TUCKER DIMENSIONALITY REDUCTION OF THREE-DIMENSIONAL ARRAYS IN LINEAR TIME *

I. V. OSELEDETS †, D. V. SAVOSTIANOV ‡, AND E. E. TYRTYSHNIKOV §

Abstract. We consider Tucker-like approximations with an $r \times r \times r$ core tensor for three-dimensional $n \times n \times n$ arrays in the case of $r \ll n$ and possibly very large n (up to $10^4 - 10^6$). As the approximation contains only $\mathcal{O}(rn + r^3)$ parameters, it is natural to ask if it can be computed using only a small amount of entries of the given array. A similar question for matrices (two-dimensional tensors) was asked and positively answered in [14]. In the present paper we extend the positive answer to the case of three-dimensional tensors. More specifically, it is shown that if the tensor admits a good Tucker approximation for some (small) rank r, then this approximation can be computed using only $\mathcal{O}(nr)$ entries. Moreover, in many cases it can be computed with $\mathcal{O}(nr^3)$ complexity.

Key words. Multidimensional arrays, Tucker decomposition, tensor approximations, low rank approximations, skeleton decompositions, dimensionality reduction, data compression, large-scale matrices, data-sparse methods.

1. Introduction. Multidimensional arrays of data appear in many different applications. One can mention signal processing, statistics [3, 1, 4], chemometrics [5], face recognition [7], solving multidimensional integral and differential equations [6](a very comprehensive list of references on the subject can be found on a Three-mode company web site, [2]). These arrays often can not be handled by standard methods because of their huge sizes: we cannot solve linear systems or calculate required decompositions due to speed or memory restrictions. The obvious solution is to perform a sort of *dimensionality reduction*: an initial "large" array is transformed to a smaller array for which we can use standard methods. However, such a reduction by conventional approaches may be computationally still too expensive. In this paper we suggest a way to make it not only feasible but even quite fast. We will focus only on three-dimensional arrays mostly to simplify the presentation and note that our results can be generalized to more dimensions.

The most useful method to reduce dimension is based on the celebrated *Tucker decomposition* [23] and solves the following problem: *given a three-dimensional array (tensor)* $\mathcal{A} = [a_{ijk}]$, $i = 1, ..., n_1$, $j = 1, ..., n_2$, $k = 1, ..., n_3$, *compute its approximation*

$$a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'} + e_{ijk} \tag{1.1}$$

with the e_{ijk} *to be sufficiently small for prescribed* r_1, r_2, r_3. Here we ignore the orthogonality requirements in the original Tucker decomposition. Despite that, the matrices $U = [u_{ii'}], V = [v_{jj'}], W = [w_{kk'}]$ will be referred to as *Tucker factors*, and the $r_1 \times r_2 \times r_3$ tensor $\mathcal{G} = [g_{i'j'k'}]$ as the *core tensor*.

*Supported by the Russian Fund of Basic Research (grants 05-01-00721, 04-07-90336 and 06-01-08052) and a Priority Research Grant ONM-3 of the Department of Mathematical Sciences of Russian Academy of Sciences.

†Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(ivan@bach.inm.ras.ru).

‡Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(draug@bach.inm.ras.ru).

§Institute of Numerical Mathematics, Russian Academy of Sciences, Gubkina Street, 8, Moscow(tee@bach.inm.ras.ru).

A well-known method for the computation of the Tucker decomposition is based on the SVD algorithm. Consider three rectangular matrices ("unfoldings") of the tensor $\mathcal{A}$:

$$\begin{aligned} A^{(1)} &= [a^1_{i(jk)}] = [a_{ijk}], \\ A^{(2)} &= [a^2_{j(ki)}] = [a_{ijk}], \\ A^{(3)} &= [a^3_{k(ij)}] = [a_{ijk}]. \end{aligned} \tag{1.2}$$

The superscripts 1,2,3 in the definitions of unfoldings define which index (first, second or third) is used; two other indices are merged into one "long" index.

The left ("short") singular vectors of the SVD-s of these matrices

$$A^{(1)} = U\Sigma_1\Phi_1^\top, \qquad A^{(2)} = V\Sigma_2\Phi_2^\top, \qquad A^{(3)} = W\Sigma_3\Phi_3^\top \tag{1.3}$$

give the factors U, V, W of the Tucker decomposition, possibly after an appropriate truncation, and the core is computed as a convolution of the tensor with these matrices:

$$g_{i'j'k'} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk}u_{ii'}v_{jj'}w_{kk'}. \tag{1.4}$$

The tensor dimension can be large (for example, $n = 10^4 - 10^6$ for some tensors coming from three-dimensional integral equations). The array itself can not be even stored in the operative memory as $\mathcal{O}(n^3)$ memory cells are needed. The computation of the SVDs in (1.3) by standard methods costs $\mathcal{O}(n^4)$ operations and is anyway prohibitive for $n \geq 1000$.

However, we are chiefly interested in the case $r \ll n$ and the Tucker decomposition contains only $\mathcal{O}(rn + r^3)$ parameters. If a good approximation exists, we can ask if it can be computed using only a small amount of entries of the tensor $\mathcal{A}$. A similar question for matrices (two-dimensional tensors) was asked and positively answered in [14]. In the present paper we extend the positive answer to the case of three-dimensional tensors. More specifically, it will be shown that if the tensor admits a good Tucker approximation for some (small) rank r, then this approximation can be computed using only $\mathcal{O}(nr)$ entries. Moreover, in many cases it can be computed with $\mathcal{O}(nr^3)$ complexity.

Prior to investigation of special low-parametric (data-sparse) representations obtained only from the knowledge of a small portion of the data entries, we use a general assumption that *some* low-parametric approximations exist. In other words, we consider the cases with sufficiently small approximate tensor rank estimates. Several estimates for many practically interesting cases are developed in [15, 19, 20]. We can mention also some practical algorithms using interpolation and other function approximation techniques or additional structural properties rather than the given arrays of data [15, 22]. The reference [21] is most close to the paradigm of a completely data-based method (using no knowledge beyond the data themselves); however, [21] contains no proof for the existence of a sufficiently good low-data representation and does not suggest a general adaptive procedure for selecting "most meaningful" entries. Recently, much attention has been paid to the approximation of a given matrix by a low rank matrix using randomized algorithms, for example see [24]. To our best knowledge, these algorithms are fast only asymptotically with very large constants in the estimates and can not be applied in practice. Moreover, the authors do not

report any numerical results in their articles so we can not compare their methods with our method. In this paper we present the existence results and the adaptive 3D cross algorithms.

