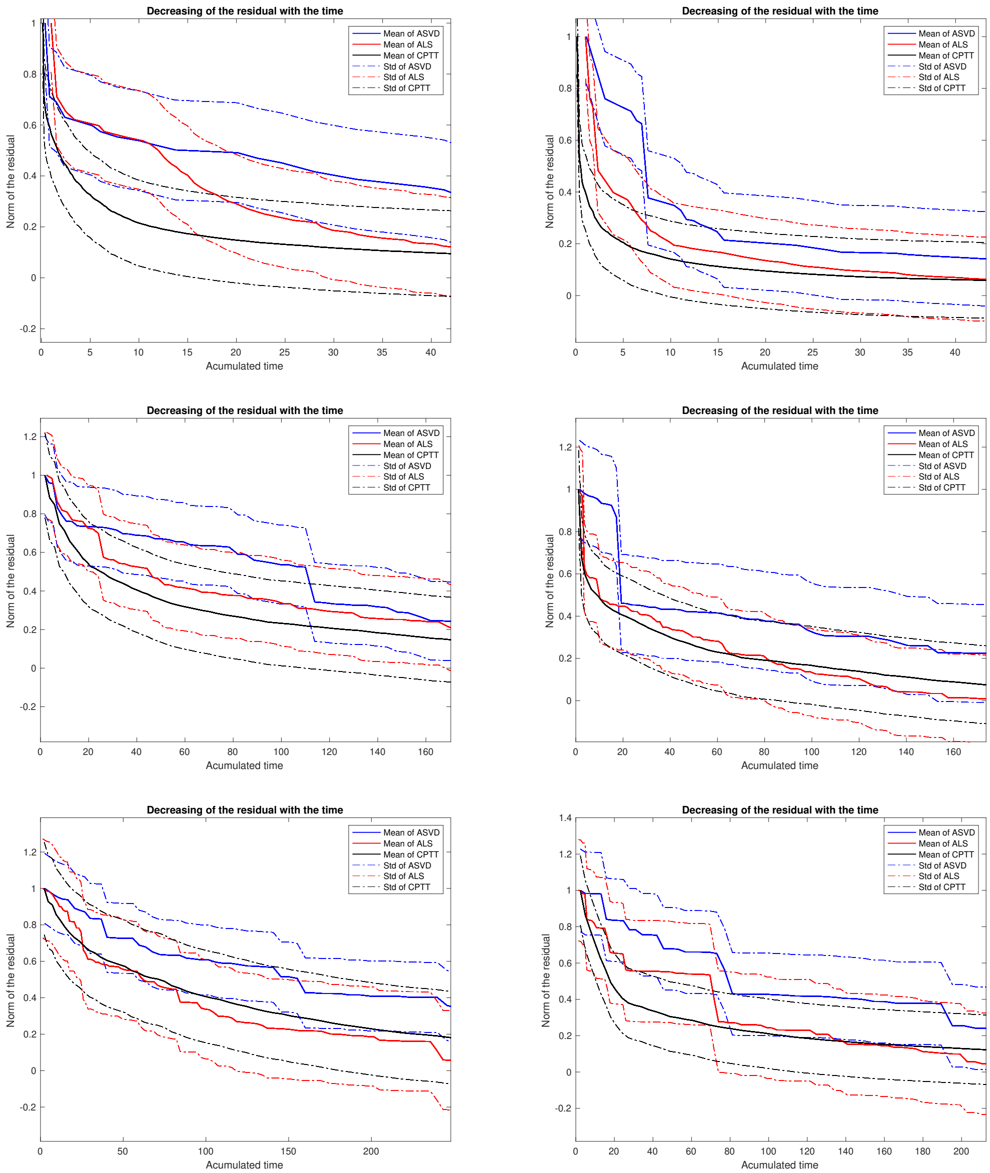


FIG. 5. *Left:* $\beta = \frac{d}{2} + 0.1$. *Right:* $\beta = \frac{d}{2} + 1.1$. *From top to bottom:* $d = 4, 12, 16$. *Mean and standard deviation of the norm of the residual as a function of the accumulated time of computation for ALS (red), ASVD (blue) and CP-TT (black).*

algorithm, which can be extended to the case of the CP-TT algorithm. Several numerical experiments are proposed to illustrate the properties of the methods. First, we compared the CP-TT to other rank-one update methods (ALS, ASVD, TTr1). Although a single iteration of CP-TT is more expensive in terms of number of operation, its stability makes it a promising candidate to compress high-dimensional tensors in CP format. We proposed a test in which we compressed the numerical solution of a

