SIAM J. SCI. COMPUT.
Vol. 31, No. 5, pp. 3744–3759

BREAKING THE CURSE OF DIMENSIONALITY, OR HOW TO USE SVD IN MANY DIMENSIONS*

I. V. OSELEDETS† AND E. E. TYRTYSHNIKOV†

Abstract. For d-dimensional tensors with possibly large $d > 3$, an hierarchical data structure, called the *Tree-Tucker* format, is presented as an alternative to the canonical decomposition. It has asymptotically the same (and often even smaller) number of representation parameters and viable stability properties. The approach involves a recursive construction described by a tree with the leafs corresponding to the Tucker decompositions of three-dimensional tensors, and is based on a sequence of SVDs for the recursively obtained unfolding matrices and on the auxiliary dimensions added to the initial "spatial" dimensions. It is shown how this format can be applied to the problem of multidimensional convolution. Convincing numerical examples are given.

Key words. Tree-Tucker, canonical decomposition, Tucker decomposition, curse of dimensionality

AMS subject classifications. 15A12, 65F10, 65F15

DOI. 10.1137/090748330

1. Introduction.

Numerical solution of multidimensional equations and calculation of multivariate integrals are often difficult or impossible due to the *curse of dimensionality*: the amount of memory and the number of operations required to solve such problems grow exponentially in the dimension d, i.e., the dependence has the form ρ^d with $\rho > 1$. For some very successful algorithms (for example, sparse grid methods) the constant ρ may be rather small, $\rho \leq 10$. However, even for $\rho = 5$ the number of dimensions in practice is naturally bounded by the memory restrictions, probably $d \leq 15$ and hardly much higher. And what should we do if we strongly need to deal with at least $d = 100$?

In order to do that, we have to construct some low-parametric representation to every object involved (operators, matrices, functions, vectors) and work only with this low-parametric representation. A nice idea is to use *separated representation* [3, 4, 10, 11, 15] which is now quite popular: for a given multivariate function f of d variables $x_1, x_2, x_3, \ldots, x_d$ we approximate it by a sum of functions with separated variables:

$$f(x_1, x_2, \ldots, x_d) \approx \sum_{\alpha=1}^{r} f_\alpha^{(1)}(x_1) f_\alpha^{(2)}(x_2) \cdot \ldots \cdot f_\alpha^{(d)}(x_d). \tag{1.1}$$

This kind of approximation exists for many important cases, for example, for asymptotically smooth functions [17] such as the kernels of important integral operators. It is important for us that the continuous decomposition (1.1) has a direct discrete analogue for matrices and vectors. Consider the values of the function f on some tensor grid in $\mathbb{R}^d$ with n nodes in each direction. Putting these values together in a natural order, we obtain a vector v of size n^d. This vector can be treated as a d-dimensional

*Received by the editors January 30, 2009; accepted for publication (in revised form) August 5, 2009; published electronically October 7, 2009. This research was supported by RFBR grants 08-01-00115 and 09-01-00565 and RFBR/DFG grant 09-01-91332.
http://www.siam.org/journals/sisc/31-5/74833.html
†Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8 IVM RAN, Moscow 119991, Russia (ivan@bach.inm.ras.ru, tee@bach.inm.ras.ru).

array $\mathcal{V}$ with the property

$$\mathcal{V} \approx \sum_{\alpha=1}^{r} u_\alpha^{(1)} \otimes u_\alpha^{(2)} \otimes \ldots \cdot \otimes u_\alpha^{(d)}, \tag{1.2}$$

where the size-n vectors $u_\alpha^{(i)}$ consist of the values of one-dimensional functions $f^{(i)}$ on the considered grid, and $\otimes$ is the tensor (outer) product of two vectors.

The right-hand side of (1.2) in multilinear algebra is called *canonical decomposition*, and many authors refer to (1.2) as *CANDECOMP/PARAFAC* model [2, 9, 5], where PARAFAC stands for *parallel factors*. The number of parameters in the separated representation in (1.2) is $\mathcal{O}(drn)$, obviously much less than n^d entries in the full vector. That entirely removes the dimension d from the exponent if r does not depend on d. Literally the same decomposition may be applied to matrices (considered as discretizations of a differential or integral operator acting on a tensor domain):

$$A \approx \sum_{\alpha=1}^{r} U_\alpha^{(1)} \otimes U_\alpha^{(2)} \otimes \ldots \cdot \otimes U_\alpha^{(d)}, \tag{1.3}$$

where A is a $n^d \times n^d$ matrix and $U^{(i)}$ are $n \times n$ matrices. The number r in the above approximations is called *tensor rank* or *canonical rank* and plays a crucial role in the complexity estimates. Note, that the problem (1.3) is in fact equivalent to the problem (1.2). After a special "reshuffling" of the elements of the matrix A it is reduced to the computation of the canonical decomposition of an $n^2 \times n^2 \times, \ldots, \times n^2$ tensor. In the two-dimensional case this observation probably belongs to Van Loan and Pitsianis [18] and the generalization to the d-dimensional case is easy (it was considered, for example, in [17, 12]).

So, the tensor decomposition gives a convenient and quite general way to represent multidimensional arrays. All the same, we need efficient numerical procedures for the following tasks:

- compute a tensor format representation;
- perform basic linear algebra operations in that format.

The second part is simple in the tensor format if we admit that the rank can grow. However, numerical computation of tensor decompositions in many dimensions is not easy even for $d = 3$, the tensor approximation problem with a prescribed rank is known to be unstable (for example, the distance between a tensor and a set of tensors with a canonical rank r is zero, but there is no canonical decomposition with rank r [6]), and a robust procedure for low-rank tensor approximation in many dimensions is in effect not available. By "robust" we mean a deterministic procedure which is guaranteed to find the decomposition when it exists. Even when the tensor is given in some canonical format with rank R and we want to approximate it by a tensor of smaller rank $r < R$, there are no robust algorithms.