2. Notations and definitions. Let us recall some basic facts about tensors [10, 11].

DEFINITION 2.1. *The norm of the $n_1 \times n_2 \times n_3$ tensor $\mathcal{A} = [a_{ijk}]$ is defined similarly to the Frobenius norm for matrices as*

$$||\mathcal{A}|| = ||\mathcal{A}||_F = (\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} a_{ijk}^2)^{1/2}.$$

Also, let

$$||\mathcal{A}||_C = \max_{i,j,k} |a_{ijk}|$$

be a Chebyshev norm *of a tensor $\mathcal{A}$.*

DEFINITION 2.2. *(Outer product.) If $\mathcal{A}$ is a p-index array $a_{i_1,i_2,\ldots,i_p}$ and $\mathcal{B}$ is a q-index array $b_{j_1,j_2,\ldots,j_q}$, then $\mathcal{C} = \mathcal{A} \otimes B$ is defined as a (p+q)-index array with elements*

$$c_{i_1,i_2,\ldots,i_p,j_1,j_2,\ldots,j_q} = a_{i_1,i_2,\ldots,i_p} b_{j_1,j_2,\ldots,j_q}.$$

Tensors can be multiplied by matrices along a specified index (mode) direction.

DEFINITION 2.3. *(Mode convolution or n-mode product) If $\mathcal{A} = [a_{ijk}]$ is a $n_1 \times n_2 \times n_3$ array, U is a $n_1 \times q$ then their product $\mathcal{B}$ is $q \times n_2 \times n_3$ is defined as*

$$\mathcal{B} = \mathcal{A} \times_i U, \qquad \mathcal{B}_{ijk} = \sum_{i'=1}^{n_1} u_{ii'} a_{i'jk}.$$

The operations $\mathcal{A} \times_j U$, $\mathcal{A} \times_k U$ are defined analogously, provided that U and $\mathcal{A}$ have appropriate sizes. In this notation, the Tucker decomposition (1.1) can be written as

$$\mathcal{A} = \mathcal{G} \times_i U \times_j V \times_k W.$$

We will say that a tensor has a *rank-(r_1, r_2, r_3) (Tucker) decomposition*, if (1.1) holds [10, 11].

The important objects are *slices* of the three-dimensional arrays.

DEFINITION 2.4. *If $\mathcal{A} = [a_{ijk}]$ is a $n_1 \times n_2 \times n_3$ array, then its k-th slice by the third index is a $n_1 \times n_2$ matrix A_k with elements*

$$(A_k)_{ij} = a_{ijk}.$$

The slices by two other indices (i and j) are defined in the same way.

The "short" vectors along the modes i, j and k will be referred to as *columns, rows* and *fibers*.

3. Existence theory. Suppose an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A} = [a_{ijk}]$ is given.

Assume also that there exists a rank-(r_1, r_2, r_3) Tucker approximation to $\mathcal{A}$ with the accuracy ε:

$$\mathcal{A} = \mathcal{G} \times_i U \times_j V \times_k W + \mathcal{E}, \quad ||\mathcal{E}|| = \varepsilon. \tag{3.1}$$

If we are aware that such an approximation exists, then a generally different approximation of the same type with the accuracy bound $c\varepsilon$ (where $c > 1$ is a *deterioration coefficient*) can be constructed from the knowledge of roughly the same amount of entries as those explicitly involved in (3.1). We want to prove this together with a bound on the deterioration coefficient c depending only upon dimensions but not on the entries of the array.

THEOREM 3.1. *Suppose* (3.1) *holds for some* U, V, W *and* $\mathcal{G}$. *Then there exist matrices* U', V' *and* W' *of sizes* $n_1 \times r_1$, $n_2 \times r_2$ *and* $n_3 \times r_3$ *and consisting of some* r_1 *columns,* r_2 *rows and* r_3 *fibers, of* $\mathcal{A}$, *respectively, and a tensor* $\mathcal{G}'$ *such that*

$$\mathcal{A} = \mathcal{G}' \times_i U' \times_j V' \times_k W' + \mathcal{E}',$$

where

$$||\mathcal{E}'||_C \leqslant (r_1 r_2 r_3 + 2r_1 r_2 + 2r_1 + 1)\varepsilon.$$

Proof. Consider an unfolding matrix $A^{(1)}$ of the array $\mathcal{A}$. Since $\mathcal{A}$ has a rank-(r_1, r_2, r_3) approximation with accuracy ε, it is easy to see from (3.1) that $A^{(1)}$ has rank-r_1 approximation with the same accuracy. A low-rank matrix can be approximated by its *skeleton decomposition*

$$A^{(1)} = C\widehat{C}^{-1}B^\top + \mathcal{E}_1$$

where C is a $n_1 \times r_1$ matrix containing some r_1 columns of $A^{(1)}$ and B is a $n_2 n_3 \times r_1$ matrix containing some r_1 rows of $A^{(1)}$, and $\widehat{C}$ is a submatrix on the intersection of these rows and columns. In [13] it was proved that if $\widehat{C}$ is a *submatrix of maximal volume* (i.e. the $r_1 \times r_1$ submatrix which has the largest absolute value of the determinant) in $\mathcal{A}^{(1)}$ then ε_1 is bounded as follows:

$$||\mathcal{E}_1||_C \leqslant (r_1 + 1)\varepsilon,$$

where $||\cdot||_C$ denotes the largest magnitude element of a matrix (array). Also, if $\widehat{C}$ is a maximal volume submatrix, it is easy to prove (cf. [13]) that the elements of $C\widehat{C}^{-1}$ are not greater than 1 in modulus. Consequently,

$$|a_{ijk} - \sum_{s=1}^{r_1} \gamma_{is} z_{jks}| \leq (r_1 + 1)\varepsilon,$$

where

$$|\gamma_{is}| \leq 1, \qquad 1 \leq i \leq n_1,$$

and z_{jks} are in a one-to-one correspondence with the entries of $B^\top$ (for a fixed s these elements present a slice by the index i of the array $\mathcal{A}$). Also note that

$$\gamma_{is} = \sum_{l=1}^{r_1} u_{il}\phi_{ls},$$

where the matrix $[u_{il}]$ consist of some columns of $\mathcal{A}$.

Now let us look more closely at the matrix $B^\top$. In a reshaped form, it becomes the tensor with the elements z_{jks}. As previously, unfold this tensor along the index j. The ε-rank of the unfolding matrix does not exceed r_2. Using again the result of [13], we obtain the following inequalities:

$$|z_{jks} - \sum_{t=1}^{r_2}\sum_{\tau=1}^{r_2}\psi_{t\tau}v_{jt}w_{ks\tau}| \le (r_2+1)\varepsilon,$$

where the arrays $[v_{jt}]$ and $[w_{ks\tau}]$ consist of some rows and fibers of $\mathcal{A}$.