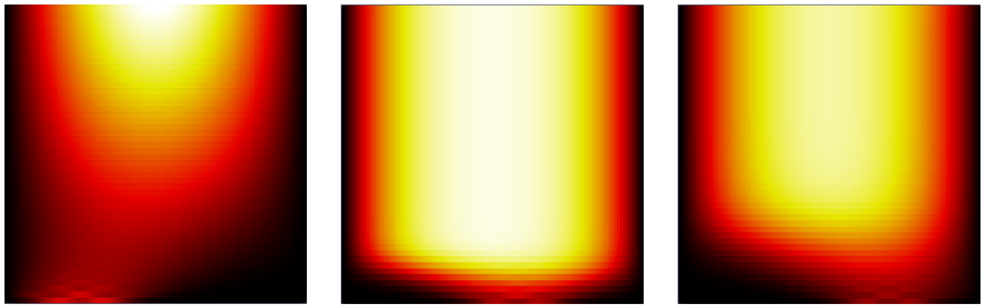


FIG. 6. *Three slices of the full tensor used in Section 5.2. The horizontal axis is the space coordinate, the vertical axis is the time coordinate, the color represents the solution value, from 0 (black), to 1 (white) for different values of the parameters determining the initial condition and the reaction coefficient.*

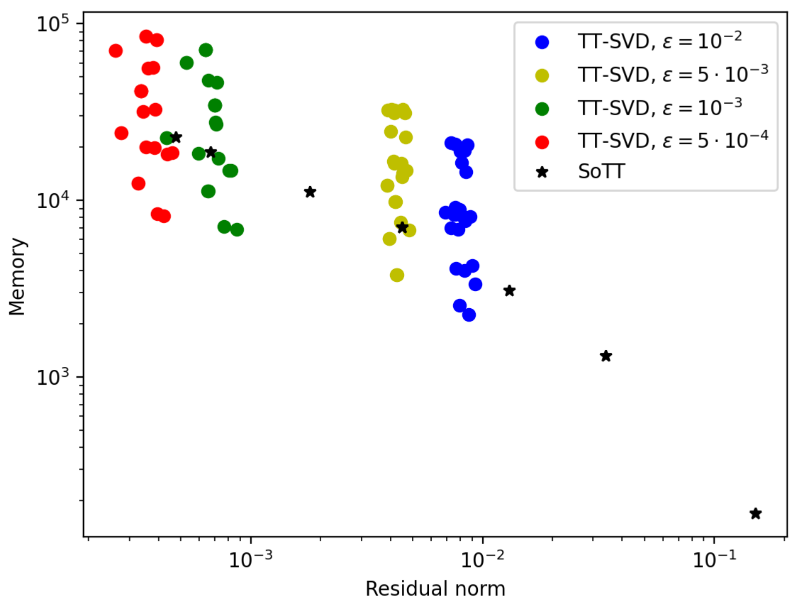


FIG. 7. *Compression test performed in Section5.2, double logarithmic plot of the memory as function of the residual norm for the TT-SVD runs (obtained by considering all the possible permutations of the indices), for several tolerances, and the SoTT approximations.*

parametric partial differential equation of reaction-diffusion type. In particular, we compared SoTT with the TT-SVD obtained by testing all the possible permutations of the indices. Although SoTT is suboptimal with respect to the best TT-SVD, it is independent of the order of the variables and its performances are comparable to the average TT-SVD. In this test, the SoTT method outperforms CP-TT. Both methods showed preliminary yet encouraging results in view of applications in scientific computing and compression of high order tensors.