Everything is simple only in two dimensions. In this case (1.2) is equivalent to the so-called skeleton (dyadic) decomposition [7] and related with the celebrated *singular value decomposition* (SVD) [8]. Indeed, in the elementwise form

$$v_{i_1 i_2} \approx \sum_{\alpha=1}^{r} u_{i_1 \alpha}^{(1)} u_{i_2 \alpha}^{(2)},$$

where the right-hand side pertains to a matrix of rank r, and the best rank-r approximation in the Frobenius (as well spectral or any unitarily invariant) norm is given by

the truncated singular value decomposition. The best rank-r approximation exists and can be computed by robust numerical procedures such as SVD or fast rank-revealing methods (e.g., cross-approximation algorithm [16]). The decomposition itself can be quite straightforwardly generalized (there are other generalization as well, [2]) to d dimensions as the tensor (canonical) decomposition, but the robust numerical procedures like the SVD are not, neither the very stability properties are not generalized. Another way to generalize the SVD is the *Tucker decomposition* [14, 10, 11]. It reads

$$v_{i_1 i_2 \dots i_d} \approx \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \cdots \sum_{\alpha_d=1}^{r_d} g_{\alpha_1 \alpha_2 \dots \alpha_d} u^{(1)}_{i_1 \alpha_1} u^{(2)}_{i_2 \alpha_2} \dots u^{(d)}_{i_d \alpha_d}.$$

For simplicity we assume that $r_1 = r_2 = \cdots r_d = r$. Then the Tucker approximation is defined by only $nrd + r^d$ parameters while the initial array has n^d entries. The compression rate is, of course, tremendous and, more than that, an almost optimal (in Frobenius norm) Tucker approximation (sometimes called HOSVD (High-order SVD)) which can be computed via several SVDs for auxiliary matrices [10, 11, 12]. It can be proved [11] that HOSVD increases the approximation error at most by $\sqrt{d}$. We are interested only in the cases when a good approximation exists (say, with accuracy $\delta = 10^{-6}$), and therefore we are not interested in computing the optimal approximation in the Frobenius norm. It is worth noting that for small dimensions (especially for $d = 3$) the Tucker decomposition can easily replace[1] the canonical decomposition for many purposes, since the storage for the core ($\mathcal{O}(r^3)$) is negligible if n is large and all operations can be performed in the Tucker, not canonical, format. Nevertheless, for large dimensions the Tucker decomposition is unapplicable since the exponential dependence still remains.

So here we come to the main question of this paper.

Can we find an approximation format for d-dimensional arrays that is free from exponential dependence on d and allows us to perform effective and efficient linear algebra operations by using some SVD-type approach?

In this paper we present such a format and prove that the existence of a certain canonical decomposition implies that the new format representation/approximation contains asymptotically the same number of parameters. We also show how to construct the new representation and how to work with it.

2. Notations. First, let us give a brief summary of notations that we will use. We can treat a d-dimensional array $\mathcal{A} = [a_{i_1 i_2 \dots i_d}]$ as an array of a smaller dimension with some indices merged. This is commonly referred to as *unfolding* or *matricization* of a tensor; see [2]. Split the set of all index variables in two disjoint subsets

$$\{i_1, \dots, i_d\} = \{i'_1, \dots, i'_{d_1}\} \cup \{i''_1, \dots, i''_{d_2}\}, \quad d = d_1 + d_2,$$

and introduce two multiindices

$$\mathcal{I}_1 = (i'_1, \dots, i'_{d_1}), \quad \mathcal{I}_2 = (i''_1, \dots, i''_{d_2}).$$

Each multiindex can be considered as one "long index" (with some appropriate ordering of all possible values). Then, we can view $\mathcal{A}$ as a matrix B with the entries

$$b_{\mathcal{I}_1 \mathcal{I}_2} = a_{i_1 \dots i_d}.$$

[1] There are applications where data follow the trilinear model, and canonical decomposition has "physical meaning" and canonical factors themselves need to be computed. However, for compression purposes and basic linear algebra computations for $d = 3$, Tucker decomposition is sufficient.

The indices in $\mathcal{I}_1$ and $\mathcal{I}_2$ are not necessarily contiguous, neither required to keep the original order. For example, for a four dimensional array $\mathcal{A} = [a_{i_1 i_2 i_3 i_4}]$ we can take

$$\mathcal{I}_1 = (i_4 i_1), \quad \mathcal{I}_2 = (i_2 i_3),$$

then the elements of the matrix $B = [b_{\mathcal{I}_1 \mathcal{I}_2}]$ are defined as

$$b_{\mathcal{I}_1 \mathcal{I}_2} = b_{(i_4 i_1)(i_2 i_3)} = a_{i_1 i_2 i_3 i_4}.$$

To avoid sprouting new letters, let us agree to keep the same letter for the new arrays assuming that

$$a_{\mathcal{I}_1 \mathcal{I}_2} = b_{\mathcal{I}_1 \mathcal{I}_2}.$$

This notation implicitly includes permutation of dimensions and the merging of them into "long dimensions" (it may look somewhat ambiguous, but the meaning is always clear from the context). Also, instead of saying "array $[a_{i_1 \dots i_d}]$ with the entries $a_{i_1 \dots i_d}$," let us allow ourselves to say simply "array $a_{i_1 \dots i_d}$."

Of course, we can also split the whole set of indices into three, four, or more parts (if necessary). If we take $\mathcal{I} = (i_1, \dots, i_d)$, then $\mathcal{A}$ can be viewed as a vector with the entries $a_{\mathcal{I}}$.

We will call the matrix $[a_{\mathcal{I}_1 \mathcal{I}_2}]$ an *unfolding* along $\mathcal{I}_1$ and $\mathcal{I}_2$, since it generalizes the notion of unfoldings for three-dimensional arrays, when a data cube a_{ijk} of size $n \times n \times n$ is "unfolded" by stacking its $n \times n$ *slices* as rows into a $n \times n^2$ matrix. In three dimensions there are only three different unfoldings, in higher dimensions there are much more of them.

For a canonical decomposition of $\mathcal{A}$ we will also use the notation

$$\mathcal{A} = \sum_{\alpha=1}^{r} u_\alpha^{(1)} \otimes u_\alpha^{(2)} \otimes \ldots \cdot \otimes u_\alpha^{(d)} = (U_1, U_2, \dots, U_d),$$

where $U_1 = [u_\alpha^{(1)}]$, $\dots$, $U_d = [u_\alpha^{(d)}]$ are assumed to be matrices with r columns, called the *canonical factors*.