Unfolding the array $[w_{kt\tau}]$ by the index k, we observe that the ε-rank of the unfolding matrix cannot be larger than k_3. Hence, this matrix admits the skeleton approximation with the error bound

$$|w_{ks\tau} - \sum_{\alpha=1}^{k_3}\sum_{\beta=1}^{k_3} x_{k\alpha}\zeta_{\alpha\beta}y_{\alpha s\tau}| \le (r_3+1)\varepsilon.$$

Finally,

$$|a_{ijk} - \sum_{l=1}^{r_1}\sum_{s=1}^{r_1}\sum_{t=1}^{r_2}\sum_{\tau=1}^{r_2}\sum_{\alpha=1}^{r_3}\sum_{\beta=1}^{r_3}(\phi_{ls}\psi_{t\tau}\zeta_{\alpha\beta}y_{\alpha t\tau})u_{il}v_{jt}x_{k\alpha}| \le |a_{ijk} - \sum_{s=1}^{r_1}\gamma_{is}z_{jks}| +$$

$$\sum_{s=1}^{r_1}|z_{jks} - \sum_{t=1}^{r_2}\sum_{\tau=1}^{r_2}\psi_{t\tau}v_{jt}w_{ks\tau}| + \sum_{s=1}^{r_1}\sum_{\tau=1}^{r_2}|w_{ks\tau} - \sum_{\alpha=1}^{k_3}\sum_{\beta=1}^{k_3}x_{k\alpha}\zeta_{\alpha\beta}y_{\alpha s\tau}| \le$$

$$(r_1+1)\varepsilon + r_1(r_2+1)\varepsilon + r_1r_2(r_3+1)\varepsilon,$$

which completes the proof.

If $r_1 = r_2 = r_3 = r$ then the error bound becomes $(r+1)(r^2+r+1)\varepsilon \le (r+1)^3\varepsilon$. In the general case, we are not completely satisfied with the error bound of this theorem, because it is not a symmetric function of r_1, r_2, r_3. Of course, the answer can be formally symmetrized, using different permutations of modes (e.g $n_3 \times n_2 \times n_1$) and taking minimum of all these error bounds, but the obtained result seems to be rather artificial. So here we note that a "truly symmetric" version of this theorem is likely to need a different technique.

COROLLARY 3.2. *Under the premises of the theorem,*

$$||\mathcal{E}'||_F \leqslant (r_1r_2r_3 + 2r_1r_2 + 2r_1 + 1)\sqrt{n_1n_2n_3}\varepsilon.$$

4. The cross approximation method. For presentation purposes from now on we will assume that $n_1 = n_2 = n_3 = n$ and $r_1 = r_2 = r_3 = r$.

4.1. The 2D-cross method. In the works [13, 14, 18] the problem of finding a rank-r approximation to a given matrix was connected with finding in matrix A a *submatrix of maximal volume* (i.e. determinant in modulus) among all $r \times r$ submatrices. The latter problem is hard to solve. However, we may be satisfied with a

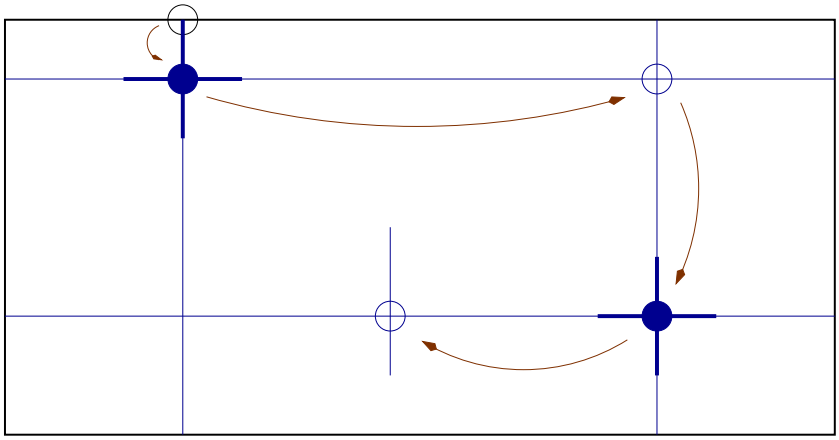


FIG. 4.1. *How a cross method works. Filled dots: elements used for the calculation of cross(i_p, j_p). Empty dots: row-pivot (step 2).*

"sufficiently good" submatrix and some heuristic algorithms. Since these algorithms are to fetch a cross of some columns and rows, we call them *cross algorithms*. Probably the most simple and effective cross algorithm is the Gauss elimination method using some pivoting technique over dynamically selected sets of the entries of the "active matrix" (for general description, see [9]). We will use here the column and row pivoting considered in [8]. This method is simple but may have break-downs (quiting when a good approximation is not obtained) if applied as it is. A cheap practical remedy proposed in [16] is a restarted version of this cross method. For the readers convenience, we give here a brief description of the algorithm.

Algorithm 1 (Cross 2D). Given a matrix A of approximate rank r, approximate it by a matrix $\tilde{A}_r$, which is a sum of r rank-1 matrices $u_p v_p^\top$ (so-called *skeletons*).

(0) To begin with, take some column in A, for example, the first one. Set $j_1 = 1$.
(1) Numbering the steps by p, set $p = 1$. Calculate column j_p of the matrix A and subtract from all elements the corresponding elements of already calculated skeletons. In the resulting vector find the largest magnitude element. Suppose it is located in the row i_p.
(2) Calculate the row i_p of the residue and the next pivot which is its largest magnitude element with a restriction that the element from the j_p-th column can not be chosen again (see Fig. 4.1). Suppose this pivot is located in the j_{p+1}-th column.
(3) Calculate the new cross centered at (i_p, j_p).
(4) If a stopping criterion is not satisfied, set $p := p + 1$ and go to step 1.

The approximation $\tilde{A}_p = \sum_{\alpha=1}^{p} u_\alpha v_\alpha^\top$ is considered good, if

$$\|A - \tilde{A}_p\| \le \varepsilon \|A\|_F \approx \varepsilon \|\tilde{A}_p\|_F.$$

However, the exact computation of the error requires all matrix elements and n^2 operations, which is unacceptable. At the same time, the norm $\|\tilde{A}_p\|_F$ can be computed

via the formula

$$\|\tilde{A}_p\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{\alpha=1}^{r} u_{i\alpha}v_{j\alpha}\right)^2 = \sum_{\alpha=1}^{r}\sum_{\alpha'=1}^{r}(u_\alpha, u_{\alpha'})(v_\alpha, v_{\alpha'}),$$

using $\mathcal{O}(p^2 n)$ operations. And as practical estimator of the error (stopping criterion), we use the norm of a newly computed rank-1 correction. Specifically, we stop if

$$(n-p)\|u_p\|_2\|v_p\|_2 \le \varepsilon\|\tilde{A}_r\|_F.$$

The number $(n-p)$ is a heuristic constant. Note that after p steps of the cross algorithm exactly p rows and columns of the residue are zeroed, so if we assume that the error is "equally distributed" among the remaining $(n-p)$ rows than we immediately arrive to the presented stopping criteria.

Such version of the cross method requires $2rn$ evaluations of matrix elements and $\mathcal{O}(r^2 n)$ additional operations (the reason to count the number of element computations is that the calculation of one element may be a very time-consuming operation). Even if the stopping criteria is satisfied, in some cases the obtained approximation is not good enough (but this happens not very often). To make the method more robust, the *restart* step is performed: we create a sample from the elements of the residue matrix $A - \tilde{A}_r$. If the error, estimated from that sample is large, we proceed with step 3 using the largest magnitude element in the sample as a pivot.

