Linear Algebra and its Applications 432 (2010) 70–88

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

TT-cross approximation for multidimensional arrays☆

Ivan Oseledets, Eugene Tyrtyshnikov*

Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkin Street, 8, Moscow 119333, Russia

ARTICLE INFO

Article history:
Received 19 July 2009
Accepted 27 July 2009
Available online 21 August 2009

Submitted by V. Sergeichuk

AMS classification:
15A12
65F10
65F15

Keywords:
Tensor decompositions
Multi-way arrays
Curse of dimensionality
Interpolation
Multidimensional integration
Cross approximation
Tensor trains
TT decomposition
Singular value decomposition
Low-rank matrices

ABSTRACT

As is well known, a rank-r matrix can be recovered from a cross of r linearly independent columns and rows, and an arbitrary matrix can be interpolated on the cross entries. Other entries by this *cross* or *pseudo-skeleton* approximation are given with errors depending on the closeness of the matrix to a rank-r matrix and as well on the choice of cross. In this paper we extend this construction to d-dimensional arrays (tensors) and suggest a new interpolation formula in which a d-dimensional array is interpolated on the entries of some *TT-cross* (tensor-train-cross). The total number of entries and the complexity of our interpolation algorithm depend on d linearly, so the approach does not suffer from the curse of dimensionality.

We also propose a TT-cross method for computation of d-dimensional integrals and apply it to some examples with dimensionality in the range from $d = 100$ up to $d = 4000$ and the relative accuracy of order 10^{-10}. In all constructions we capitalize on the new tensor decomposition in the form of *tensor trains* (TT-decomposition).

1. Introduction

Multidimensional arrays can be encountered in many applications but a direct numerical treatment of arrays in really many dimensions is impossible due to the *curse of dimensionality*. By the curse of dimensionality we mean that the memory required to store an array with d indices and the amount of

☆ This work was supported by RFBR Grants 08-01-00115, 09-01-00565, 09-01-12058, RFBR/DFG Grant 09-01-91332 and Priority Research Programme of Department of Mathematical Sciences of Russian Academy of Sciences.

* Corresponding author.
E-mail addresses: ivan.oseledets@gmail.com (I. Oseledets), tee@inm.ras.ru (E. Tyrtyshnikov).

0024-3795/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2009.07.024

operations required to perform basic operations with such an array grows exponentially in the dimensionality d. Therefore, direct manipulation of general d-dimensional arrays seems to be impossible. It is indeed so unless a multidimensional array (we will call it also a d-dimensional tensor) comes from some physical problem. That implies that it is not arbitrary but possesses some hidden structure that has to be revealed. Sometimes the structure is obvious and follows from the model. This includes sparse tensors [2] and tensors with shift-invariant (Toeplitz or Hankel) structure [1]. However, these classes are not very general, and some other low-parametric representation is needed.

Possibly the most popular one is the *canonical decomposition* [27,9,11,5,6] in which a d-dimensional array $A(i_1, i_2, \ldots, i_d)$ is expressed (maybe approximately) in the form

$$A(i_1, i_2, \ldots, i_d) \approx \sum_{\alpha=1}^{R} U_1(i_1, \alpha) U_2(i_2, \alpha) \cdots U_d(i_d, \alpha), \tag{1}$$

the number of terms R is called (approximate) *canonical rank* of the representation (1), and matrices $U_k = [U_k(i_k, \alpha)]$ are $n_k \times R$ and called *factor matrices*. The model (1) is also known in the literature as CANDECOMP/PARAFAC model [27,9]. Some state-of-the-art issues in multilinear algebra and tensor decompositions are sketched in the recent review [3].

For several known classes of tensors (arising, for example, from discretization of integral or differential operators), by analytical techniques one can prove the existence of low-tensor-rank approximations to $A(i_1, i_2, \ldots, i_d)$ (cf. [40,26,28,10,5,6,25]). Such approaches are often constructive and lead to sufficiently small R. The memory to store the factor matrices is [1] $\mathcal{O}(dnR)$ and is acceptable if R is small (of order of tens or sometimes hundreds).

However, analytical considerations more frequently give suboptimal values of ranks (affordable but not optimal) and there are no robust numerical methods to reduce this rank while maintaining the approximation accuracy (such a procedure is known as *recompression*). There are successful approaches [13,5,6] but even when using them you have to guess the value of the rank and the convergence may be slow.

Here we come to the main topic of this paper. In all analytical considerations and in many practical cases a tensor is given implicitly by a procedure enabling us to compute any its element. So the tensor appears rather as a black box and we need to solve the following problem: given a procedure for computation of tensor elements, find some suitable low-parametric approximation of this tensor using only a small portion of all tensor elements. How this can be done and what tensor representation could fit the task?

Here the situation is well studied only in two and three dimensions. If $d = 2$ then the canonical decomposition is nothing else than the dyadic (skeleton) decomposition for matrices. And for black-box matrices a fruitful and simple cross approximation method [4,39] is available. It allows one to approximate large close-to-rank-r matrices in $\mathcal{O}(nr^2)$ time by computing only $\mathcal{O}(nr)$ elements.

If $d > 2$ then the situation is more complicated, and the canonical decomposition is not very convenient for black-box tensors. Instead of it, for construction of the black-box approximation it was proposed in [34] to use the *Tucker decomposition* of the form [38]

$$A(i_1, \ldots, i_d) \approx \sum_{\alpha_1, \ldots, \alpha_d} G(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \cdots U_d(i_d, \alpha_d). \tag{2}$$

The summation indices α_k take values from 1 to ρ_k, and ρ_k are called the *Tucker ranks*.[2] There are different naming conventions for (2), and if additional orthogonality assumptions on the *core tensor* G and *Tucker factors* are imposed then (2) becomes (truncated) Higher Order Singular Value Decomposition (HOSVD) [11,12]. We, however, do not require these orthogonality properties and will refer to any representation of the form (2) as the (truncated) Tucker decomposition, and the word "truncated" will be omitted for brevity.

The Tucker decomposition gives a nice compression rate provided that the Tucker ranks are much smaller than the mode size. The difference between the canonical decomposition and the Tucker

[1] Here and below, for complexity estimates let us assume that all mode sizes are equal: $n_1 = n_2 = \cdots = n_d = n$.

[2] In complexity estimates we usually assume that $\rho_1 = \cdots = \rho_d = \rho$.

decomposition is that a quasi-optimal Tucker decomposition can be computed by a sequence of SVD [19] approximations to so-called *unfolding matrices*, i.e. by using standard subroutines.

In three dimensions, one has to store additionally only $\mathcal{O}(\rho^3)$ elements of the core tensor, which is often negligible for large n, and the matrix cross approximation method can be (non-trivially) generalized to the computation of the Tucker approximation in three dimensions [34] so that only $\mathcal{O}(n\rho + \rho^3)$ tensor elements are used. An existence theorem for such a cross approximation for three-dimensional tensors was first given in [34] and then a refined version [20] was proved as a generalization to d dimensions of the corresponding result for the pseudoskeleton decomposition of matrices [23].

The Tucker decomposition requires to store a full d-dimensional core tensor with the number of elements ρ^d. If d becomes large, say $d > 10$, it is infeasible even for small ρ and definitely suggests that something else has to be done. Recently [30,31,35,36] a new decomposition, named *TT decomposition*, was proposed for compact representation and *approximation* of high-dimensional tensors. It can be computed via standard decompositions (such as the SVD and QR) but does not suffer from the curse of dimensionality. The TT decomposition is written as a *tensor train* of the form

$$A(i_1, i_2, \ldots, i_d) \approx \sum_{\alpha_1,\ldots,\alpha_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) G_d(\alpha_{d-1}, i_d), \tag{3}$$

with *tensor carriages* $G_1, \ldots, G_d$, where any two neighbors have a common summation index.

The summation indices α_k run from 1 to r_k and are referred to as *auxiliary indices*, in contrast to the initial indices i_k that are called *spacial indices*. The quantities r_k are called *compression ranks*.[3] The *tensor carriages* G_k have sizes $r_{k-1} \times n_k \times r_{k+1}$ except for $k = 1$ and $k = d$ where they have sizes $n_1 \times r_1$ and $r_{d-1} \times n_d$, respectively. It is sometimes convenient to assume that G_1 and G_d are not two-dimensional but three-dimensional with sizes $1 \times n_1 \times r_1$ and $r_{d-1} \times n_d \times 1$ and additional auxiliary indices $\alpha_0 = \alpha_d = 1$ and compression ranks $r_0 = r_d = 1$. This makes the decomposition look more symmetric and simplifies certain algorithms.