Throughout the paper we use a tensor-by-matrix multiplication referred to as the mode-k contraction or mode-k multiplication or k-mode product [1, 10, 11]. Given an array (tensor) $\mathcal{A} = [a_{i_1 \dots i_d}]$ and a matrix $U = [u_{i_k \alpha}]$, we define the mode-k multiplication result as a new tensor $\mathcal{B} = [b_{i_1 \dots \alpha \dots i_d}]$ (α is on the kth place) obtained by the convolution over the kth axis:

$$b_{i_1 \dots \alpha \dots i_d} = \sum_{\mathbf{i}_k=1}^{n} a_{i_1 \dots i_k \dots i_d} u_{i_k \alpha}.$$

We denote this operation as follows:

$$\mathcal{B} = \mathcal{A} \times_k U.$$

3. Two basic lemmas. The unfolding matrices of a tensor (all of them) have an important connection with any canonical decomposition for the same tensor. Above all, the following simple lemma is valid.

LEMMA 3.1. If a tensor $\mathcal{A} = [a_{i_1 \dots i_d}]$ has a canonical decomposition of rank r, then for any unfolding matrix $B = [a_{\mathcal{I}_1 \mathcal{I}_2}]$

$$\operatorname{rank} B \ \leq \ r.$$

Proof. From the canonical decomposition it immediately follows that

$$a_{\mathcal{I}_1\mathcal{I}_2} = a_{i_1 i_2 \ldots i_d} = \sum_{\alpha=1}^{r} u^{(1)}_{i_1\alpha} u^{(2)}_{i_2\alpha} \cdot \ldots \cdot u^{(d)}_{i_d\alpha} = \sum_{\alpha=1}^{r} \mathcal{U}_{\mathcal{I}_1\alpha} \mathcal{V}_{\mathcal{I}_2\alpha}$$

with some properly defined tensors $\mathcal{U}_{\mathcal{I}_1\alpha}$ and $\mathcal{V}_{\mathcal{I}_2\alpha}$, which consist of the columnwise Kronecker product of the corresponding set of vectors $u_{i\alpha}$. Therefore,

$$B = UV^\top, \quad U = [\mathcal{U}_{\mathcal{I}_1\alpha}], \quad V = [\mathcal{V}_{\mathcal{I}_2\alpha}], \tag{3.1}$$

the matrices U and V being of sizes $\#\mathcal{I}_1 \times r$ and $\#\mathcal{I}_2 \times r$, respectively.[2] ∎

Although the proved claim is that the rank of any unfolding matrix is not larger than r, we observe in practice that the rank can be significantly smaller than r. It happens when either U or V are not of full column rank and the examples illustrating this will be given in the numerical examples section. The representation of $\mathcal{A}$ as a low-rank matrix reduces the number of parameters only to $2rn^{d/2}$, which is still exponential in d.

However, the following lemma states that U and V in (3.1) possess a useful additional structure.

LEMMA 3.2. *Assume that a tensor $\mathcal{A} = [a_{i_1\ldots i_d}]$ has a canonical representation of rank r and an unfolding matrix $B = [a_{\mathcal{I}_1\mathcal{I}_2}]$ is defined by the "long indices" $\mathcal{I}_1$ and $\mathcal{I}_2$ containing d_1 and d_2 initial indices, respectively. Consider an arbitrary decomposition of B of the form*

$$B = UV^\top, \quad U = [\mathcal{U}_{\mathcal{I}_1\alpha}], \quad V = [\mathcal{V}_{\mathcal{I}_2\alpha}],$$

where U and V have $\widehat{r} = \operatorname{rank} B$ columns. Then $\mathcal{U}_{\mathcal{I}_1\alpha}$ and $\mathcal{V}_{\mathcal{I}_2\alpha}$ can be regarded as tensors of dimension $d_1 + 1$ and $d_2 + 1$, respectively, and for both tensors there exist canonical decompositions of rank not higher than r.

Proof. We can express U as a function of V as follows:

$$U = BV(V^\top V)^{-1}.$$

Let $(V^\top V)^{-1} = [g_{\alpha\beta}]$. Then, using the above identity and our multiindex notation, we conclude that

$$\mathcal{U}_{\mathcal{I}_1\beta} = \sum_{\alpha=1}^{\widehat{r}} \sum_{\mathcal{I}_2} a_{\mathcal{I}_1\mathcal{I}_2} \mathcal{V}_{\mathcal{I}_2\alpha} g_{\alpha\beta},$$

or using the canonical representation for the tensor $\mathcal{A}$

$$\mathcal{U}_{\mathcal{I}_1\beta} = \sum_{\mathcal{I}_2} \sum_{\alpha=1}^{\widehat{r}} \sum_{\gamma=1}^{r} \left(\prod_{k=1}^{d} u^{(k)}_{i_k\gamma} \right) V_{\mathcal{I}_2\gamma} g_{\alpha\beta}.$$

Now we can sum over all indices in $\mathcal{I}_2$, and this summation gives us a matrix $w_{\gamma\alpha}$:

$$w_{\gamma\alpha} = \sum_{\mathcal{I}_2} V_{\mathcal{I}_2\gamma} \prod_{k:i_k\in\mathcal{I}_2} u^{(k)}_{i_k\alpha},$$

[2] By $\#\mathcal{I}$ we denote the number of all possible values for the "long index" $\mathcal{I}$. For example, for $\mathcal{I} = \{i_1 i_2 i_3\}$ with n_1, n_2, n_3 possible values for i_1, i_2, i_3, respectively, we obtain $\#\mathcal{I} = n_1 n_2 n_3$.

and then the summation over α gives us a new matrix Z with elements

$$z_{\beta\gamma} = \sum_{\alpha=1}^{\widehat{r}} w_{\gamma\alpha} g_{\alpha\beta}.$$

With the help of it $\mathcal{U}$ is now represented in the canonical format:

$$\mathcal{U}_{\mathcal{I}_1\beta} = \sum_{\gamma=1}^{r} \left(\prod_{i_k \in \mathcal{I}_1} u^{(k)}_{i_k\gamma} \right) z_{\beta\gamma}.$$

The proof for V is similar. ∎

Note that the size of the extra dimension for the tensors $\mathcal{U}$ and $\mathcal{V}$ is equal to $\widehat{r} = \operatorname{rank} B$ and, by Lemma 3.1, does not exceed r.