4.2. Towards the 3D-cross method. Consider the unfoldings of the array $\mathcal{A}$ (rectangular matrices of sizes $n \times n^2$ defined by (1.2)) and apply to them the cross approximation algorithm. If the array $\mathcal{A}$ possesses a good Tucker rank-(r,r,r) approximation, then there exist rank-r approximations for the unfoldings $A^{(1)}$, $A^{(2)}$, $A^{(3)}$ which are also good:

$$\tilde{A}_r^{(1)} = U\Psi_1^\top \qquad \tilde{A}_r^{(2)} = V\Psi_2^\top \qquad \tilde{A}_r^{(3)} = W\Psi_3^\top,$$

where U, V, W are $n \times r$ matrices with *orthonormal columns* and matrices Ψ_1, Ψ_2, Ψ_3 are $n^2 \times r$. The Tucker core is calculated by the convolution of the form (1.4) with a_{ijk} being replaced with their approximate values. For example, using the decomposition by the first direction, $\tilde{A}_r^{(1)} = U\Psi_1^\top$, we have

$$a_{ijk} \approx \tilde{a}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha}\psi^1_{jk\alpha}.$$

Substituting this into (1.4), we obtain

$$\begin{aligned} g_{i'j'k'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r} u_{i\alpha}\psi^1_{jk\alpha}\right)\tilde{u}_{ii'}\tilde{v}_{jj'}\tilde{w}_{kk'} = \\ &= \sum_{\alpha=1}^{r}(u_\alpha, \tilde{u}_{i'})\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\tilde{v}_{jj'}\tilde{w}_{kk'}\psi^1_{jk\alpha}\right). \end{aligned} \tag{4.1}$$

This computation needs $\mathcal{O}(n^2 r)$ evaluations of the elements of $\mathcal{A}$ plus $\mathcal{O}(n^2 r^2)$ operations.

Of course, $\mathcal{O}(n^2)$ is much smaller than the total number of elements in the array $\mathcal{A}$, but it is still too large when n is about 10^3.

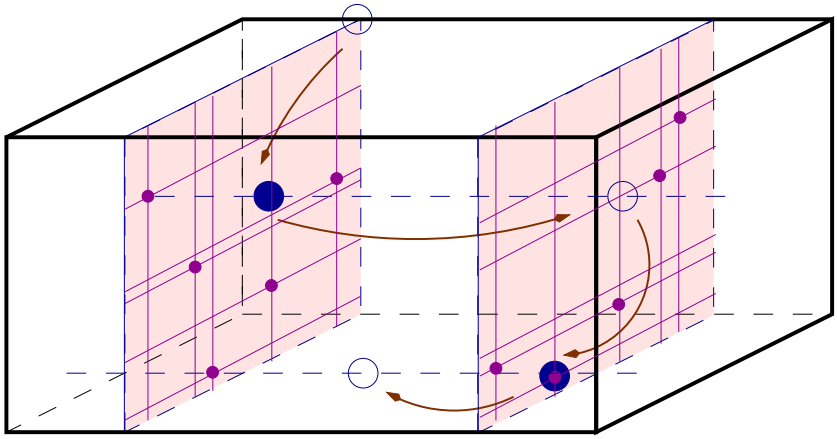


FIG. 4.2. *The work of the 3D-cross method. The big filled and empty dots correspond to elements for the "outer" cross algorithm and small dots show elements used for the "inner" cross algorithm, approximating a particular two-dimensional slice*

4.3. How to achieve linear complexity. We want to achieve linear complexity in n. To this end, we have to get rid of the computation of all elements in the slices of $\mathcal{A}$ used in the unfoldings (1.2) (then we avoid n^2-long vectors). We suggest to approximate the slices by the same cross algorithm developed for matrices. Since $\mathcal{A}$ has a good Tucker approximation with the accuracy ε, each slice $A_k = [(a_k)_{ij}]$ can be accurately approximated by a rank-r matrix. In what follows, we will never store a slice as a full $n \times n$ matrix and never refer to all its elements. Instead, we deal only with some low-rank approximations for the slices.

Algorithm 2. Given an $n \times n \times n$ array $\mathcal{A}$, take one of the indices i, j, k as the "leading index", let it be k. Then consider the corresponding unfolding matrix of size $n \times n^2$ and approximate it applying the cross method. The columns of the unfolding matrix are calculated as usual, but each of the "long" rows is considered as a matrix of size $n \times n$ to be approximated by the same cross method. The expected number of arithmetic operations is *almost linear* in sizes of the array: the complexity is $\mathcal{O}(nr^d)$ operations for some small $d > 0$.

(0) Numbering the steps by p, set p to 1 and select a slice $A_k = [(a_k)_{ij}]$ in $\mathcal{A}$, for example, the first one. Set k_1 to 1 and let

$$\tilde{\mathcal{A}} = 0.$$

(1a) Find an approximation A_{k_p} to the k_p-th slice of the residue $\mathcal{R} = \mathcal{A} - \widetilde{A}_p$ by the cross-method:

$$A_{k_p} = \sum_{q=1}^{r} u_{pq} v_{pq}^{\top}.$$

(1b) Find the largest magnitude element in the matrix A_{k_p}, let it be located at (i_p, j_p).

(2) Compute the row

$$(w_p)_k = \mathcal{R}_{i_p, j_p, k}$$

corresponding to the index (i_p, j_p) and find its largest magnitude element from those whose index is not equal to k_p. [1] Suppose it is located at the k_{p+1}-th position of w_p. Perform the scaling:

$$w_p := w_p / w_{k_p p}.$$

(3) Compute a new approximation:

$$\tilde{\mathcal{A}} = \tilde{\mathcal{A}} + A_{k_p} \otimes w_p = \tilde{\mathcal{A}} + \left(\sum_{q=1}^{r} u_{pq} v_{pq}^{\top} \right) \otimes w_p = \tilde{\mathcal{A}} + \sum_{q=1}^{r} u_{pq} \otimes v_{pq} \otimes w_p.$$

(4) If the stopping criterion is not satisfied, set $p := p + 1$, and go to step 1.

In the end, the array $\mathcal{A}$ is approximated by $\tilde{\mathcal{A}} = [\tilde{a}_{ijk}]$ having a Tucker-like decomposition (also viewed as a trilinear decomposition) of the form

$$\tilde{a}_{ijk} = \sum_{p=1}^{r} \left(\sum_{q=1}^{r} u_{ipq} v_{jpq} \right) w_{kp} = \sum_{\alpha=1}^{r^2} u_{i\alpha} v_{j\alpha} w_{k\alpha}. \tag{4.2}$$

During the implementation of this method, we encounter several problems that should be solved with a linear complexity in n:

- Determine the largest magnitude element in a low-rank matrix (step **1**b);
- Estimate the quantities in the relationships

$$\|\mathcal{A} - \tilde{\mathcal{A}}\|_F \leqslant \varepsilon \|\mathcal{A}\|_F \approx \varepsilon \|\tilde{\mathcal{A}}\|_F$$

so that to have a sound stopping criterion (step **4**).

The first problem is not trivial and we do not know if there is an exact and fast way to find a maximal element in a low-rank matrix. However, we are able to design a heuristic algorithm, based on the submatrix of maximal volume. It manifests a very good practical performance (see Appendix).