It was shown in [30] that compression ranks r_k are bounded from below by the ranks of auxiliary unfolding matrices, a TT decomposition with minimal possible compression ranks always exists and, moreover, can be computed by a sequence of SVD decompositions. The ranks r_k are also bounded from above by the canonical rank R of the tensor. However, it is remarkable that they can be much smaller, and often the TT format gives better compression than the canonical one. A really big gain in comparison to the canonical format is that the TT format admits an efficient and effective recompression procedure. It implies that one can use some iterative method with compression at each step and obtain an approximate solution to high-dimensional equations.

If some canonical representation is already known (e.g. for discrete analogs of some operators), then a fast canonical-to-TT conversion algorithm is available and often leads to a reduced TT decomposition with fewer representation parameters. However, more frequently we have to deal with tensors given implicitly via a subroutine that allows one to compute any prescribed element of a tensor. The principal question addressed in this paper is whether it is possible to treat black-box tensors in the spirit of previous cross approximation techniques but in the TT format.

The method of [14] interpolates the tensor at some *fibre crosses* but exploits the canonical representation and, as a consequence, provides no guarantee: even if a tensor possesses a low-canonical-rank approximation, then a good approximation might not be found. Now we can fall back to the new tool such as the TT decomposition. And since the TT decomposition can be computed via low-rank approximation of auxiliary matrices, it is natural to expect that it be possible to find out some interpolation formula based on tensor trains.

The main contribution of this paper is exactly the new interpolation formula in d dimensions. It generalizes the skeleton (dyadic) decomposition to d dimensions via the tensor-train format, and it turns out that if all compression ranks are bounded by r then it is sufficient to compute only $\mathcal{O}(dnr^2)$ elements of a tensor in certain prescribed positions and use them to completely recover this tensor.

[3] In complexity estimates we usually assume that $r_1 = \cdots = r_{d-1} = r$.

Thus, from a virtual (black-box) tensor on input we get to a tensor-train representation of this tensor on output.

A simple variant of the TT cross algorithm is presented and convincing numerical experiments are given. Then the TT cross algorithm is applied to approximate some black-box tensors arising in the numerical computation of d-dimensional integrals.

Throughout the paper we will use several notations from linear algebra that are now becoming standard: $\|\mathbf{A}\|_F$ is the Frobenius norm of a tensor, $\mathbf{A} \times_k B$ is the tensor-by-matrix multiplication ($\mathbf{A}$ is a tensor and B is a matrix), and others. For the purpose of not distracting the reader with notations, we do not give them in detail here, and refer to previous works, for example [35] or to a comprehensive review [3]. Standard MATLAB notation will be used for handling tensor and matrix operations, especially to describe algorithms.

2. Computing TT decomposition by a sequence of singular value decompositions

The TT decomposition (3) can be computed by a sequence of SVD decompositions. For a given tensor $\mathbf{A} = [A(i_1, \dots, i_d)]$ consider the following *unfolding matrices* A_k:

$$A_k = [A(i_1 \dots i_k; i_{k+1} \dots i_d)],$$

i.e. the first k indices enumerate the rows of A_k and the last $d - k$ ones enumerate the columns. Here and below we use semicolon to separate the row and column indicators in the form of *long indices* (multi-indices [4]).

Theorem 2.1 [30]. *For any tensor $\mathbf{A} = [A(i_1, \dots, i_d)]$ there exists a TT decomposition with compression ranks*

$$r_k = \operatorname{rank} A_k.$$

As a complement, it is not difficult to prove that $\operatorname{rank} A_k \leqslant R$, where R is the canonical rank of the tensor.

Theorem 2.1 pertains to the "exact case", i.e. when the tensor is represented by tensor trains exactly. In applications it is crucial to consider the "approximate case", i.e. when each matrix A_k might be not of low rank exactly but admits a low-rank approximation with some accuracy ε_k. Theorem 2.1 can be extended to that case in the following way.

Theorem 2.2. *For any tensor $\mathbf{A} = [A(i_1, \dots, i_d)]$ there exists a TT approximation $\mathbf{T} = [T(i_1, \dots, i_d)]$ with compression ranks r_k such that*

$$\|\mathbf{A} - \mathbf{T}\|_F \leqslant \sqrt{\sum_{k=1}^{d-1} \varepsilon_k^2}, \tag{4}$$

where ε_k is the distance (in the Frobenius norm) from A_k to its best rank-r_k approximation:

$$\varepsilon_k = \min_{\operatorname{rank} B \leqslant r_k} \|A_k - B\|_F.$$

Proof. Consider first the case $d = 2$. The TT decomposition in this case reads

$$T(i_1, i_2) = \sum_{\alpha_1=1}^{r_1} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2)$$

and coincides with the skeleton (dyadic) decomposition of the matrix T. Using the SVD of A, we obtain the best rank-r_1 approximation T by keeping only r_1 senior singular values and nullifying the others.

[4] When considering a multi-index as one *long index* we can always take the lexicographical ordering; however, in most places the ordering is not essential.

This choice of T guarantees that the norm $\|A - T\|_F = \varepsilon_1$ is minimal possible. Thus, the case $d = 2$ follows from the standard matrix theory.

Then, proceed by induction. Consider the first unfolding matrix and its SVD in the form

$$A_1 = [A(i_1; i_2 \dots i_d)] = U\Sigma V, \tag{5}$$

where U has orthonormal columns, V has orthonormal rows, and Σ is a diagonal matrix of the singular values $\sigma_1 \geqslant \sigma_2 \geqslant \cdots$. As an approximation to A_1, consider

$$B_1 = U_1 \Lambda V_1, \quad \Lambda = \operatorname{diag}(\sigma_1, \dots, \sigma_{r_1}), \tag{6}$$

where U_1 and V_1 contain the first r_1 columns of U and rows of V_1, respectively. Then B_1 is the best rank-r_1 approximation to A_1, i.e.

$$A_1 = B_1 + E_1, \quad \operatorname{rank} B_1 \leqslant r_1, \quad \|E_1\|_F = \varepsilon_1.$$

Obviously, B_1 can be considered as a tensor $\mathbf{B} = [B(i_1, \dots, i_d)]$ and the approximation problem reduces to the one for $\mathbf{B}$.

It is important for us to observe the following: if we take an arbitrary tensor $\mathbf{T} = [T(i_1, \dots, i_d)]$ with the first unfolding matrix $T_1 = [T(i_1; i_2, \dots, i_d)]$ in the form

$$T_1 = U_1 W \tag{7}$$

with U_1 from (6) and an arbitrary matrix W with r_1 rows and as many columns as in T_1, then $E_1^* T_1 = 0$ and this implies that

$$\|(\mathbf{A} - \mathbf{B}) + (\mathbf{B} - \mathbf{T})\|_F^2 = \|\mathbf{A} - \mathbf{B}\|_F^2 + \|\mathbf{B} - \mathbf{T}\|_F^2. \tag{8}$$

Note, however, that the tensor $\mathbf{B}$ is still of dimensionality d. To reduce dimensionality, rewrite the matrix equality (6) in the element-wise form

$$B(i_1; i_2, \dots, i_d) = \sum_{\alpha_1 = 1}^{r_1} U_1(i_1; \alpha_1) \widehat{A}(\alpha_1; i_2, \dots, i_d),$$

where

$$\widehat{A} = \Lambda V_1.$$

Then, concatenate indices α_1 and i_2 into one long index and consider $\widehat{A}$ as a tensor

$$\widehat{\mathbf{A}} = [\widehat{A}(\alpha_1 i_2, i_3, \dots, i_d)]$$

of dimensionality $d - 1$.