4. Tree-Tucker format. Lemma 3.2 is the key for what follows. It shows that a d-dimensional array $\mathcal{A}$ of canonical rank r can be completely defined through the two arrays of orders $d_1 + 1$ and $d_2 + 1$, respectively, and of canonical rank not higher than r. In fact, if we compute the $UV^\top$ decomposition for B, then we are able to compute the corresponding canonical decompositions for $\mathcal{U}$ and $\mathcal{V}$ (which are the tensor equivalents for matrices U and V). Moreover, the size of the extra dimension is $\widehat{r} = \operatorname{rank} B \le r$ and, as we later observe in a few examples, can be much smaller than r. From now on, the extra, "nonspatial" dimension is referred to as *auxiliary dimension* and $\widehat{r}$ as *splitting rank* of $\mathcal{A}$.

Since $\mathcal{U}$ and $\mathcal{V}$ possess rank-r canonical representations, we can further split them recursively until the number of dimensions becomes small. For example, for $d = 9$ we can take up $d_1 = 5$ and $d_2 = 4$. Then an array of dimension 9 gets represented via two arrays of dimensions 6 and 5. The six-dimensional array can be further replaced by two four-dimensional arrays, and then each of them is reduced to two three-dimensional arrays. Here we have to stop, since the splitting of three-dimensional arrays by our method keeps one of the dimensions equal to 3. For a five-dimensional array we have the splitting $5 = 3 + 2$ and come up with two arrays of dimensions 4 and 3. The four-dimensional array is replaced with the two of dimension 3 and, as we already noted, three-dimensional arrays are not split. At this point we complete the recursion. The initial nine-dimensional array $\mathcal{A}$ is now represented via $4 + 1 + 2 = 7$ new three-dimensional arrays with canonical decompositions of rank r. Thus, we can store only canonical representation parameters for 7 arrays for dimension 3. The process is illustrated by the tree in Figure 4.1.

Moreover, before we do all that let us do the following. Use the canonical format for the tensor $\mathcal{A}$ and find its Tucker factors (this can be done in a fast way, we will show the details later on). We need nrd memory cells to store them, and since then we go ahead with the Tucker core array of dimension d and size $r \times r \times \ldots \times r$. Thus, the representation of a nine-dimensional array in our scheme requires $9nr + 7r^3$ parameters, the latter term being the memory required to store 7 three-dimensional arrays that appear on the leaf nodes of the tree.

The recursive data structure presented above will be called the *Tree-Tucker format*, or simply *TT format*, for it uses the tree structure with the Tucker decompositions.

We can also estimate the number of parameters for the Tree-Tucker decomposition (TT decomposition) in the general d-dimensional case. The Tucker factors require nrd memory cells. And how many memory cells are needed for the whole tree structure?

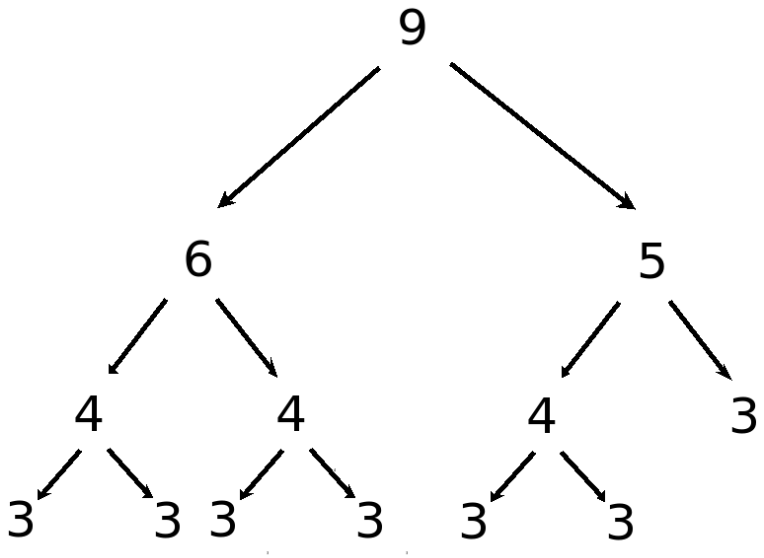

FIG. 4.1. *The splitting of the dimensions.*

These correspond only to the leafs, where the storage is needed for the final three-dimensional arrays. So we have to estimate the number of leafs and denote it by $h(d)$. We already know that $h(3) = 1$ and $h(4) = 2$. For the even argument

$$h(2d) = 2h(d+1),$$

and for the odd one

$$h(2d+1) = h(d+1) + h(d+2).$$

The solution is obviously unique. Our guess (provided by simple calculations) is $h(d) = d - 2$. It suffices just to check that the function of this form satisfies both of the equations and the initial conditions as well.

So the following theorem is proved.

THEOREM 4.1. *Suppose $\mathcal{A}$ is a d-dimensional tensor with a canonical decomposition of rank r. Then there exists a TT decomposition with*

$$nrd + (d-2)r^3 \tag{4.1}$$

parameters.

It is worthy to note that this expression for the number of parameters is valid even for $d = 2$ since in that case we have the skeleton (dyadic) decomposition.

Moreover, what is especially important for us is that the TT decomposition for a given canonical decomposition of rank R can be constructed so that the estimate (4.1) contains *the lowest possible* Tucker rank r! So we seem to have found what we were looking for, that is a computable d-dimensional alternative to the canonical decomposition.

Remark 1. The dimensions in the tensors $\mathcal{U}$ and $\mathcal{V}$ are of the two kinds: the initial dimensions, which we will call *spatial dimensions*, and *auxiliary dimensions* added at each step of recursion. Although we may not distinguish between them

during the compression step (in subsequent nodes they can be split, in principle, arbitrarily), the way of their distribution in the children nodes is important when performing operations in the TT format. We give more details when we consider a simple operation with an array in the TT format.

Remark 2. Theorem 4.1 and Lemmas 3.1 and 3.2 describe the case when the decomposition is *exact* and the TT decomposition is also exact. However, the *approximate TT decomposition* is also possible, now the ranks involved are not the ranks of the matrix but their δ-ranks,[3] where δ is an accuracy parameter. A rigorous analysis of this case is not performed in this paper.

5. Canonical-to-Tree-Tucker compression. Now we are going to propose a numerical procedure for converting an array from the canonical decomposition to the TT format. As a starting point we have a canonical rank-R representation for the tensor $\mathcal{A}$.