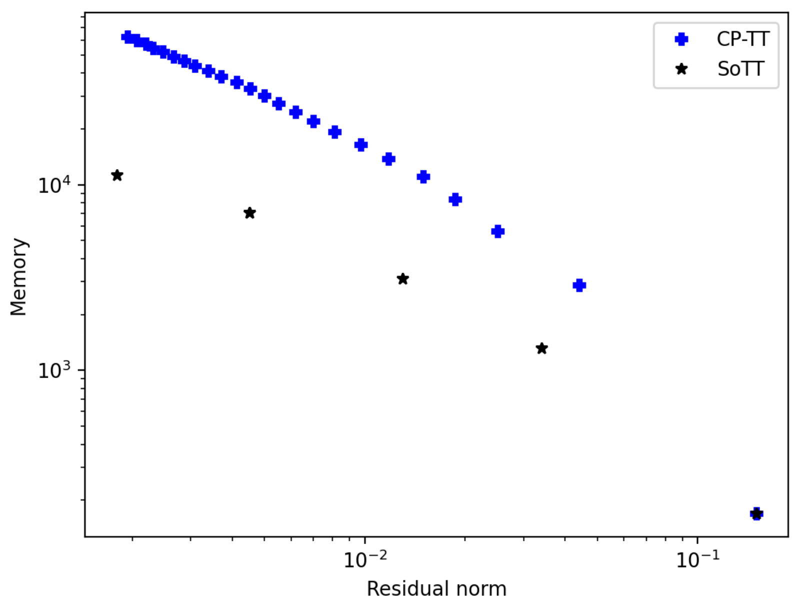


FIG. 8. *Compression test performed in Section5.2, double logarithmic plot of the memory as function of the residual norm for the SoTT and the CP-TT algorithms.*

Acknowledgments. Virginie Ehrlacher acknowledges support from the ANR COMODO project (ANR-19-CE46-0002).

Damiano Lombardi acknowledges support from the ANR ADAPT project (ANR-18-CE46-0001).

This publication is part of the EMC2 project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme – Grant Agreement $n°$ 810367.

REFERENCES

[1] K. Batselier, H. Liu, and N. Wong, A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 1315–1337.

[2] G. Beylkin and M. J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, SIAM J. SCI. COMPUT, 26 (2005), pp. 2133–2159.

[3] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, Spectral tensor-train decomposition, SIAM Journal on Scientific Computing, 38 (2016), pp. A2405–A2439.

[4] A. Cichocki, N. Lee, I. Oseledets, A. P. amd Q. Zhaonand, and D. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Now Publishers Inc., 35 (2016), https://doi.org/10.1561/2200000059.

[5] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends® in Machine Learning, 9 (2016), pp. 249–429.

[6] V. de Silva and L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM Journal on Matrix Analysis and Applications, 30 (2008), p. 1084–1127, https://doi.org/10.1137/06066518X.

[7] I. Domanov and L. Lathauwer, On uniqueness and computation of the decomposition of