By induction, $\widehat{\mathbf{A}}$ admits a TT approximation $\widehat{\mathbf{T}} = [\widehat{T}(\alpha_1 i_2, i_3, \dots, i_d)]$ of the form

$$\widehat{T}(\alpha_1 i_2, i_3, \dots, i_d) = \sum_{\alpha_2, \dots, \alpha_{d-1}} G_2(\alpha_1 i_2, \alpha_2) G_3(\alpha_2, i_3, \alpha_3) \cdots G_d(\alpha_{d-1}, i_d)$$

such that

$$\|\widehat{\mathbf{A}} - \widehat{\mathbf{T}}\|_F \leqslant \sqrt{\sum_{k=2}^{d-1} \hat{\varepsilon}_k^2}$$

with

$$\hat{\varepsilon}_k = \min_{\operatorname{rank} C \leqslant r_k} \|\widehat{A}_k - C\|_F, \quad \widehat{A}_k = [\widehat{A}(\alpha_1 i_2, \dots i_k; i_{k+1}, \dots, i_d)].$$

Now let us set $G_1(i_1, \alpha_1) = U(i_1, \alpha_1)$, separate indices α_1, i_2 from the long index $\alpha_1 i_2$ and define $\mathbf{T}$ by the following tensor train:

$$T(i_1, \dots, i_d) = \sum_{\alpha_1, \dots, \alpha_d} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_3) \cdots G_d(\alpha_{d-1}, i_d).$$

It remains to estimate $\|\mathbf{A} - \mathbf{T}\|_F$. First of all, from (5) and (6) it stems that

$$\widehat{A} = \Lambda V_1 = U_1^* A_1,$$

and consequently,[5]

[5] Overlined numbers mean complex conjugates.

$$\widehat{A}(\alpha_1 i_2, i_3, \ldots, i_d) = \sum_{i_1} \overline{U}_1(i_1, \alpha_1) A(i_1, i_2, \ldots, i_d).$$

Let $A_k = B_k + E_k$ with $\mathrm{rank} B_k \leqslant r_k$ and $\|E_k\|_F = \varepsilon_k$. We can consider B_k and E_k as tensors $B_k(i_1, \ldots, i_d)$ and $E_k(i_1, \ldots, i_d)$. Since B_k admits a skeleton decomposition with r_k terms, we obtain

$$A(i_1, \ldots, i_d) = \sum_{\gamma=1}^{r_k} P(i_1, \ldots, i_k; \gamma) Q(\gamma; i_{k+1}, \ldots, i_d) + E_k(i_1, \ldots, i_d).$$

Hence, $\widehat{A}(\alpha_1 i_2, i_3, \ldots, i_d) = H_k(\alpha_1, i_2, i_3, \ldots, i_d) + R_k(\alpha_1, i_2, i_3, \ldots, i_d)$ with

$$H_k(\alpha_1, i_2, i_3, \ldots, i_d) = \sum_{i_1} \overline{U}_1(i_1, \alpha_1) \sum_{\gamma=1}^{r_k} P(i_1, \ldots, i_k; \gamma) Q(\gamma; i_{k+1}, \ldots, i_d),$$

$$R_k(\alpha_1, i_2, i_3, \ldots, i_d) = \sum_{i_1} \overline{U}_1(i_1, \alpha_1) E_k(i_1, \ldots, i_d).$$

Let us introduce a tensor L as follows:

$$L(\alpha_1, i_2, \ldots, i_k, \gamma) = \sum_{i_1} \overline{U}_1(i_1, \alpha_1) P(i_1, \ldots, i_k; \gamma).$$

Then we can consider H_k as a matrix with the elements defined by a skeleton decomposition

$$H_k(\alpha_1, i_2, \ldots, i_k; i_{k+1}, \ldots, i_d) = L(\alpha_1, i_2, \ldots, i_k; \gamma) Q(\gamma; i_{k+1}, \ldots, i_d)$$

and it makes it evident that the rank of H_k does not exceed r_k. As well we can consider R_k as a matrix with the elements defined by

$$R_k(\alpha_1; i_2, i_3, \ldots, i_d) = \sum_{i_1} \overline{U}_1(i_1; \alpha_1) E_k(i_1; i_2, \ldots, i_d).$$

We know that U_1 has orthonormal columns, and it means that the matrix E_k is premultiplied by a matrix with orthonormal rows. Since this cannot increase its Frobenius norm, we conclude that

$$\hat{\varepsilon}_k \leqslant \|R_k\|_F \leqslant \|E_k\|_F = \varepsilon_k, \quad 2 \leqslant k \leqslant d-1.$$

Hence, for the error tensor $\widehat{\mathbf{E}}$ with the elements

$$\widehat{E}(\alpha_1 i_2, i_3, \ldots, i_d) = \widehat{A}(\alpha_1 i_2, i_3, \ldots, i_d) - \widehat{T}(\alpha_1 i_2, i_3, \ldots, i_d),$$

we obtain

$$\|\widehat{\mathbf{E}}\|_F \leqslant \sqrt{\sum_{k=2}^{d-1} \varepsilon_k^2}.$$

Further, the error tensor $\mathbf{E} = \mathbf{B} - \mathbf{T}$ can be considered as matrix of the form

$$E(i_1; i_2, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} U_1(i_1; \alpha_1) \widehat{E}(\alpha_1; i_2, \ldots, i_d),$$

which shows that the matrix $\widehat{E}$ is premultiplied by a matrix with orthonormal rows, so we have

$$\|\mathbf{E}\|_F \leqslant \|\widehat{\mathbf{E}}\|_F \leqslant \sqrt{\sum_{k=2}^{d-1} \varepsilon_k^2}.$$

Finally, observe that the first unfolding matrix T_1 for $\mathbf{T}$ is exactly of the form (7). Thus, (8) is valid and completes the proof. □

Corollary 2.1. *If a tensor* $\mathbf{A}$ *admits a canonical approximation with* R *terms and accuracy* ε, *then there exists a tensor-train approximation with compression ranks* $r_k \leqslant R$ *and accuracy* $\sqrt{d-1}\,\varepsilon$.

Corollary 2.2. *Given a tensor* $\mathbf{A}$, *denote by* $\varepsilon = \inf_{\mathbf{B}} \|\mathbf{A} - \mathbf{B}\|_F$ *the infimum of distances between* $\mathbf{A}$ *and tensor trains* $\mathbf{B}$ *with prescribed upper bounds* r_k *on the ranks of unfolding matrices (compression ranks), i.e.* $\operatorname{rank} B_k \leqslant r_k$. *Then the optimal* $\mathbf{B}$ *exists (the infimum is in fact minimum) and the tensor-train approximation* $\mathbf{T}$ *constructed in the proof of Theorem 2.2 is quasi-optimal in the sense that*

$$\|\mathbf{A} - \mathbf{T}\|_F \leqslant \sqrt{d-1}\varepsilon.$$

Proof. By the definition of infimum, there exists a sequence of tensor trains $\mathbf{B}^{(s)}$ $(s = 1, 2, \ldots,)$ with the property $\lim_{s\to\infty} \|\mathbf{A} - \mathbf{B}^{(s)}\|_F = \varepsilon$. If we knew that all elements of the corresponding tensor carriages were bounded, then we would immediately claim that there is a subsequence of $\mathbf{B}^{(s)}$ with converging sequences of tensor carriages. We cannot say that much at this moment. Nevertheless, all *elements* of the tensors $\mathbf{B}^{(s)}$ are bounded, and hence, some subsequence $\mathbf{B}^{(s_t)}$ converges element-wise to some tensor $\mathbf{B}^{(\min)}$. The same holds true for the corresponding unfolding matrices: $B_k^{(s_t)} \to B_k^{(\min)}$, $1 \leqslant k \leqslant d$. It is well known that a sequence of matrices with a common rank bound cannot converge to a matrix with a larger rank. Thus, $\operatorname{rank} B^{(s_t)} \leqslant r_k$ implies that $\operatorname{rank} B_k^{(\min)} \leqslant r_k$ and $\|\mathbf{A} - \mathbf{B}^{(\min)}\|_F = \varepsilon$, so $\mathbf{B}^{(\min)}$ is the minimizer. It is now sufficient to note that $\varepsilon_k \leqslant \varepsilon$. The reason is that ε is the approximation accuracy for every unfolding matrix A_k delivered by a special structured skeleton (dyadic) decomposition with r_k terms while ε_k stands for the best approximation accuracy without any restriction on the vectors of skeleton decomposition. Hence, $\varepsilon_k \leqslant \varepsilon$. Then the quasi-optimality bound follows directly from (4). □

The proof of Theorem 2.2 also gives a constructive method for computing a TT-approximation. It is summarized in Algorithm 1.

Algorithm 1. Full-to-TT compression algorithm

Require: a tensor $\mathbf{A}$ of size $n_1 \times n_2 \cdots \times n_d$ and accuracy bound ε.
Ensure: tensor carriages G_k, $k = 1, \ldots, d$, defining a TT (tensor-train) approximation to $\mathbf{A}$ with the relative error bound ε.

1: Compute $\mathtt{nrm} := \|\mathbf{A}\|_F$.
2: Sizes of the first unfolding matrix: $N_l = n_1, N_r = \prod_{k=2}^{d} n_k$.
3: Temporary tensor: $\mathbf{B} := \mathbf{A}$.
4: First unfolding: $M := \operatorname{reshape}(\mathbf{B}, [N_l, N_r])$.
5: Compute the SVD of $M \approx U\Lambda V$ truncated so that the approximate rank r satisfies

$$\sqrt{\sum_{k=r+1}^{\min(N_l,N_r)} \sigma_k^2} \leqslant \frac{\varepsilon \cdot \mathtt{nrm}}{\sqrt{d-1}}.$$

6: Set $G_1 := U, M := \Lambda V^{\top}, r_1 = r$.
7: {Process other modes}
8: **for** $k = 2$ **to** $d - 1$ **do**
9: Redefine the sizes: $N_l := n_k, N_r := \frac{N_r}{n_k}$.
10: Construct the next unfolding: $M := \operatorname{reshape}(M, [rN_l, N_r])$.