We begin with the computation of the Tucker factors of $\mathcal{A}$. It is known that the Tucker factors for the kth index (mode) are equal to the left singular vectors of the $n \times n^{d-1}$ matrix $B_k = [a_{i_k \mathcal{I}_k}]$, where $\mathcal{I}_k$ includes all indices except i_k. Therefore, these factors are equal to the matrix of eigenvectors of the Gram matrix

$$\Gamma_k = B_k B_k^\top.$$

These Gram matrices are easily computable from the canonical decomposition. If $\mathcal{A} = (U_1, U_2, \ldots, U_d)$, then

$$\Gamma_k = G_1 \circ G_2 \circ \cdots \circ G_{k-1} \circ G_{k+1} \circ \cdots G_d,$$

where $\circ$ is the Hadamard (elementwise) product of matrices and G_k is the Gram matrix for U_k:

$$G_k = U_k^\top U_k.$$

We see that Γ_k is the Hadamard product of all G_s with $s \neq k$. This algorithm for the computation of the Tucker factors from the canonical decomposition based on Gram matrices is known [1].

The total cost is determined by the computation of all Gram matrices G_k in, obviously, $\mathcal{O}(dn^2R)$ operations followed by the acquisition of all Γ_k in $\mathcal{O}(d \cdot dR^2)$ operations and by the truncated eigenvalue decompositions for all Γ_k in $\mathcal{O}(dR^3)$ operations.

It has to be taken into account that the eigenvalues of Γ_k are the squares of the singular values of B_k. So, if we want to achieve the approximation accuracy of order of δ, the eigenvalues should be truncated at $\varepsilon = \delta^2$. This also leads to a loss of numerical precision, because the highest accuracy obtained is the square root of the machine accuracy. However, this is not very frustrating as in many applications the accuracy of order 10^{-5} is more than enough.

If n is large indeed, the n^2 complexity may be prohibitive. But, it can be easily avoided by the preliminary QR-decomposition for each of the factor matrices U_k. Then n is replaced by R, and the total cost is $\mathcal{O}(dR^3 + d^2R^2)$.

As soon as the Tucker factors $\widehat{U}_k$ are computed, we project each of the canonical factors U_k onto them and obtain matrices $U'_k = \widehat{U}_k^\top U_k$ which are the canonical factors for the Tucker core of the tensor $\mathcal{A}$.

[3] A δ-rank of a matrix A is a minimal number k such that there exists a rank-k matrix B such that $||A - B|| \leq \delta$.

A brief formal summary is encapsulated in Algorithm 1.

ALGORITHM 1: COMPUTATION OF THE TUCKER FACTORS. *Given a tensor $\mathcal{A}$ of dimension d in a rank-R canonical format $\mathcal{A} = (U_1, U_2, \ldots, U_d)$, proceed with the following steps:*

For $k = 1$ to d.

(1) *Compute the Gram matrices $G_k = U_k^\top U_k$.*
(2) *Compute the matrices $\Gamma_k = \circ_{i \neq k} G_i$.*
(3) *Truncate the eigenvalues of G_k with a prescribed accuracy ε and compute the truncated eigendecomposition $G_k = \widetilde{U}_k \Lambda_k \widetilde{U}_k^\top$, where $\widetilde{U}_k$ is a matrix of size $n \times r$ with orthonormal columns and Λ_k is a diagonal matrix of size $r \times r$. The matrices U_k' are the Tucker factors.*
(4) *Compute the canonical factors of the Tucker core as $\widehat{U}_k = (U_k')^\top U_k$.*

End for

From now on, we will work only with the core tensor for which we leave the same notations: $\mathcal{A}$, n, and R now pertain to the core tensor. In the recursive approximation we split the indices into the two "long indices" and eventually need to find some canonical decompositions for the corresponding "children tensors." They are given by the $UV^\top$ decomposition of the matrix $B = [a_{\mathcal{I}_1 \mathcal{I}_2}]$. A certain decomposition of this form with R columns in U and V easily emanates from the canonical representation of $\mathcal{A}$: the latter separates all initial indices and automatically does the job of separating just two groups of them merged in $\mathcal{I}_1$ and $\mathcal{I}_2$. Hence, we know that

$$B = UV^\top$$

with U and V having R columns. However, the rank of B might be less than R. We can reveal this rank from the singular value decomposition of B. For a singular triplet σ, u, v we have

$$Bv = \sigma u, \quad B^\top u = \sigma v,$$

and since only singular vectors related to nonzero singular values are needed, we look only for those u and v that span the columns of U and V, respectively:

$$u = Ux, \quad V = Vy.$$

By substituting these into the above equations we obtain

$$(UV^\top)Vy = \sigma Ux, \quad (VU^\top)Ux = \sigma Vy.$$

Consequently,

$$G_v y = \sigma x, \quad G_u x = \sigma y, \tag{5.1}$$

where

$$G_u = U^\top U, \quad G_v = V^\top V$$

are the Gram matrices for U and V. The Gram matrices for U and V are computed similarly to those in the case of Tucker factors. Namely, G_u and G_v are equal to the Hadamard products of all Gram matrices for the factors with indices in $\mathcal{I}_1$ and in $\mathcal{I}_2$, respectively:

$$G_u = \circ_{k : i_k \in \mathcal{I}_1} G_k, \tag{5.2}$$

$$G_v = \circ_{k:i_k \in \mathcal{I}_2} G_k. \tag{5.3}$$

Now we can find σ, x, y from the eigendecomposition of $G_v G_u$, since

$$G_v G_u x = \sigma^2 x. \tag{5.4}$$

Also we ought to impose the normalization conditions: from $||u|| = ||v|| = 1$ it follows that

$$(G_u x, x) = 1, \quad (G_v y, y) = 1.$$

Once the Gram matrices are known, the computation of the SVD of B reduces to the eigendecomposition of a matrix of size $R \times R$.