The stopping criterion in the 3D-Cross method is identical to the 2D case, by the comparison of the approximant norm and the norm of a newly computed cross-correction. The norm $\|\tilde{\mathcal{A}}\|_F$ is computed by the formulas

$$\begin{aligned} \|\tilde{\mathcal{A}}\|_F^2 &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \left(\sum_{\alpha=1}^{r^2} u_{i\alpha} v_{j\alpha} w_{k\alpha} \right)^2 = \\ &= \sum_{\alpha=1}^{r^2} \sum_{\alpha'=1}^{r^2} (u_\alpha, u_{\alpha'})(v_\alpha, v_{\alpha'})(w_\alpha, w_{\alpha'}). \end{aligned}$$

The cost of Algorithm 2 is $\mathcal{O}(nr^2)$ evaluations of the elements of $\mathcal{A}$ plus $\mathcal{O}(nr^4)$ arithmetic operations (at each "outer" step of the method we compute a new slice from which we should subtract the elements of the previously computed approximation, and that results in a relatively big constant r^4 at the size n). This is already a linear complexity. However, we are going to present a "clever" implementation with a significantly better performance.

[1] It is the worth to note we can not use also elements with indices $k_1, ..., k_{p-1}$ but it can be verified that they are all zeroes, so they can not have maximal modulus.

4.4. The 3D-cross algorithm. We can improve the efficiency of Algorithm 2 by using a more compact way to store and handle the slices A_{k_p} so that the required number of vectors to represent them is reduced from $\mathcal{O}(r^2)$ to $\mathcal{O}(r)$.

At each step we approximate the computed slices A_{k_p} in the format

$$A_{k_p} = UB_pV^\top, \tag{4.3}$$

where $n \times r$ matrices U, V are orthogonal and the core matrices B_p are $r \times r$. It is worth to note that the equation (4.3) is also known as a Tucker2 decomposition, where only 2 of 3 modes are compressed. The storage for p slices is now $2nr + pr^2$ which is asymptotically equal to $\mathcal{O}(nr)$. The existence of matrices U, V, follows from the existence of a "good" Tucker approximation. In fact, we can try U and V as the Tucker factors. The computation of this simultaneous matrix decomposition is equivalent to the computation of the Tucker decomposition of a $n \times n \times p$ array

$$\mathcal{A}' = [A_{k_1} \dots A_{k_p}],$$

Indeed, if

$$a_{ijk_p} = \sum_{i'=1}^{r} \sum_{j'=1}^{r} \sum_{p'=1}^{r} g_{i'j'p'} u_{ii'} v_{jj'} w_{pp'},$$

then, setting

$$\sum_{p'=1}^{r} g_{i'j'p'} w_{pp'} = b_{i'j'p} = (b_p)_{i'j'},$$

we immediately arrive at (4.3).

Another important modification concerns the computation of the slices. Suppose the p steps are done and we are going to compute the $p+1$-th slice $A_{k_{p+1}}$. Instead of using the "full" cross method for this slice, we first find an approximation of the form

$$A_{k_{p+1}} \approx U\Phi V^\top,$$

where U, V come from (4.3) and a matrix Φ is $r \times r$. Such an approximation can be obtained quite cheaply by the following scheme:

- Find $r \times r$ *submatrices of maximal volume* in U and V. Suppose they have indices $i_1, ..., i_r$ and $j_1, ..., j_r$. Denote these submatrices by $\widehat{U}$ and $\widehat{V}$.
- Compute the $r \times r$ submatrix S in $A_{k_{p+1}}$ lying on the intersection of rows with indices $i_1, ..., i_r$ and columns with indices $j_1, ...j_r$.
- Compute

$$\Phi = \widehat{U}^{-1} S \widehat{V}^{-1}. \tag{4.4}$$

We can prove that this approximation approach is robust (see Appendix B). After Φ is computed, we check the approximation error by taking some random samples of a true matrix A_{k_p}. If the approximation is not good enough, then we perform some steps of the cross approximation algorithm, starting from a good approximation. However, as a rule, only a few steps (or even none) of the cross algorithm are required.

Algorithm 3 (Cross 3D). Suppose an $n \times n \times n$ three-way array $\mathcal{A}$ is given.

(0) Perform one step of Algorithm 2 (with $p = 1$). Upon completion, $p = 2$ and $\mathcal{A}$ is represented as

$$\tilde{A} = \left(\sum_{q=1}^{r} u_{1q} v_{1q}^{\top} \right) \otimes w_1 = \sum_{q=1}^{r} u_{1q} \otimes v_{1q} \otimes w_1.$$

Compute orthogonal U, V by the two QR-decompositions

$$U_1 = U R_u, \qquad V_1 = V R_v.$$

Then $\mathcal{A}$ is represented as

$$\tilde{A} = (U R_u R_v^{\top} V^{\top}) \otimes w_1 = (U B_1 V^{\top}) \otimes (w_1 / \|w_1\|_2),$$

$$B_1 = R_u R_v^{\top} \|w_1\|_2.$$

Set $w_1 := w_1 / \|w_1\|$,. Note that we will normalize all computed vectors u_p, v_p, w_p. In the vector w_1 compute the largest magnitude element, suppose it has index k_2.

(1.1) Compute Φ from (4.4). If necessary, perform some additional steps of the cross method to obtain an approximation to the slice A_{k_p}.

$$A_{k_p} \approx \tilde{A}_{k_p} = U \Phi V^{\top} + \sum_{q=1}^{r_1} u_{pq} v_{pq}^{\top}.$$

(Note that r_1 is supposed to be small even compared to r).

(1.2) Add new vectors u_{pq}, v_{pq}, $q = 1, \ldots, r$, to basises U, V and orthogonalize the extended matrices $[UU_p]$, $[VV_p]$

$$[UU_p] = [U\hat{U}_p] \begin{bmatrix} I & M_u \\ 0 & R_u \end{bmatrix}, \quad [VV_p] = [V\hat{V}_p] \begin{bmatrix} I & M_v \\ 0 & R_v \end{bmatrix},$$

$$U^{\top}\hat{U}_p = 0, \qquad V^{\top}\hat{V}_p = 0 \qquad ;U_p^{\top}\hat{U}_p = I \qquad ;V_p^{\top}\hat{V}_p = I.$$

(1.3) Compute new $(r + r_1) \times (r + r_1)$ core B_p

$$B_p = \begin{bmatrix} M_u \\ R_u \end{bmatrix} \left[M_v^{\top} R_v^{\top} \right].$$

Other slices in new basis have the form

$$A_{k_q} = [U\hat{U}_p] \begin{bmatrix} B_q & 0 \\ 0 & 0 \end{bmatrix} [V\hat{V}_p]^{\top}, \qquad q = 1, \ldots, p-1.$$

Therefore, approximation $\mathcal{A}_{p-1}$ is represented as

$$\mathcal{A}_{p-1} = \left(\sum_{q=1}^{p-1} U' B'_q V'^{\top} \right) \otimes w_{\beta},$$

$$U' = [U\hat{U}_p], \quad V' = [V\hat{V}_p], \qquad B'_q = \begin{bmatrix} B_q & 0 \\ 0 & 0 \end{bmatrix}, \quad q = 1, \ldots, p-1.$$

Set also $B'_p = B_p$.