Algorithm 1 (*Continued*).

11: Compute the SVD of $M \approx U\Lambda V$ truncated so that the approximate rank r satisfies

$$\sqrt{\sum_{k=r+1}^{\min(N_l,N_r)} \sigma_k^2} \leqslant \frac{\varepsilon \cdot \mathtt{nrm}}{\sqrt{d-1}}.$$

12: Reshape the matrix U into a tensor:

$$G_k := \mathrm{reshape}(U, [r_{k-1}, n_k, r_k]).$$

13: Recompute $M := \Lambda V$.

14: **end for**

15: $G_d = M$.

3. Skeleton decomposition of matrices and of tensors

A big problem with application of Algorithm 1 is in computation of the truncated SVD for large-scale and possibly dense unfolding matrices. It is clearly unaffordable in many dimensions.

An agreeable solution is to replace SVD for A_k by some other dyadic decomposition

$$A_k \approx UV^{\top},$$

which can be computed with lower complexity. An excellent candidate is the *skeleton* [16] or *pseudoskeleton* decomposition [22,23].

3.1. Skeleton decomposition in the exact case

Let us recall what the skeleton decomposition is. If a $m \times n$ matrix A has rank r, then it can be represented as

$$A = C\widehat{A}^{-1}R, \tag{9}$$

where $C = A(:, \mathcal{J})$ are some r columns of A, $R = A(\mathcal{I}, :)$ are some r rows of A and

$$\widehat{A} = A(\mathcal{I}, \mathcal{J})$$

is the submatrix on their intersection that should be nonsingular. From (9) it follows that a rank-r matrix can be recovered from r linearly independent columns and r linearly independent rows. An important question is how to select such rows and columns in the case when a matrix is only approximately of low rank. In that case an answer is to use the intersection submatrix with maximal volume (i.e. determinant in modulus) [24]. In practice, such a submatrix can be replaced by a certain quasi-optimal one that can be computed via cross approximation techniques [39,4].

How to use the skeleton decomposition to compute the TT approximation? It is quite simple to design such an algorithm by modifying Algorithm 1, just by replacing truncated singular value decomposition with the skeleton decomposition. In the matrix case an efficient approach relies on successive rank-one corrections to the current approximation [4,15]. However, in the work with tensor trains we need some additional properties for the skeleton decomposition in two dimensions and find it most useful to fall back to constructions of the cross approximation method proposed in [39]. In the matrix case it seems to be a bit more complicated than the successive rank-1 corrections, but proves to be viable and better suited to tensor trains.

First consider the matrix case. From computational point of view, to construct a good skeleton approximation it is sufficient to know either the row or column positions that contain a submatrix

of sufficiently large volume. For example, suppose that we know the column positions, $\mathcal{J} = [j_k], k = 1, \ldots, r$. Then we have to compute the elements of an $n \times r$ matrix

$$C = A(:, \mathcal{J})$$

and it takes nr element evaluations. Then a quasi-maximal volume submatrix $\widehat{A}$ in C can be found in $\mathcal{O}(nr^2)$ operations [39] (for more details of the algorithm, called *maxvol* algorithm, and some roundabout matters see [21]). The indices of the rows that contain the required submatrix will be denoted by $\mathcal{I}$. Then the row matrix R is set to be

$$R = A(\mathcal{I}, :).$$

Thus, only one index set, either $\mathcal{I}$ or $\mathcal{J}$ is needed, and that helps a lot in the tensor case.

Now let us generalize this idea to the d-dimensional case. We will follow the same train of thought as in the proof of Theorem 2.2 and pursue the purpose of replacing the SVD with some skeleton decompositions.

Given a tensor $\mathbf{A} = [A(i_1, \ldots, i_d)]$, at the first step let us consider the first unfolding matrix A_1 and suppose that we know a set $\mathcal{J}_1$ of indices of r_1 linearly independent columns of A_1 (they exist since $\mathrm{rank} A_1 = r_1$). Each column of A_1 is naturally pointed to by a multiindex $(i_2, \ldots, i_d)$, so $\mathcal{J}_1$ is a collection of $(d-1)$-tuples:

$$\mathcal{J}_1 = [j_l^{(\alpha_1)}], \quad \alpha_1 = 1, \ldots, r_1, \ l = 2, \ldots, d,$$

where α_1 corresponds to the column number and l to the particular mode. Matrix A_1 has a skeleton decomposition of the form (9), where the matrix C consisting of r_1 columns of A_1 is of size $n_1 \times r_1$. In the black-box case, when $\mathbf{A}$ is given by a procedure for computation of any tensor entry, the size of C should be considered as small compared to the total number of the entries of $\mathbf{A}$ and we certainly afford to compute the entries

$$C(i_1, \alpha_1) = A(i_1, j_2^{(\alpha_1)}, \ldots, j_d^{(\alpha_1)}).$$

Then, in this column matrix C we compute a submatrix of quasi-maximal volume $\widehat{A}_1$ (by the *maxvol* procedure [39,21]) and retain the corresponding row indices in the set

$$\mathcal{I}_1 = [i_1^{(\alpha_1)}], \quad \alpha_1 = 1, \ldots, r_1.$$

Matrix A_1 is represented as

$$A_1 = C\widehat{A}_1^{-1}R,$$

and analogously to the SVD-based approach we set

$$G_1 = C\widehat{A}_1^{-1}$$

to become the first tensor carriage of the TT decomposition we intend to construct. If $\widehat{A}$ is of maximal volume in C then it follows that the matrix G_1 has elements not higher than 1 in modulus, and the latter property is valid for the output of the *maxvol* algorithm [39,21]. Thus, G_1 can be considered as an analogue to the unitary factor in the SVD.

Now we are going to acquire R in the tensor-train form. It consists of r_1 rows of A_1 and can be reshaped into a *subtensor* $\mathbf{R}$ of the tensor $\mathbf{A}$ as follows:

$$R(\alpha_1, i_2, \ldots, i_d) = A(i_1^{(\alpha_1)}, i_2, \ldots, i_d).$$

By concatenation of the indices α_1 and i_2 into one long index, $\mathbf{R}$ can be treated as a tensor of dimensionality $d-1$ and of size $r_1 n_2 \times n_3 \times \cdots \times n_d$. Similarly to the proof of Theorem 2.2 it can be shown that if the ranks of the unfolding matrices of $\mathbf{A}$ are equal to r_k then the corresponding ranks for $\mathbf{R}$ cannot be larger, and as soon as we obtain an exact TT representation for $\mathbf{A}$ these ranks ought to be equal to r_k.

To obtain the second carriage we consider the first unfolding matrix

$$R_2 = R(\alpha_1 i_2; i_3, i_4, \ldots, i_d)$$

in **R** and suppose that we know a set of $(d-2)$-tuples

$$\mathcal{J}_2 = [j_l^{\alpha_2}], \quad \alpha_2 = 1, \dots, r_2, \quad l = 3, \dots, d$$

indicating the columns with a quasi-maximal volume submatrix in R_2. Then, in order to obtain a skeleton decomposition of R_2 we consider a matrix C_2 of size $r_1 n_2 \times r_2$ with r_2 linearly independent columns of R_2 and compute its entries

$$C_2(\alpha_1 i_2; \alpha_2) = R(\alpha_1 i_2, j_3^{(\alpha_2)}, j_4^{(\alpha_2)}, \dots, j_d^{(\alpha_2)})$$

or, in the terms of the initial array,

$$C_2(\alpha_1 i_2; \alpha_2) = A(i_1^{(\alpha_1)}, i_2, j_3^{(\alpha_2)}, j_4^{(\alpha_2)}, \dots, j_d^{(\alpha_2)}),$$

where the indices $i_1^{(\alpha_1)}$ are taken from the set $\mathcal{I}_1$ computed at the previous step. As before, we find the row positions for a quasi-maximal volume submatrix in C_2 and construct a skeleton decomposition of R_2 with the column matrix C_2 and some row matrix with r_2 rows. The latter gives rise then to a new tensor of size $r_2 n_3 \times n_4 \times \cdots \times n_d$, which plays the same role as **R** and for which we keep the same notation in what follows.

The row indices that determine the quasi-maximal volume submatrix in C_2 are picked up from the values of long indices $\alpha_1 i_2$, where α_1 corresponds to "one-dimensional" indices in $\mathcal{I}_1$. Therefore, they can be considered as long indices of the form $i_1^{(\alpha_2)} i_2^{(\alpha_2)}$. This is easily generalized to other unfoldings.