Having computed all the vectors x, y related to the left and right singular vectors of B, we put them together as columns of matrices X and Y. Then, consider matrices $U' = UX$, $V' = VY$, and the diagonal matrix Λ composed of the singular values. The matrices U' and V' can be treated as tensors of order $d_1 + 1$ and $d_2 + 1$ as explained in the previous section. By Lemma 3.2, these tensors possess canonical rank-R representations. How can we compute them? Consider, for example, U'. The $d_1 + 1$ indices comprise d_1 of them accounting for the "old" dimensions and one for the auxiliary dimension. The canonical factors for the "old" dimensions are exactly the same as they were. The auxiliary dimension arises by the SVD, has the size equal to the splitting rank $r = \operatorname{rank} B$, and its canonical factor is simply equal to X.

The recursive procedure is summarized in Algorithm 2.

ALGORITHM 2: CANONICAL-TO-TREE-TUCKER COMPRESSION. *Given an array $\mathcal{A}$ of dimension d in the canonical rank-R format $(U_1, U_2, \ldots, U_d)$, compute its Tree-Tucker representation (TT representation).*

1. *If $d > 3$, then split all indices in the two "long indices" $\mathcal{I}_1$ and $\mathcal{I}_2$ so that the numbers of all possible values for each are as close as possible (if sizes for all axes are equal, then it is sufficient to take $d_1 = [d/2], d_2 = d - d_1$).*
2. *Compute the Gram matrices G_u and G_v by formulas* (5.2), (5.3) *and the matrices X, Y, and Λ according to* (5.4).
3. *Find canonical representations of the children tensors $\mathcal{U}'$ and $\mathcal{V}'$: take all previous factors corresponding to $\mathcal{I}_1$ and the extra one as X for $\mathcal{U}'$, and all factors corresponding to $\mathcal{I}_2$ and the extra one as Y for $\mathcal{V}'$, i.e., X, Y are factor matrices for the auxiliary dimensions for $\mathcal{U}'$ and $\mathcal{V}'$, respectively.*
4. *Repeat the above recursively for the children tensors $\mathcal{U}'$ and $\mathcal{V}'$ of smaller dimension.*

6. A simple operation in the TT format. What can we learn from the TT representation in the work with tensors? Full linear algebra in that format is the subject of a separate study and will be reported in the forthcoming paper. Here we expose some spectacular advantages on an example of simple but widespread operation.

Given a tensor $\mathcal{A}$ of dimension d and size-n vectors $w_1, w_2, \ldots, w_d$, we are interested in computing a d-dimensional convolution defined as follows:

$$I = \mathcal{A} \times_1 w_1^\top \times_2 w_2^\top \ldots \times_d w_d^\top.$$

This is just a number that may be interpreted as the approximation to a multidimensional integral of a discretized function over the d-dimensional cube.

Using the TT format, we consider first the root node splitting

$$a_{\mathcal{I}_1\mathcal{I}_2} = \sum_{\alpha=1}^{r} \mathcal{U}_{\mathcal{I}_1\alpha}\mathcal{V}_{\mathcal{I}_2\alpha},$$

from which it stems that

$$I = \sum_{\alpha=1}^{r} I_\alpha^{(u)} I_\alpha^{(v)}.$$

Consequently, for each value of α we have to compute the convolutions with $\mathcal{U}$ along the indices in $\mathcal{I}_1$ and with $\mathcal{V}$ along the indices in $\mathcal{I}_2$. Despite that only one number is needed, from the leafs of the root node we ask for $\widehat{r}$ numbers (as the convolution is performed in all "spatial" dimensions but no longer in the auxiliary ones). Since the procedure is going to be recursive, the auxiliary numbers could be too many.

We should take some measures against the blow-up of these numbers during recursion. In order to do that, we have to watch carefully how the auxiliary dimensions are distributed between the children nodes. In the root, there are no auxiliary dimensions. In the children nodes we have one. Take one of the children, after the splitting we have two nodes, one with one auxiliary dimension and the other with two auxiliary dimensions. And here is the point which is most important. When we split a node with two auxiliary dimensions, we should put one of them to the left child and another to the right child, so both of them would acquire two auxiliary dimensions. Thus, by using this constraint on the splitting procedure we have at maximum two auxiliary dimensions in each node. In this case we will say that auxiliary dimensions are *properly distributed*. The splitting scheme is illustrated in Figure 6.1, where the first number in brackets denotes the number of spatial dimensions, and the second does the number of auxiliary dimensions. If the splitting obeys that rule, then the convolution is evaluated efficiently. In the recursive procedure, for each node we are to convolve over all its "spatial" dimensions. If there is only one auxiliary dimension, the procedure returns a vector, if there are two, then it returns a matrix.

Finally we arrive at a recursive algorithm that computes the convolution over all spatial dimensions.

ALGORITHM 3: TREE-TUCKER CONVOLUTION. *Given a tensor of dimension d in the TT format with properly distributed auxiliary dimensions and vectors $w_1, \ldots, w_d$ of appropriate sizes, compute the convolution.*

(1) RECURSION STAGE.
In every node of the tree, if it is not a leaf, then recursively compute the convolution in spatial dimensions in the left and the right children, otherwise compute it directly in 2 *(the number of auxiliary dimensions is* 1*) or* 1 *(the number of auxiliary dimensions is* 2*) mode multiplications of a three-dimensional tensor.*

(2) COMBINATION STAGE.
If the number of auxiliary dimensions in the node is 0 *and the children nodes keep vectors $I_u = [I_\alpha^{(u)}]$ and $I_v = [I_\alpha^{(v)}]$, then return the scalar product $I_u^\top I_v$.*
If the number of auxiliary dimensions in the node is 1 *and the left node keeps a matrix $I_u = [I_{\alpha\gamma}^{(u)}]$ of size $r_1 \times r_2$ and the right node keeps a vector $I_v = [I_\gamma^{(v)}]$ of size r_2, then return the vector $I_u I_v$ of size r_1.*
If the number of auxiliary dimensions is 2 *and both children nodes keep matrices I_u and I_v of sizes $r_1 r_3$ and $r_2 r_3$, respectively, then return the matrix*