a tensor into multilinear rank-(1, lr, lr) terms, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 747–803.
[8] I. Domanov and L. D. Lathauwer, Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 636–660, https://doi.org/10.1137/130916084.
[9] M. Espig, W. Hackbusch, and A. Khachatryan, On the convergence of alternating least squares optimisation in tensor format representations, arXiv preprint arXiv:1506.00062, (2015).
[10] S. Friedland, V. Mehrmann, R. Pajarola, and S. K. Suter, On best rank one approximation of tensors, Numerical Linear Algebra with Applications, 20 (2013), pp. 942–955.
[11] S. Friedland and G. Ottaviani, The number of singular vector tuples and uniqueness of best rank-one approximation of tensors, Found Comput Math, 14 (2014), p. 1209–1242, https://doi.org/10.1007/s10208-014-9194-z.
[12] L. Grasedyck, D. Kressner, and C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen, 36 (2013), pp. 53–78.
[13] W. Hackbusch, Tensor spaces and numerical tensor calculus, vol. 42, Springer, 2012.
[14] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics, 6 (1927), pp. 164–189.
[15] B. Khoromskij and V. Khoromskaia, Low rank tucker-type tensor approximation to classical potentials, Open Mathematics, 5 (2007), pp. 523–550.
[16] B. N. Khoromskij, Tensor numerical methods in scientific computing, vol. 19, Walter de Gruyter GmbH & Co KG, 2018.
[17] B. N. Khoromskij and V. Khoromskaia, Multigrid accelerated tensor approximation of function related multidimensional arrays, SIAM Journal on Scientific Computing, 31 (2009), pp. 3002–3026.
[18] T. Kolda and B. Bader, Tensor decompositions and applications, SIAM Review, 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111X.
[19] T. G. Kolda, Tensor decomposition: A mathematical tool for data analysis., tech. report, Sandia National Lab.(SNL-CA), Livermore, CA (United States), 2018.
[20] T. G. Kolda and J. Sun, Scalable tensor decompositions for multi-aspect data mining, in 2008 Eighth IEEE international conference on data mining, IEEE, 2008, pp. 363–372.
[21] K. Kour, S. Dolgov, M. Stoll, and P. Benner, Efficient structure-preserving support tensor train machine, arXiv preprint arXiv:2002.05079, (2020).
[22] I. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
[23] I. Oseledets and E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use svd in many dimensions, SIAM Journal on Matrix Analysis and Applications, 31 (2009), p. 1084–1127, https://doi.org/10.1137/090748330.
[24] I. V. Oseledets, M. V. Rakhuba, and A. Uschmajew, Alternating least squares as moving subspace correction, SIAM Journal on Numerical Analysis, 56 (2018), pp. 3459–3479.
[25] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavsky, V. Glukhov, I. Oseledets, and A. Cichocki, Stable low-rank tensor decomposition for compression of convolutional neural network, ECCV2020, (2020), https://doi.org/arXiv:2008.05441.
[26] M. Rajih, P. Comon, and R. Harsman, Enhanced line search: A novel method to accelerate parafac, SIAM Journal on Matrix Analysis and Applications, 30 (2008), https://doi.org/10.1137/06065577.
[27] M. Rakhuba and I. Oseledets, Calculating vibrational spectra of molecules using tensor train decomposition, The Journal of Chemical Physics, 145 (2016), p. 124101.
[28] T. Rohwedder and A. Uschmajew, On local convergence of alternating schemes for optimization of convex problems in the tensor train format, SIAM Journal on Numerical Analysis, 51 (2013), pp. 1134–1162.
[29] L. Sorber, M. Van Barel, and L. De Lathauwer, Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l_r,l_r,1) terms, and a new generalization, SIAM Journal on Optimization, 23 (2013), pp. 695–720.
[30] V. N. Temlyakov, Greedy algorithms andm-term approximation with regard to redundant dictionaries, Journal of Approximation Theory, 98 (1999), pp. 117–145.
[31] A. Uschmajew, Local convergence of the alternating least squares algorithm for canonical tensor approximation, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 639–652.
[32] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, Nonlinear least squares updating of the canonical polyadic decomposition, in 2017 25th European Signal Processing Conference (EUSIPCO), 2017, pp. 663–667.

[33] X. Wang, C. Navasca, and S. Kindermann, On accelerating the regularized alternating least square algorithm for tensors, arXiv preprint arXiv:1507.04721, (2015).
[34] X. Wang, L. T. Yang, Y. Wang, X. Liu, Q. Zhang, and M. J. Deen, A distributed tensor-train decomposition method for cyber-physical-social services, ACM Transactions on Cyber-Physical Systems, 3 (2019), pp. 1–15.
[35] T. Zhang and G. H. Golub, Rank-one approximation to high order tensors, SIAM Journal on Matrix Analysis and Applications, 23 (2001), pp. 534–550.