At the kth step we have a subtensor **R** of dimensionality $d-k+1$ and of size $r_{k-1} n_k \times n_{k+1} \times n_d$. This subtensor is defined by a subset of entries of the initial tensor **A** as follows:

$$R(\alpha_{k-1} n_k, n_{k+1}, \dots, n_d) = A(i_1^{(\alpha_{k-1})}, i_2^{(\alpha_{k-1})}, \dots, i_{k-1}^{(\alpha_{k-1})}, i_k, i_{k+1}, \dots, i_d).$$

The first $k-1$ indices which we call *left indices* are taken as $(k-1)$-tuples from the already computed *left index set*

$$\mathcal{I}_{k-1} = [i_s^{(\alpha_{k-1})}], \quad \alpha_{k-1} = 1, \dots, r_{k-1}, \quad s = 1, \dots, k-1.$$

Then we consider the first unfolding $R_k = [R(\alpha_{k-1} i_k; i_{k+1}, \dots, i_d]$ in the current subtensor **R**. As previously, positions of columns containing a submatrix of quasi-maximal volume are supposed to be known and taken from the set of $(d-k)$-tuples

$$\mathcal{J}_k = [j_l^{(\alpha_1)}], \quad \alpha_1 = 1, \dots, r_k, \quad l = k+1, \dots, d,$$

These columns comprise a matrix C_k of size $r_{k-1} n_k \times r_k$ with the entries

$$C_k(\alpha_{k-1} i_k, \alpha_k) = A(i_1^{(\alpha_{k-1})}, i_2^{(\alpha_{k-1})}, \dots, i_{k-1}^{(\alpha_{k-1})}, i_k, j_{k+1}^{(\alpha_k)}, \dots, j_d^{(\alpha_k)}).$$

The next *left index set* $\mathcal{I}_k$ is defined by the row positions of the quasi-maximal volume submatrix in C_k. The new tensor carriage G_k is obtained by reshaping from the matrix C_k postmultiplied by the inverse to that submatrix, absolutely in the same way as on the steps $k=1$ and $k=2$.

Let us sum everything up. If we know the index sets $\mathcal{J}_k$

$$\mathcal{J}_k = [j_s^{(\alpha_k)}], \quad \alpha_s = 1, \dots, r_s, \quad s = k+1, \dots, d,$$

that correspond to the columns containing quasi-maximal volume submatrices in the unfolding matrices A_k, and $r_k = \operatorname{rank} A_k$ are the compression ranks of the TT decomposition that is sought, then the TT decomposition can be recovered by computing

$$n_1 r_1 + n_2 r_1 r_2 + \cdots + n_{d-1} r_{d-2} r_{d-1} + n_d r_{d-1}$$

elements of **A**. If $n_k = n$ and $r_k = r$ then we have to compute only $\mathcal{O}(dnr^2)$ elements. Additionally we need to perform d times the search for a quasi-maximal volume submatrix in matrices of size $nr \times r$ (d calls for the *maxvol* procedure), and this can be done in $\mathcal{O}(dnr^3)$ operations.

It is worth noting that usually the evaluation of a function of d variables takes at least $\mathcal{O}(d)$ operations, so the method would scale quadratically in the dimensionality d, and for really large d it would be useful to write an additional subroutine that computes the subtensor

$$C_k = A(i_1^{(\alpha_{k-1})}, \ldots, i_{k-1}^{(\alpha_{k-1})}, i_k, j_{k+1}^{(\alpha_k)}, \ldots, j_d^{(\alpha_k)})$$

in some fast way.

Along with tensor carriages of the TT decomposition, the *left index sets* $\mathcal{I}_k$ are computed. In comparison with the *right index sets* $\mathcal{J}_k$ assumed to be given, they are not arbitrary but depend on each other. Each of the sets $\mathcal{I}_k$ is of the form

$$\mathcal{I}_k = [i_k^{(\alpha_k)}], \quad \alpha_k = 1, \ldots, k, \quad k = 1, \ldots, d-1,$$

and connected to $\mathcal{I}_{k-1}$ (for $k \geqslant 2$) in the following way:

$$(i_1, i_2, \ldots, i_{k-1}, i_k) \in \mathcal{I}_k \Rightarrow (i_1, i_2, \ldots, i_{k-1}) \in \mathcal{I}_{k-1}, \quad k = 2, \ldots, d-1.$$

We will call such a sequence of index sets a *left-nested sequence*.

A similar definition can be given for *right-nested sequences* of right index sets $\mathcal{J}_k$:

$$(j_{k+1}, j_{k+2}, \ldots, j_d) \in \mathcal{J}_k \Rightarrow (j_{k+2}, \ldots, j_d) \in \mathcal{J}_{k+1}, \quad k = 1, \ldots, d-2.$$

Note, however, that in the above constructions the right-nested property was not imposed on $\mathcal{J}_k$.

We have proved the following theorem.

Theorem 3.1. *Let* $\mathbf{A}$ *be an arbitrary d-dimensional tensor of size* $n_1 \times \cdots \times n_d$ *with compression ranks*

$$r_k = \operatorname{rank} A_k, \quad A_k = A(i_1 i_2 \ldots i_k; i_{k+1} \ldots i_d).$$

Assume that the right index sets

$$\mathcal{J}_k = [j_l^{(\beta_l)}], \quad \beta_l = 1 \ldots, r_k, \; l = k, \ldots, d-1$$

and the left index sets

$$\mathcal{I}_k = [i_l^{(\alpha_l)}], \quad \alpha_l = 1, \ldots, r_k, \; l = 1, \ldots, k-1$$

are given such that the left index sets form a left-nested sequence and the intersection $r_k \times r_k$ *matrices*

$$\widehat{A}_k(\alpha_k, \beta_k) = A(i_1^{(\alpha_k)}, i_2^{(\alpha_k)}, \ldots, i_k^{(\alpha_k)}; j_k^{(\beta_k)}, \ldots, j_d^{(\beta_k)}), \quad \alpha_k, \; \beta_k = 1, \ldots, r_k$$

are all nonsingular.

Then $\mathbf{A}$ *can be recovered from three-dimensional* $r_{k-1} \times n_k \times r_k$ *tensors* $\mathbf{C}_k$ *with the elements*

$$C_k(\alpha_k, i_k, \beta_k) = A(i_1^{(\alpha_k)}, i_2^{(\alpha_k)}, \ldots, i_{k-1}^{(\alpha_k)}, i_k, j_{k+1}^{(\beta_k)}, \ldots, j_d^{(\beta_k)})$$

by the tensor-train interpolation formula

$$A(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_{d-1}} \widehat{C}_1(i_1, \alpha_1) \widehat{C}_2(\alpha_1, i_2, \alpha_2) \cdots \widehat{C}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) \widehat{C}_d(\alpha_{d-1}, i_d),$$

where the tensor carriages $\widehat{\mathbf{C}}_k$ *are obtained from* $\mathbf{C}_k$ *by the following type of scaling*:

$$\widehat{\mathbf{C}}_k = \mathbf{C}_k \times_3 \widehat{A}_k^{-1}, \quad k = 2, \ldots, d-1,$$
$$\widehat{\mathbf{C}}_1 = \mathbf{C}_1 \widehat{A}_1^{-1}, \quad \widehat{\mathbf{C}}_d = \mathbf{C}_d.$$

We can recall that the tensor-by-matrix operation $\mathbf{C}_k \times_3 \widehat{A}^{-1}$ reduces to computation of the product of two matrices as follows:

$$[\widehat{C}_k(\alpha_{k-1} i_k; \alpha_k)] = [C_k(\alpha_{k-1} i_k; \alpha_k)] \widehat{A}_k^{-1}.$$

This interpolation theorem is valid formally only in the exact low-rank case. Note also that the choice of submatrices of quasi-maximal volume was not necessary in the above proof. The only important

thing was that the intersection submatrices are nonsingular. In order to obtain a numerically viable algorithm, all the same, we have to carefully select a *sufficiently good* submatrix among those that are just nonsingular.

As in the matrix case, one can consider skeleton *approximations* in the form of tensor trains instead of exact decompositions. A rigorous error analysis for the possible growth factor for the errors related with the approximation of unfolding matrices is underway and will be reported in future papers, but already now the numerical evidence confirms (see our examples below) that the proposed TT-cross decomposition possesses good stability properties provided that the index sets on which the interpolation is based are chosen in a smart way. We proceed with more details for this issue.

3.2. Towards the practical algorithm

In practice the index sets $\mathcal{J}_k$ are not known in advance, and have to be computed somehow. There are several ways to find them in the matrix case. The one that is most straightforwardly generalized to the multidimensional case is the row-column alternating algorithm [39], which proceeds as follows.