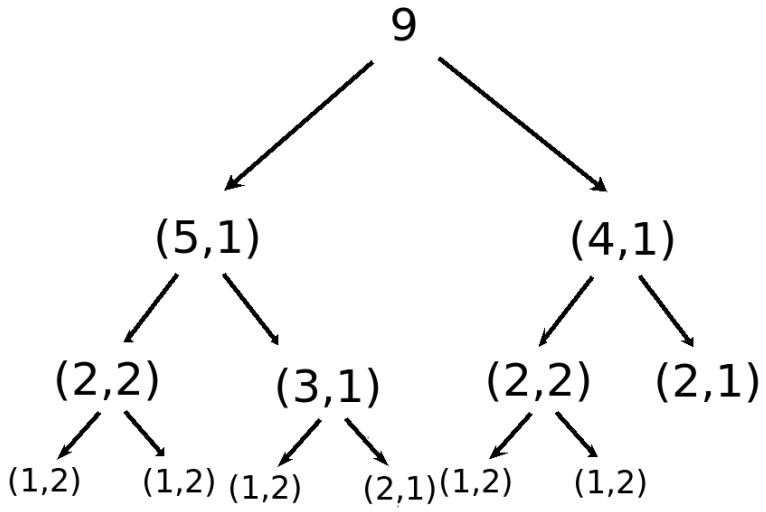

FIG. 6.1. *Distribution of the auxiliary dimensions, the first number in brackets is the number of spatial dimensions in a node, and the second is the number of auxiliary dimensions in a node.*

$I_u I_v^\top$ *of size* $r_1 r_2$.

Prior to the call to this algorithm, we should compute the convolution with the Tucker factors, and only after that with the core tensor. In the preliminary step all convolution vectors are projected into the Tucker basis by the cost of dnr operations, where r is the Tucker rank. It is remarkable that the complexity of the recursive computation is linear in d. It is readily seen that the complexity estimate for the convolution with the core is $\mathcal{O}(dr^4)$.

7. Examples. We have implementations of the previously described algorithms and are ready to apply them to some tensors. The algorithms were implemented in the mixed Fortran and C code, with C language for the hierarchical tree structure and with Fortran for "low level" array operations, and all the timings were performed on an Intel Core2 notebook with 2 GB of RAM, OpenSolaris operating system, and C and Fortran compilers from SunStudio Express 11. All timings that will be presented are averaged over a series of runs.

Consider first some examples for the canonical-to-Tree-Tucker (canonical-to-TT) compression. The SVD is stable and thence the optimal low-rank approximation always exists. In contrast, the canonical decomposition is unstable, i.e., the distance from a tensor to the manifold of rank-r tensors may be equal to zero while the optimal rank-r tensor does not exist. In that case, all ranks in the TT decomposition will still be equal to r, which makes TT much more effective in terms of compression.

7.1. Laplace operator. It is instructive to consider the d-dimensional Laplacian (cf. [3]) of the form

$$\triangle_d = (\triangle \times I \times \cdots \times I) + \cdots + (I \times \cdots \times I \times \triangle), \tag{7.1}$$

where $\triangle$ is a discretization of the one-dimensional Laplacian. The canonical rank of the given representation (7.1) is obviously d. However, as was shown in [3], this tensor can be approximated by a rank-2 tensor with any prescribed accuracy. The distance

TABLE 7.1
Timings and splitting ranks for the d-dimensional Laplacian, $n = 1024$; decompositions are exact.

d	Time for TT construction	Canonical rank	Splitting rank
10	0.01 sec	10	2
20	0.09 sec	20	2
40	0.78 sec	40	2
80	13 sec	80	2
160	152 sec	160	2
200	248 sec	200	2

between $\triangle_d$ and matrices of a rank-2 tensor is equal to zero by the following reason. Bring in a rank-1 parametric tensor function

$$C(t) = \otimes_{i=1}^{d}(I + t\triangle)$$

and take its derivative at $t = 0$, the result is

$$C'(0) = \triangle_d.$$

Since $C(t)$ is a polynomial in t, it can be approximated by a finite difference as follows:

$$C'(0) = (C(h) - C(0))/h + \mathcal{O}(h).$$

Here $C(h)$ and $C(0)$ are both of tensor rank 1, so rank-2 tensors are located in whatever small vicinity of $\triangle_d$. In machine arithmetic, the numerical precision comes into play and enters the estimate for the approximate canonical rank as $\log \varepsilon$. However, the TT format is stable and all the splitting ranks are *exactly* equal to 2. Therefore, the storage for TT is simply $2nd+8(d-2)$ floats, which is smaller than for the canonical rank-d format for an unstructured tensor. This is confirmed by the numerical results presented in Table 7.1 along with timings for the canonical-to-TT compression by using Algorithm 1. The one-dimensional grid size is 32, so the tensor size in each dimension is $n = 1024$.

7.2. Sine function. Another example borrowed from [3] is the following function-related tensor:

$$f(x_1, x_2, \ldots, x_d) = \sin(x_1 + x_2 + \cdots + x_d). \tag{7.2}$$

In [3] it was shown that there exists a tensor rank-d representation by using trigonometrical identities. However, if we consider this tensor in the complex field, then its tensor rank is exactly 2 due to the identity

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

With $x = x_1 + x_2 + \cdots + x_d$ we see that each exponent is a separable function. One of the interesting properties of the canonical rank is that it may be different over the complex and the real fields. The rank of a matrix, however, does not change if we extend the field. Therefore, for the TT format we have all splitting ranks equal to 2, so the memory in the TT format for this tensor is $2nd + 8(d-2)$. The timings for the canonical-to-TT compression in this case are the same as those for the Laplacian.

TABLE 7.2
Timings and splitting ranks for the second-order operator, $n = 1024$; decompositions are exact.

d	Time for TT construction	Canonical rank	Splitting rank
10	$2 \cdot 10^{-4}$ sec	55	7
20	18 sec	210	12
40	1286 sec	820	22