(1.4) In the new slice A_{k_p} the largest magnitude element is found. Suppose it is located in (i_p, j_p).

(2.1) The fiber w_p, of the residue $\mathcal{A} - \tilde{\mathcal{A}}_{p-1}$ corresponding to (i_p, j_p) is computed.

(2.2) Vector w_p is orthogonalized to vectors $W = [w_1, \ldots, w_{p-1}]$

$$w_p = \sum_{q=1}^{p-1} c_q w_q + \hat{w}_p, \qquad \omega_q^\top \hat{w}_p = 0, \quad q = 1, \ldots, p-1.$$

Cores of "old" slices B'_q, $q = 1, \ldots p-1$, are modified

$$B''_q = B'_q + c_q B'_p, \qquad q = 1, \ldots, p-1,$$

vector $\hat{w}_p$ is normalized

$$w_p := \hat{w}_p / \|\hat{w}_p\|_2, \qquad B''_p = \|\hat{w}_p\|_2 B'_p.$$

(3) The approximation $\tilde{\mathcal{A}}_p$ is represented as

$$\tilde{\mathcal{A}}_p = \left(\sum_{q=1}^{p} U' B''_q V'^\top \right) \otimes w_q. \tag{4.5}$$

To reduce the sizes of the matrices B''_q (they are $(r+r_1) \times (r+r_1)$), we apply the Tucker reduction method:

(3.1) Create a three-way array $(r+r_1) \times (r+r_1) \times p$ $\mathcal{B}'' = [B''_1 \ldots B''_p]$ and compute its Tucker decomposition.

$$b''_{ijk} = \sum_{i'=1}^{r} \sum_{j'=1}^{r} \sum_{k'=1}^{r} g^{\clubsuit}_{i'j'k'} u^{\clubsuit}_{ii'} v^{\clubsuit}_{jj'} w^{\clubsuit}_{kk'}.$$

If we introduce

$$\sum_{k'=1}^{r} g^{\clubsuit}_{i'j'k'} w^{\clubsuit}_{kk'} = b^{\clubsuit}_{i'j'k} = (b^{\clubsuit}_k)_{i'j'},$$

the we have

$$B''_k = U_{\clubsuit} B_{\clubsuit k} V_{\clubsuit}^\top, \qquad k = 1, \ldots, p. \tag{4.6}$$

where matrices $U_{\clubsuit}$ and $V_{\clubsuit}$ are $(r + r_1) \times r$, cores $B_{\clubsuit k}$ are $r \times r$.

(3.2) Substituting (4.6) into (4.5), we obtain that

$$\tilde{\mathcal{A}}_p = \left(\sum_{k=1}^{p} (U' U_{\clubsuit}) B_{\clubsuit k} (V' V_{\clubsuit})^\top \right) \otimes w_k = \left(\sum_{k=1}^{p} U B_k V^\top \right) \otimes w_k, \tag{4.7}$$

where $U = U' U_{\clubsuit}$, $V = V' V_{\clubsuit}$ are orthogonal, cores $B_k = B_{\clubsuit k}$ are $r \times r$. The format (4.3) is restored.

(4) Check stopping criteria, if it is not satisfied, go to **1.1**.

This is the final version of Cross-3D. The numerical complexity of the method is $\mathcal{O}(nr)$ evaluations of the array elements and $\mathcal{O}(nr^3)$ additional operations.

5. Numerical experiments. We illustrate the performance of our algorithm on some model tensors which allow good low-rank approximation.

Specifically, we consider the following two types of arrays:

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n,$$

$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2+j^2+k^2}}, \qquad 1 \leqslant i,j,k \leqslant n.$$

The rank estimates obtained in [15, 19, 20] have form

$$r \leqslant C(\log n \log^2 \varepsilon),$$

where ε is an error of the approximation, so the rank grows only logarithmically with n and ε.

These two examples arise from the numerical solution of the integral equations. For example, the array $\mathcal{B}$ is obtained from the integral equation with kernel $\frac{1}{||x-y||}$ acting on a unit cube and disretizied by the Nyström method on a uniform grid.

Table 5.1 shows the ranks, accuracies and size of the computed Tucker approximation for the array $\mathcal{A}$, Table 5.2 shows the same for $\mathcal{B}$. The accuracy of the approximation was computed by sampling the elements of the array, since it is not possible to check all the elements for large n. The size of the sample was determined by the following rule: if the sample size was doubled, the estimated error should change by no more than 10%. As it can be seen, the approximation method is robust and leads to astonishing memory savings: the arrays that would need in the full format an enormous storage of 2 petabytes ($2 \cdot 2^{50}$ PB) are compressed to the sizes of 100 MB.

Moreover, our algorithm works with arrays on this huge scale on a personal workstation. The timings made on a personal computer `Pentium-4` with 3.4 Ghz clock are shown on Fig. 5.1. This figure confirms that the approximation time is almost linear in n. The somewhat irregular behavior on this plots is caused by the effects of caching (for small n) and by some rank overestimation by the stopping criteria for large n.

TABLE 5.1
Numerical results

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n$$

Rank and accuracy of the decomposition.

ε	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	5	$2.5_{10}-4$	8	$2.3_{10}-6$	10	$1.4_{10}-8$	12	$3.1_{10}-10$
128	6	$6.8_{10}-4$	8	$4.4_{10}-6$	11	$3.1_{10}-8$	13	$6.4_{10}-10$
256	6	$8.8_{10}-4$	9	$3.9_{10}-6$	12	$5.4_{10}-8$	15	$3.0_{10}-10$
512	7	$7.4_{10}-4$	10	$1.4_{10}-6$	13	$6.8_{10}-8$	16	$5.2_{10}-10$
1024	7	$5.7_{10}-4$	11	$4.8_{10}-6$	14	$4.0_{10}-8$	18	$2.7_{10}-10$
2048	7	$6.6_{10}-4$	12	$2.0_{10}-6$	16	$4.0_{10}-8$	19	$4.0_{10}-10$
4096	8	$3.2_{10}-4$	12	$6.3_{10}-6$	17	$3.4_{10}-8$	21	$3.5_{10}-10$
8192	8	$6.3_{10}-4$	13	$3.3_{10}-6$	18	$1.9_{10}-8$	22	$4.5_{10}-10$
16384	9	$7.9_{10}-4$	14	$3.5_{10}-6$	19	$7.2_{10}-8$	24	$5.6_{10}-10$
32768	9	$6.4_{10}-4$	14	$8.8_{10}-6$	20	$5.2_{10}-8$	25	$3.8_{10}-10$
65536	9	$4.0_{10}-4$	15	$6.3_{10}-6$	21	$2.5_{10}-8$	26	$5.2_{10}-10$

Rank and size (MB) of the Tucker decomposition
The sizes smaller than 1MB are not shown.