Suppose we have a matrix A (given as a subroutine that computes any prescribed element) and want to find columns and rows that contain a submatrix of sufficiently large volume. To begin with, we take some r columns of A arbitrarily (for example at random) and compute the corresponding column matrix

$$C = A(:, \mathcal{J}).$$

In this matrix C we find a set $\mathcal{I}$ of indices of rows with a submatrix of quasi-maximal volume. Then, *the rows* of A corresponding to the indices from $\mathcal{I}$ are computed. In the corresponding row matrix

$$R = A(\mathcal{I}, :)$$

we find a set $\mathcal{J}$ of indices of columns with a quasi-maximal volume submatrix in R, and so on.

At each iteration step a new intersection submatrix $\widehat{A}$ is obtained. Obviously, its volume is non-decreasing, so we can hope to find a sufficiently good submatrix in a few iteration steps. This is exactly what is observed in practice, but a theoretical analysis is still welcome.

Another issue is that the value of rank r might be not known to us. In the matrix case it may be not very frustrating, but in the tensor case there maybe dozens or hundreds of ranks, and we can only estimate them from above by some number. If the rank is overestimated, we are left with almost-rank-deficient $n \times r$ matrices and the computation of the maximum volume submatrix in C as well as the computation of $C\widehat{A}^{-1}$ becomes an unstable operation.

A simple remedy is available, still. Instead of computing $C\widehat{A}^{-1}$ directly we first compute the QR-decomposition of C:

$$C = QT,$$

with orthogonal Q, compute a quasi-maximal volume submatrix in Q (in exact arithmetics maximum volume submatrices in Q and C coincide, but for cases near to rank-deficiency it may not be true in the finite precision). Denote this submatrix by $\widehat{Q}$. Then

$$C\widehat{A}^{-1} = QTT^{-1}\widehat{Q}^{-1} = Q\widehat{Q}^{-1}.$$

The matrix $Q\widehat{Q}^{-1}$ can be computed in a stable way since it can be proved that $\|\widehat{Q}^{-1}\|$ is bounded from below by a function $t(r, n)$ which grows mildly with n and r [22,23].

So, instead of computations with the initial "column matrices" we perform computation with their Q-factors and do not have to worry about any singularities: the rank can be set to any number not smaller than the "true" rank of the matrix.

An algorithm for the matrix case is summarized in Algorithm 2. In this form it is surely not the fastest possible, but it has a simple structure and the complexity is linear in the matrix sizes.

Algorithm 2. Low rank approximation algorithm.

Require: An $n \times m$ matrix A, rank upper bound r, stopping accuracy parameter δ.
Ensure: r rows with indices $\mathcal{I}$ and r columns indices $\mathcal{J}$ providing that their intersection $\widehat{A} = A(\mathcal{I}, \mathcal{J})$ contains a submatrix of sufficiently large volume.

1: Initialization: $\mathcal{J} = 1 : r, A_0 = 0_{n\times m}, k = 0$
2: **Do**
3: {Row cycle}
4: $R = A(:, \mathcal{J})$
5: Compute the QR decomposition: $R = QT$, Q is $n \times r$.
6: {Quasi-maximal volume submatrix}
7: $\mathcal{I} = \text{maxvol}(Q)$
8: {Column cycle}
9: $C = A(\mathcal{I}, :), C{:=}C^\top, C$ is now $m \times r$
10: Compute the QR decomposition: $C = QT$, Q is $m \times r$.
11: {Quasi-maximal volume submatrix}
12: $\mathcal{J} = \text{maxvol}(Q)$
13: {New approximation}
14: $\widehat{Q} = Q(\mathcal{J}, :), A_{k+1} = A(:, \mathcal{J})(Q(\widehat{Q})^{-1}))^\top$.
15: $k{:=}k + 1$
16: **While** $||A_k - A_{k-1}||_F > \delta ||A_k||_F$.

The same idea applies for tensors. First, we set some upper estimates for the ranks of unfolding matrices and create right index sets $\mathcal{J}_k, k = 2, \ldots, d$ of the required sizes. Then, in chime with the proof of Theorem 3.1, we obtain a tensor-train approximation to the given black-box tensor and, what is even more important for what will follow, the left index sets $\mathcal{I}_k, k = 1, \ldots, d-1$, that form a left-nested sequence by the very construction.

At each step of the algorithm we compute Q-factors of matrices C_k and submatrices of quasi-maximal volume inside Q, just as in the matrix case. Then the left-to-right iteration is performed, that is just the reversion of the previous right-to-left sweep: first we separate the mode d, then $d-1$ and so on. At this "reverse" step, from $\mathcal{I}_k$ we compute the new index sets $\mathcal{J}_k$ and the process is repeated until "convergence". Since at each step not only indices but also the approximation is computed, we can monitor "convergence" of the process by computing the Frobenius-norm distance between two approximations. This can be done very efficiently by an algorithm from [30]. After the iterations are reported to stop, we recompress the obtained approximation by the TT recompression algorithm from [30]. The output is the final TT approximation.

Last and maybe not least, an important point is to check if the ranks were indeed overestimated: if the TT-cross approximation is computed with ranks equal to r but there exists a mode whose rank has not been reduced in the recompression stage, then this is an indication that we underestimated the rank and possibly have to restart the process with an increased rank.

4. Application to high-dimensional integration

The TT-cross algorithm can be considered as an adaptive interpolation algorithm for a multivariate function given on a tensor grid, and it is quite natural to apply it to the problem of high-dimensional integration. Here we will give only the basic scheme for the numerical integration leaving the details for future research.

Suppose we have a function f of d variables and are required to calculate a d-dimensional integral of the form

$$I(f) = \int_{[0,1]^d} f(x_1, x_2, \ldots, x_d) dx_1 dx_2 \ldots dx_d. \tag{10}$$

For large values of the dimensionality d such integrals are usually computed by Monte Carlo or Quasi Monte Carlo methods [37], or by sparse grid approaches [18], or sometimes by using analytical decompositions [8].

Above all, in order to evaluate $I(f)$ we need to construct some quadrature rule. Consider some (supposedly highly accurate) one-dimensional quadrature rule on n points:

$$\int_0^1 g(x)dx \approx \sum_{k=1}^{n} w_k g(x_k).$$

For such quadratures we can take, for example, Gauss quadrature, Clenshaw–Curtis quadrature (with Chebyshev nodes) [17] or some other quadrature that might be well-suited for a particular function g. Then a d-dimensional quadrature rule for $I(f)$ can be constructed straightforwardly as a tensor product of one-dimensional rules:

$$I(f) \approx Q(f) = \sum_{k_1, k_2, \ldots, k_d} f(x_{k_1}, x_{k_2}, \ldots, x_{k_d}) w_{k_1} w_{k_2} \ldots w_{k_d}. \tag{11}$$

The number of function evaluations in this rule grows in d exponentially.

However, let us introduce a black-box tensor $\mathbf{A} = [A(i_1, \ldots, i_d)]$ with the entries

$$A(i_1, i_2, \ldots, i_d) = f(x_{i_1}, x_{i_2}, \ldots, x_{i_d}).$$

This tensor is a virtual object and no memory is needed as yet. Then, if this tensor can be accurately enough approximated by a tensor train by the TT-cross algorithm, then computations can be performed fast even for high values of d and pretty accurate if the function f is sufficiently smooth (sometimes even $n = 5$ gives the machine precision accuracy).

As wee see from (11), the approximate integration rule reduces to the so-called mode contractions (tensor-by-vector multiplications):

$$I(f) \approx Q(f) = \mathbf{A} \times_1 w \times_2 w \times \cdots \times_d w.$$

If $\mathbf{A}$ is in the TT format, then the right-hand side can be computed in $\mathcal{O}(dnr^3)$ operations by the algorithm from [30]. However, a simple modification reduces the cost to $\mathcal{O}(dnr^2)$. Indeed, if $\mathbf{A}$ is given in the TT format with tensor carriages $G_1, \ldots, G_d$, then

$$Q(f) = \sum_{i_1, \ldots, i_d} \sum_{\alpha_1, \ldots, \alpha_{d-1}} G_1(i_1, \alpha_1) \cdots G_d(\alpha_d, i_d) w(i_1) \cdots w(i_d). \tag{12}$$

Evidently, the summation in (12) over spacial indices $i_1, \ldots, i_d$ can be done in parallel. Then it remains to obtain the number $Q(f)$ by successive summations over auxiliary indices $\alpha_1, \ldots, \alpha_{d-1}$. This already gives the claimed complexity. However, this approach would require additional memory of size dr^2 which can be reduced to r^2 if we interlace the summations over spacial and auxiliary indices.