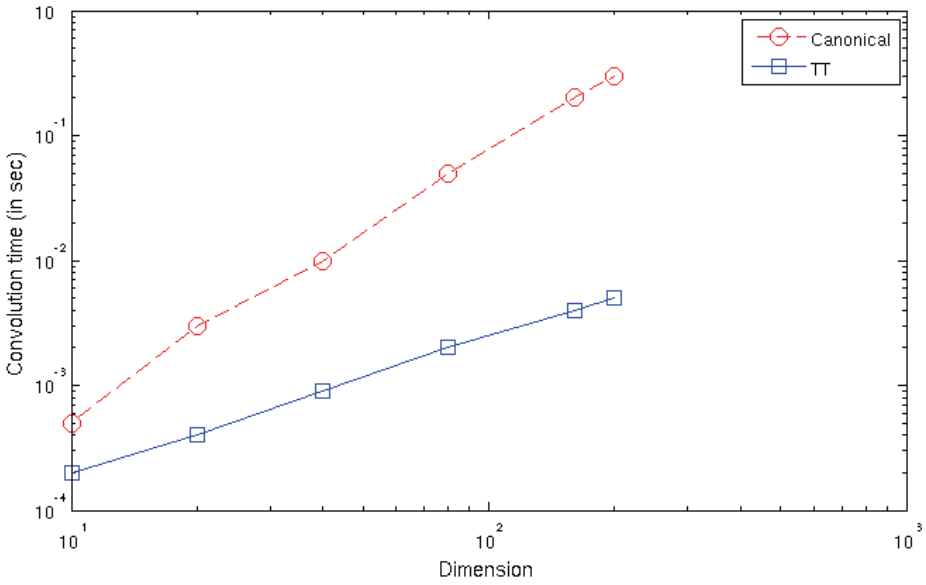

FIG. 7.1. *TT versus canonical convolution timings, sine example, $n = 1024$, decompositions are exact.*

7.3. General second-order operator. Our third example is where we do not know the true value of rank. It is a discretization of the general second-order operator

$$\mathcal{L}P = \sum_{i \le j} \sigma_{ij} \frac{\partial^2 P}{\partial x_i \partial x_j} \tag{7.3}$$

with Dirichlet boundary conditions. A similar operator appears in the Black-Scholes equation for multiasset modelling (cf. [13]). In our examples we take σ_{ij} at random and observe that it does not affect the rank values.

Each summand has the form

$$\frac{\partial^2 P}{\partial x_i \partial x_j},$$

and after a natural discretization with finite differences it becomes a rank-1 tensor, so obvious rank estimate is $d(d+1)/2$. However, our numerical experiments show that it rather behaves like $d/2$. The results are given in Table 7.2. Although we obtain different splitting ranks, the Tucker ranks themselves are equal to 3. The canonical-to-TT compression rates in this example look very convincing.

7.4. Convolution timings. Consider the above sine example and compute the convolution by using the TT format and the original canonical format. As we already

remarked, this is an approximate way to compute the multidimensional integral

$$\int \sin(x_1 + x_2 + \cdots + x_d) u(x_1) u(x_2) \dots u(x_d) dx_1 dx_2 \dots dx_d$$

over the d-dimensional cube. In Figure 7.1 we plot (in the log-log scale) the results with 1024 points in each direction.

The TT convolution is clearly faster and has better asymptotics with respect to the dimension d.

7.5. Discussion. All examples considered in this paper contain decompositions that are exact, i.e., the ranks are exact ranks. They illustrate a different nature of the canonical decomposition and the TT decomposition. It is interesting to consider the cases when the decomposition is not exact but *approximate* with some accuracy parameter δ, which often is true in practice. These numerical experiments are out of the scope of the current paper and are the subject of ongoing research.

8. Conclusions and future work. We have presented a new way to represent multidimensional arrays of possibly large dimension in a hierarchical structure called the Tree-Tucker format. We have shown that the number of parameters in the new structure is never significantly higher than in the canonical format, but, as seen from our examples, it can be pronouncedly smaller. We also proposed a Tree-Tucker convolution algorithm (a discrete analogue to the multidimensional integration) and showed the benefits of the new format in a few examples.

The new decomposition is based only on low-rank matrix approximations, so it can be computed by using standard tools (SVD) and looks very promising. Our next goal is to develop efficient arithmetic in this format: approximation (recompression) of a given TT format tensor by another TT format tensor with a smaller number of parameters, addition of tensors, and, which is most important, matrix-by-vector and matrix-by-matrix multiplications. This will be reported in a forthcoming paper.

We are looking forward to applying the new format to multidimensional differential equations. We already tried our method on the discretized Black–Scholes operator in the multiasset option pricing, where preliminary experiments show that everything goes smoothly and allows one to tackle the dimensions up to 100 (which is impossible for all modern methods up to now).

A very tempting application could be the multidimensional Schrödinger equation. Although everything looks rather straightforward and it was even claimed in [3] that the multiparticle Schrödinger equation admits a nice low-canonical-rank approximation; it might not be the exact case. Even the wavefunction cannot be sufficiently well approximated in the canonical format. It can be shown that even for the helium atom, where $d = 6$ and there are two electrons, the ground state wavefunction depends on the distance between these electrons like $\frac{1}{r_{12}}$, and it is well known that this function is not globally well separable. The results of [3] are, of course, valid in the part of dependence on d, but the practical estimates are not very useful. For the Schroedinger operator we have to look for another format (neither canonical, nor TT), maybe using some ideas of multiresolution. This work is ongoing and the results will be reported elsewhere.

Acknowledgments. We are thankful to the anonymous referees who helped to improve the paper with their comments and suggestions.

REFERENCES

[1] B. W. Bader and T. G. Kolda, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30 (2007), pp. 205–231.
[2] B. Bader and T. Kolda, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[3] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 10246–10251.
[4] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.
[5] J. Carroll and J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition*, Psychometrica, 35 (1970), pp. 283–319.
[6] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.
[7] F. Gantmacher, *Theory of Matrices*, Vols. 1 and 2, Chelsea, New York, 1959.
[8] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., John Hopkins University Press, Baltimore, 1996.
[9] R. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.
[10] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multlinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[11] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of high-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.
[12] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 939–956.
[13] J. Persson and L. von Sydow, *Pricing European multi-asset options using a space-time adaptive FD-method*, Comput. Vis. Sci., 10 (2007), pp. 173–183.
[14] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[15] E. Tyrtyshnikov, *Mosaic-skeleton approximations*, Calcolo, 33 (1997), pp. 47–58.
[16] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic–skeleton method*, Computing, 64 (2000), pp. 367–380.
[17] E. Tyrtyshnikov, *Tensor approximations of matrices generated by asymptotically smooth functions*, Sb. Math., 194 (2003), pp. 941–954.
[18] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Linear Algebra for Large Scale and Real-Time Applications (Leuven, 1992), NATO Adv. Sci. Inst. Ser. E Appl. Sci. 232, Kluwer Acad. Publ., Dordrecht, 1993, pp. 293–314.