ε		$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2Mb	5		8		10		12	
128	16Mb	6		8		11		13	
256	128Mb	6		9		12		15	
512	1Gb	7		10		13		16	
1024	8Gb	7		11		14		18	
2048	64Gb	7		12		16		19	
4096	512Gb	8	¡1	12	1.1	17	1.6	21	2
8192	4Tb	8	1.5	13	2.5	18	3.5	22	4.2
16384	32Tb	9	3.4	14	5.25	19	7.2	24	9
32768	256Tb	9	6.75	14	10.5	20	15	25	19
65536	2Pb	9	13.5	15	22	21	31	26	39

TABLE 5.2
Numerical results

$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2 + j^2 + k^2}}, \qquad 1 \leqslant i, j, k \leqslant n$$

Rank and size (MB) of the Tucker decomposition.
Values less than 1MB are not shown.

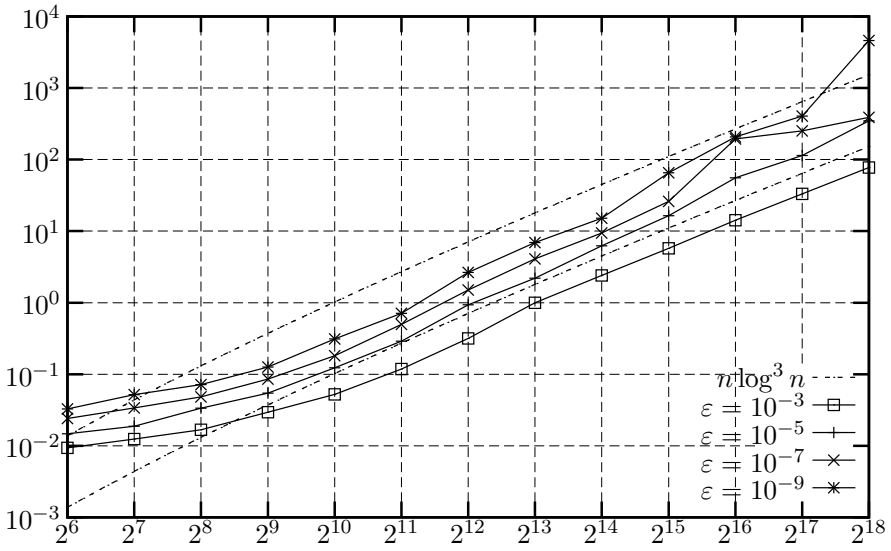
ε	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	7	$3.7_{10}-4$	11	$3.9_{10}-6$	14	$5.7_{10}-8$	18	$2.2_{10}-10$
128	8	$5.1_{10}-4$	12	$5.9_{10}-6$	17	$2.0_{10}-8$	20	$5.6_{10}-10$
256	9	$4.1_{10}-4$	14	$6.4_{10}-6$	19	$3.4_{10}-8$	23	$4.5_{10}-10$
512	10	$4.9_{10}-4$	15	$6.7_{10}-6$	21	$2.9_{10}-8$	26	$3.2_{10}-10$
1024	10	$5.5_{10}-4$	17	$3.2_{10}-6$	23	$3.9_{10}-8$	29	$4.7_{10}-10$
2048	11	$5.0_{10}-4$	18	$5.2_{10}-6$	25	$6.8_{10}-8$	31	$5.9_{10}-10$
4096	12	$8.4_{10}-4$	19	$4.2_{10}-6$	27	$3.5_{10}-8$	34	$3.3_{10}-10$
8192	12	$6.8_{10}-4$	20	$6.0_{10}-6$	28	$5.8_{10}-8$	36	$3.6_{10}-10$
16384	13	$2.7_{10}-4$	22	$4.8_{10}-6$	31	$5.6_{10}-8$	39	$2.6_{10}-10$
32768	13	$8.5_{10}-4$	23	$6.1_{10}-6$	32	$7.1_{10}-8$	41	$5.5_{10}-10$
65536	14	$6.2_{10}-4$	24	$6.5_{10}-6$	34	$7.8_{10}-8$	44	$1.4_{10}-9$

Rank and accuracy of the Tucker decomposition.
Values less than 1MB are not shown

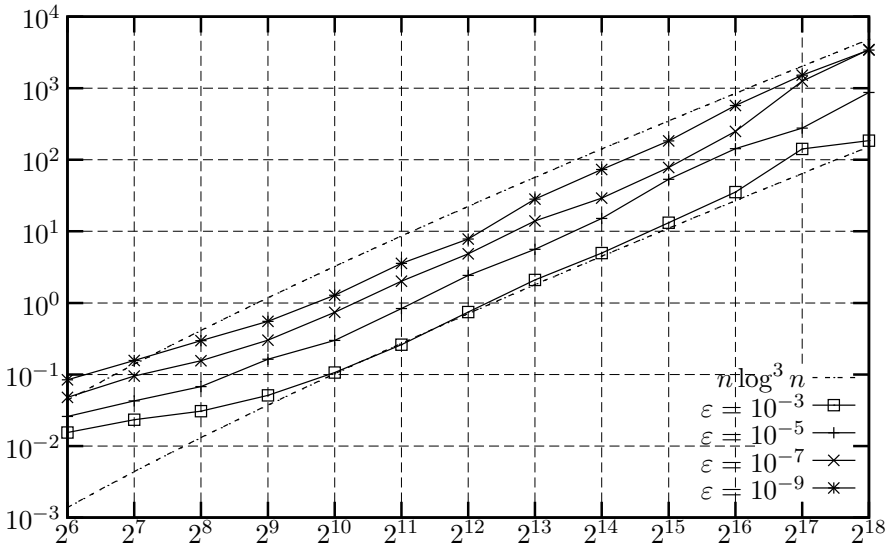
ε		$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2Mb	7		11		14		18	
128	16Mb	8		12		17		20	
256	128Mb	9		14		19		23	
512	1Gb	10		15		21		26	
1024	8Gb	10		17		23		29	
2048	64Gb	11	¡1	18	¡1	25	1.18	31	1.46
4096	512Gb	12	1.15	19	1.78	27	2.54	34	3.2
8192	4Tb	12	2.25	20	3.75	28	5.3	36	6.8
16384	32Tb	13	4.9	22	8.25	31	11.7	39	14.7
32768	256Tb	13	9.75	23	17.25	32	24	41	31
65536	2Pb	14	21	24	36	34	51	44	66

FIG. 5.1. *Approximation time, sec.*

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n$$



$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2+j^2+k^2}}, \qquad 1 \leqslant i,j,k \leqslant n$$



Two dense tensors considered come from a simple disretization of integral equations. Despite their "regularity" they are quite representative: in more complex cases our method behaves similarly. In other areas where tensor decomposition is used the

researchers often obtain more irregular and possibly sparse tensors. We want to note that sparseness is *ideal* for the Cross-3D because in that case the residue can be measured *exactly* and the pivots during the cross approximation stage can be also found exactly, leading to a *theoretically robust* method. The applications of the 3D-cross approach to more complex tensors will be reported elsewhere.

Appendix A: How to find the maximal element in a slice. One of the important ingredients of the 3D-cross method is the determination of the maximal element in a given low-rank matrix in linear time.