Let us summate first over i_1 and obtain a vector v_1 of size r_1 with elements

$$v_1(\alpha_1) = \sum_{i_1=1}^{n_1} G_1(i_1, \alpha_1) w(i_1), \quad \alpha_1 = 1, \ldots, r_1.$$

Then we summate over α_1, which gives a $n_2 \times r_2$ matrix W_2 with elements

$$W_2(i_2, \alpha_2) = \sum_{\alpha_1=1}^{r_1} v(\alpha_1) G_2(\alpha_1, i_2, \alpha_2), \quad \alpha_2 = 1, \ldots, r_2, \ i_2 = 1, \ldots, n_2.$$

Then we have to summate over i_2 and obtain a vector v_2 of size r_2 with elements

$$v_2(\alpha_2) = \sum_{i_2=1}^{r_2} W_2(i_2, \alpha_2) w(i_2), \quad \alpha_2 = 1, \ldots, r_2.$$

We continue this process (summate over α_2, then over i_3 and so on) until the last mode is reached, then a single scalar product has to be computed.

At each step the summation over α_k is equivalent to a matrix-by-vector product of a $(n_{k+1}r_{k+1}) \times r_k$ matrix by a vector of size r_k (the auxiliary vector v_k is of size r_k), which can be performed in $n_{k+1}r_{k+1}r_k$ operations, and the summation over i_k is equivalent to a matrix-by-vector product of a $n_k \times r_k$ matrix by a vector of size n_k, which costs merely $\mathcal{O}(n_k r_k)$ operations. If all mode sizes are equal to n and all ranks are equal to r, then the total cost of the algorithm is $\mathcal{O}(dnr^2)$. In our integration scheme the weights w are the same for all modes, but there could be as well schemes with mode-dependent weights.

The described above contraction procedure is summarized in Algorithm 3.

Algorithm 3. Fast TT contraction algorithm

Require: A tensor $\mathbf{A}$ of size $n_1 \times n_2 \times \cdots \times n_d$ in the TT format with tensor carriages $\mathbf{G}_k$; vectors w_k of size $n_k, k = 1, \ldots, d$.
Ensure: $I = \mathbf{A} \times_1 w_1 \times_2 w_2 \times \cdots \times_d w_d$.

1: $v := \mathbf{G}_1 \times_1 w_1$.
2: **for** $k = 2$ **to** d **do**
3: {Summate over α_{k-1}}
4: $W := \mathbf{G}_k \times_1 v$, W is $n_k \times r_k$.
5: {Summate over i_k}
6: $v := W^\top w_k$, v is of size r_k.
7: **end for**
8: $I = v$.

5. Numerical experiments

The proposed above algorithms were implemented as a part of the TT Toolbox [29] in MATLAB, the computations were performed on an Intel Core 2 notebook with 2.26 GHz clock and 3 GB of RAM.

5.1. Random canonical tensor

As a first test, the "sanity check" was performed. A random tensor of canonical rank r was generated, by first creating random factor matrices $U_1, U_2, \ldots, U_d$ of size $n \times r$. The elements were drawn from the standard normal distribution. Then the TT approximation was computed by TT-cross algorithm. Since a TT approximation from the canonical representation of a tensor can be also computed directly [30], we can compare the two results and check the accuracy.

Computation of a single element in the canonical format takes $\mathcal{O}(dr)$ operations, so the expected cost of the TT-cross approximation method in this situation is

$$\mathcal{O}(d^2nr^2 + dnr^3).$$

The ranks computed from the TT-cross approximation coincided with the "true" compression ranks of the tensor. A sample of numerical results is presented in Table 1.

5.2. Hilbert tensor

The second example is the Hilbert tensor with elements

$$A(i_1, i_2, \ldots, i_d) = \frac{1}{i_1 + i_2 + \ldots + i_d}.$$

Table 1
Timings for the compression of a random canonical-format tensor with rank r to the TT format using TT-cross algorithm, $n = 32, r = 10$.

d	Time	Iterations	Final residual
5	2.48	3	1×10^{-15}
10	13.52	3	2×10^{-15}
20	48.23	3	4×10^{-15}
40	178.22	3	6×10^{-15}
80	688.12	3	2×10^{-14}

Table 2
Timings for the compression of the Hilbert tensor with the rank bound $r_{\max}$ to the TT format by the TT-cross algorithm, $n = 32, d = 60$.

$r_{\max}$	Time	Iterations	Final relative residual
2	1.37	5	1.897278e+00
3	4.22	7	5.949094e−02
4	7.19	7	2.226874e−02
5	15.42	9	2.706828e−03
6	21.82	9	1.782433e−04
7	29.62	9	2.151107e−05
8	38.12	9	4.650634e−06
9	48.97	9	5.233465e−07
10	59.14	9	6.552869e−08
11	72.14	9	7.915633e−09
12	75.27	8	2.814507e−09

For this kind of tensors it is known that a nice low-canonical-rank approximation exists (cf. [7]) and can be computed in various ways.

To test the TT-cross algorithm we first computed the TT approximation in $d = 60$ dimensions with overestimated rank $r_{\max} = 50$ and took that as a reference solution so that we need not compute all n^d entries of d-dimensional tensor to reliably check the accuracy when taking smaller values of rank.

The timings are presented in Table 2. Ranks are given up to 12, since further increase of rank did not result in decrease of observed accuracy. The source of this stagnation should be investigated more carefully and we think that it may be caused by the properties of the alternating row-column *maxvol*-based algorithm for computing a low rank approximation (for some cases the accuracy can not be much smaller than the square root of the machine precision). This will be a subject of future work.

5.3. High-dimensional integration

In this subsection we present some results for computing high-dimensional integrals by tensor product quadratures and the TT-cross algorithm.

The first integration example is the sine example as follows:

$$f(x_1, x_2, \ldots, x_d) = \sin(x_1 + x_2 + \cdots + x_d).$$

We can prove that compression ranks in this case are equal to 2, whereas only a rank-d canonical representation with real numbers is known. Of course, over the complex field the canonical rank is 2 due to the identity

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$

and that gives an exact value of the integral:

$$I(d) = \operatorname{Im} \int_{[0,1]^d} e^{i(x_1+x_2+\cdots+x_d)} dx_1 dx_2 \cdots dx_d = \operatorname{Im} \left(\left(\frac{e^i - 1}{i} \right)^d \right).$$

Table 3
Error and timings for the sine example.

d	I	Relative error	Time
10	−6.299353e−01	1.409952e−15	0.14
100	−3.926795e−03	2.915654e−13	0.77
500	−7.287664e−10	2.370536e−12	4.64
1000	−2.637513e−19	3.482065e−11	11.60
2000	2.628834e−37	8.905594e−12	33.05
4000	9.400335e−74	2.284085e−10	105.49

Table 4
Error and timings for the TT-cross approximate computation of the d-dimensional integral (13).

$r_{\max}$	Relative error	Time
2	1.747414e−01	1.76
4	2.823821e−03	11.52
8	4.178328e−05	42.76
10	3.875489e−07	66.28
12	2.560370e−07	94.39
14	4.922604e−08	127.60
16	9.789895e−10	167.02
18	1.166096e−10	211.09
20	2.706435e−11	260.13

As a one-dimensional quadrature we take Clenshaw–Curtis quadrature with $n = 11$ points. The TT-cross method computes approximation with all ranks equal to 2 with almost machine precision. Timings are presented in Table 3.

Since all the compression ranks are all equal to 2, the computations are much faster than in the previous example. Note that the very small values of the integral are computed with high *relative* accuracy. This completely confirms good stability properties of the TT decomposition.

For the second example we take the following integral:

$$I(d) = \int_{[0,1]^d} \sqrt{x_1^2 + x_2^2 + \cdots x_d^2}\, dx_1 dx_2 \cdots dx_d. \tag{13}$$

No closed form solution is known for $I(d)$.

As a quadrature rule, we took a Clenshaw–Curtis rule with 41 nodes, computed a TT-cross approximation with $r_{\max} = 32$ and made that to serve as our reference solution. However, Table 4 shows that 41 nodes is much more than enough in this case. The results are presented in $d = 100$ dimensions with 11 nodes for the one-dimensional quadrature rule; the rank bound $r_{\max}$ for TT compression ranks is varying. Time is only given for the TT-cross approximation stage since the time to compute the TT contraction is negligible.

6. Conclusions and future work

In this paper we presented a multidimensional generalization of the skeleton decomposition to the tensor case. It is as well a new interpolation formula in the form of tensor trains on the entries of some tensor-train cross (Theorem 3.1). The formula itself and the TT-cross interpolation algorithm do not suffer from the curse of dimensionality.

The TT-cross formula represents a tensor exactly if its auxiliary unfolding matrices are of low rank. The latter is always the case when a tensor has a low canonical rank. In other cases this formula can represent a tensor approximately. For general TT approximations, in this regard we prove that with the best possible accuracies ε_k for approximations of predetermined ranks to the unfolding matrices, there exists a TT approximation with error bounded from above by $\sqrt{\sum_{k=1}^{d-1} \varepsilon_k^2}$ (Theorem 2.2). It immediately follows that if ε is the best possible accuracy for TT approximations with fixed compression rank

bounds, then the constructions of Theorem 2.2 provide a quasi-optimal TT approximation with the error bounded from above by $\sqrt{d-1}\,\varepsilon$.

A simple variant of the TT-cross approximation method is proposed in the case when a tensor is given as a black box by a procedure for computation of any required element. This method allowed us to obtain TT approximations to some functional tensors in really many dimensions in a moderate time (up to a few minutes) on a notebook. Then, as soon as we have tensors in the TT format, we are able to perform many basic operations with these tensors in a very efficient way [31,30,35,36].

There is still a lot of work to be done. From algorithmical point of view, the alternating row-column algorithm for low-rank approximation of matrices is clearly suboptimal and likely to require more tensor elements than necessary. This depends on the number of row-column iterations, and a factor of 3–5 (at least) can be saved by using cross approximation techniques which rely on successive rank-one approximations to a matrix. It is not quite clear at this moment how this approach can be extended to the d-dimensional case, and maybe some other method should be used. An upper bound on compression ranks of the tensor should be given on input, and how to avoid this is again a topic for future research (if the ranks are seriously overestimated, then the number of elements to be computed may be much larger than required).

From theoretical viewpoint, formulations and proofs of existence theorems in the approximate case should be obtained: if a tensor admits a TT approximation with accuracy ε, then we want to be assured that it admits a tensor-train skeleton (TT-cross) decomposition with accuracy $c(n,r)\varepsilon$ and a reasonable behavior of the deterioration factor $c(n,r)$. It is important to investigate what is the worst case and "average case" dependence of this factor $c(n,r)$ on the mode sizes and compression ranks.

And at last, we envisage many promising applications of the TT-cross algorithm to various problems, including high-dimensional integration and multivariate function approximation (for example, in the cases when the solution depends on several parameters and each function evaluation requires that some partial differential equations are solved). Moreover, the TT-cross method can be applied to matrix problems by recognizing high-dimensional tensors inside matrices. The latter recently led to the so-called TTM decomposition [33,32], and in many interesting cases it could result in matrix algorithms with a logarithmic complexity in the size. Here, the multidimensional TT-cross algorithm can be used as a substitute for the one used in the TTM approach. Preliminary experiments for two-dimensional and three-dimensional model problems suggest that such an approach can be efficient.

All in all, the presented interpolation method for high-dimensional tensors is given with a complete proof in the exact case, and numerical experiments confirm that it is a very efficient approximation tool for black-box tensors. We hope that more possible applications and algorithmical improvements will be presented in forthcoming papers.

References

[1] R. Badeau, R. Boyer, Fast multilinear singular value decomposition for structured tensors, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 1008–1021.
[2] B.W. Bader, T.G. Kolda, Efficient MATLAB computations with sparse and factored tensors, SIAM J. Sci. Comput. 30 (1) (2007) 205–231.
[3] B.W. Bader, T.G. Kolda, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009).
[4] M. Bebendorf, Approximation of boundary element matrices, Numer. Math. 86 (4) (2000) 565–589.
[5] G. Beylkin, M.J. Mohlenkamp, Numerical operator calculus in higher dimensions, Proc. Nat. Acad. Sci. USA 99 (16) (2002) 10246–10251.
[6] G. Beylkin, M.J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, SIAM J. Sci. Comput. 26 (6) (2005) 2133–2159.
[7] D. Braess, W. Hackbusch, Approximation of $1/x$ by exponential sums in $[1,\infty)$, IMA J. Numer. Anal. 25 (4) (2005) 685–697.
[8] D. Braess, W. Hackbusch, On the efficient computation of high-dimensional integrals and the approximation by exponential sums, Preprint 3, MPI MIS, Leipzig, 2009.
[9] J.D. Carroll, J. J Chang, Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition, Psychometrika 35 (1970) 283–319.
[10] S.R. Chinnamsetty, M. Espig, W. Hackbusch, B.N. Khoromskij, H.J. Flad, Optimal Kronecker tensor-product approximation in quantum chemistry, J. Chem. Phys. 127 (2007) 84–110.
[11] L. de Lathauwer, B. de Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (2000) 1253–1278.
[12] L. de Lathauwer, B. de Moor, J. Vandewalle, On best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of high-order tensors, SIAM J. Matrix Anal. Appl. 21 (2000) 1324–1342.

[13] M. Espig, Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen, Ph.D. Thesis, Leipzig, 2007.
[14] M. Espig, L. Grasedick, W. Hackbusch, Black box low tensor rank approximation using fibre-crosses, Preprint 60, MPI MIS, Leipzig, 2008.
[15] J.M. Ford, E.E. Tyrtyshnikov, Combining Kronecker product approximation with discrete wavelet transforms to solve dense function-related systems, SIAM J. Sci. Comput. 25 (3) (2003) 961–981.
[16] F.R. Gantmacher, Theory of Matrices, Chelsea, New York, 1959.
[17] W. Gautschi, Numerical Analysis: An Introduction, Birkhauser, Boston, 1997.
[18] Th. Gerstner, M. Griebel, Numerical integration using sparse grids, Numer. Algor. 18 (3–4) (1998) 209–232.
[19] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., John Hopkins Univ. Press, 1996.
[20] S.A. Goreinov, On cross approximation of multi-index array, Doklady Math. 420 (4) (2008) 1–3.
[21] S.A. Goreinov, I.V. Oseledets, D.V. Savostyanov, E.E. Tyrtyshnikov, N.L. Zamarashkin, How to find a good submatrix, Research Report 08-10, ICM HKBU, Kowloon Tong, Hong Kong, 2008.
[22] S.A. Goreinov, E.E. Tyrtyshikov, N.L. Zamarashkin, Pseudo-skeleton approximations of matrices, Rep. Russian Acad. Sci. 342 (2) (1995) 151–152.
[23] S.A. Goreinov, E.E. Tyrtyshikov, N.L. Zamarashkin, A theory of pseudo-skeleton approximations, Linear Algebra Appl. 261 (1997) 1–21.
[24] S.A. Goreinov, E.E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, Contemporary Math. 208 (2001) 47–51.
[25] L. Grasedyck, Existence and computation of low Kronecker-rank approximations for large systems in tensor product structure, Computing 72 (2004) 247–265.
[26] W. Hackbush, B.N. Khoromskij, E.E. Tyrtyshnikov, Hierarchical Kronecker tensor-product approximations, J. Numer. Math. 13 (2005) 119–156.
[27] R.A. Harshman, Foundations of the Parafac procedure: models and conditions for an explanatory multimodal factor analysis, UCLA Working Papers Phonet. 16 (1970) 1–84.
[28] B.N. Khoromskij, On tensor approximation of Green iterations for Kohn–Sham equations, Comput. Visual. Sci. 11 (4–6) (2008) 259–271.
[29] I.V. Oseledets, TT Toolbox 1.0: Fast multidimensional array operations in MATLAB, Preprint 2009-06, INM RAS, August 2009.
[30] I.V. Oseledets, Compact matrix form of the d-dimensional tensor decomposition, Preprint 2009-01, INM RAS, March 2009.
[31] I.V. Oseledets, On a new tensor decomposition, Doklady Math. (2009).
[32] I.V. Oseledets, On the approximation of matrices with logarithmical number of parameters, Doklady Math. (2009).
[33] I.V. Oseledets, Tensors inside of matrices give logarithmic complexity, SIAM J. Matrix Anal. Appl. (2009).
[34] I.V. Oseledets, D.V. Savostianov, E.E. Tyrtyshnikov, Tucker dimensionality reduction of three-dimensional arrays in linear time, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 939–956.
[35] I.V. Oseledets, E.E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, SIAM J. Sci. Comput. (2009).
[36] I.V. Oseledets, E.E. Tyrtyshnikov, Recursive decomposition of multidimensional tensors, Doklady Math. (2009).
[37] I.M. Sobol, Quasi-Monte Carlo methods, Prog. Nucl. Energy 24 (1990) 55–61.
[38] L.R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311.
[39] E.E. Tyrtyshnikov, Incomplete cross approximation in the mosaic-skeleton method, Computing 64 (4) (2000) 367–380.
[40] E.E. Tyrtyshnikov, Tensor approximations of matrices generated by asymptotically smooth functions, Sbornik: Math. 194 (50-6) (2003) 941–954.