Suppose we have computed a skeleton approximation to a low-rank matrix

$$A = UV^{\top},$$

where U, V are $n \times r$, and we want to find the largest magnitude element in it. This problem can be solved by comparing all the elements of the matrix, but it costs $\mathcal{O}(n^2)$ operations. The proposed algorithm is based on the following hypothesis.

Hypothesis. Consider $r \times r$ submatrices in a rank-r matrix A. Let B be a submatrix of maximal volume among all such submatrices. Then

$$||B||_C \geqslant \frac{||A||_C}{r}.$$

So the maximal element in the submatrix of maximal volume can not be very much different from the maximal element in the whole matrix A.

How to determine a submatrix of maximal volume? This submatrix of A lies on the intersection of rows $i_1, \ldots, i_r$, coinciding with rows which contain the submatrix of maximal volume in U, and columns $j_1, \ldots, j_r$, which contain the submatrix of maximal volume in $V^{\top}$. To find the submatrix of maximal volume in a $n \times r$ matrix we will use the algorithm, proposed in [12]. For the readers convenience we describe it below.

Algorithm 4. Suppose U is $n \times r$. and its $r \times r$ submatrix with maximal volume is needed.

(0) Let A_γ be a leading submatrix. In the beginning set A_γ to any nonsingular submatrix of A and permute the rows so that A_γ is located in the first r rows.

(1) Compute

$$AA_\gamma^{-1} = \begin{bmatrix} I_{r\times r} \\ Z \end{bmatrix} = B.$$

(2) Find the largest magnitude element $|z_{ij}|$ in Z.

(3) **If** $\gamma = |z_{ij}| > 1$, **then**

Permute in B rows $r + i$ and j. The upper submatrix in B after the permutation has the form

$$\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ * & * & \gamma & * & * & \\ & & & \ddots & & \\ & & & & 1 & \end{pmatrix}$$

and its determinant is equal to $\gamma \geqslant 1 + \varepsilon$, i.e. it increased. Denote by A_γ the new submatrix in the first r rows of A and return to step **1.**

Otherwise terminate the algorithm.

In practice, to avoid huge number of transpositions a more "soft" stopping criteria is used in step **3**. The algorithm stops if $|z_{ij}| \leqslant 1+\nu$, where ν is a some small parameter.

Appendix B: The $U\Phi V^\top$ decomposition. In this appendix we will prove that the usage of (4.4) for the construction of the low-rank approximation to a slice is "legal".

THEOREM 5.1. *Suppose A is a $n_1 \times n_2$ matrix, U, V are $n_1 \times r_1$ and $n_2 \times r_2$ matrices with orthogonal columns and there exists matrix Φ such that*

$$A = U\Phi V^\top + E, \quad ||E||_F \leqslant \varepsilon.$$

Then, if we compute Φ' by the formula (4.4) *then*

$$||A - U\Phi' V^\top||_F \leqslant \sqrt{n_1 r_1 n_2 r_2}\varepsilon.$$

Proof.

If $\widehat{U}$ and $\widehat{V}$ are submatrices of maximal volume in U and V respectively and $\widehat{A}$ is a submatrix in A lying on the intersection of the selected rows from U and columns from $V^\top$ then

$$\widehat{A} = \widehat{U}\Phi\widehat{V}^\top + \widehat{E},$$

where $\widehat{E}$ is a submatrix of E occupying the same positions in E as $\widehat{A}$ in A.

$$||\Phi - \Phi'|| \leqslant ||\widehat{U}^{-1}||\ ||\widehat{E}||\ ||\widehat{V}^{-1}||.$$

The norms $\widehat{U}^{-1}, \widehat{V}^{-1}$ can be estimated as follows. We know that the elements of

$$U\widehat{U}^{-1}$$

are not greater than 1 in modulus (because $\widehat{U}$ is a submatrix of maximal volume). Therefore,

$$||\widehat{U}^{-1}||_F \leqslant \sqrt{n_1 r_1}.$$

Using this estimate we immediately complete the proof.

6. Acknowledgements. We are very grateful to both of the referees of our paper. The remark of one of the referees helped us to discover a nasty bug in the program code.

REFERENCES

[1] R. A. Harshman, Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis, *UCLA Working Papers in Phonetics.* V. 16. P. 1-84(1970).
[2] Three-mode Company, `three-mode.leidenuniv.nl`
[3] P. Comon, Tensor decomposition: State of the Art and Applications, *IMA Conf. Math. in Signal Proc.* Warwick, UK, Dec. 18-20, 2000.
`http://www.i3s.fr/~comon/FichiersPs/ima2000.ps`
[4] J. D. Caroll, J. J. Chang, Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition, *Psychometrica.* V.35. P. 283-319(1970).
[5] R. Bro, PARAFAC: Tutorial and applications *Chemom. Intelligent Lab. Systems.*, V. 38. pp. 149-171 (1997).
[6] G. Beylkin, M.M. Mohlenkamp, Numerical operator calculus in higher dimensions, *PNAS*, V. 99, No. 16, pp. 10246-10251(2002)
[7] M. A. O. Vasilescu, D. Terzopoulos, Multilinear Image Analysis for Facial Recognition, Proc. of Int. Conf. on Pattern Recognition (ICPR 2002), V. 2, Quebec City, Canada, Aug, pp. 511-514 (2002).

[8] M. Bebendorf, Approximation of boundary element matrices, *N*umer. Math., V. 86, No. 4, P. 565–589 (2000).

[9] J. M. Ford, E. E. Tyrtyshnikov, Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related systems, *SIAM J. Sci. Comp.*, V. 25, No. 3. P. 961–981 (2003).

[10] L. De Lathauwer, B. De Moor and J. Vandewalle, A multilinear singular value decomposition, *SIAM J. Matrix Analysis Appl.*, 21, pp. 1253–1278 (2000).

[11] L. De Lathauwer, B. De Moor and J. Vandewalle, On best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of high-order tensors, *SIAM J. Matrix Analysis Appl.*, 21, pp. 1324–1342 (2000).

[12] S. A. Goreinov, *Pseudoskeleton approximations of block matrices generated by asymptotically smooth kernels*, Ph. D. Thesis, INM RAS, 2001 (in Russian).

[13] S. A. Goreinov, E. E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, *Contemporary Mathematics*, V. 208, P. 47–51 (2001).

[14] S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin, A theory of pseudo–skeleton approximations, *Linear Algebra Appl.* 261, P. 1–21 (1997).

[15] W. Hackbusch, B. N. Khoromskij, E. E. Tyrtyshnikov, Hierarchical Kronecker tensor-product approximations, *J. Numer. Math.*, V. 13, P. 119–156 (2005).

[16] D. V. Savostianov, *Mosaic-skeleton approximations*, Master Thesis, INM RAS, 2001 (in Russian).

[17] E. E. Tyrtyshnikov, Mosaic–skeleton approximations, *Calcolo*, V. 33 (1-2), P. 47–57 (1996).

[18] E. E. Tyrtyshnikov, Incomplete cross approximation in the mosaic–skeleton method, *Computing*, V. 4, P. 367–380 (2000).

[19] E. E. Tyrtyshnikov, Kronecker-product approximations for some function-related matrices, *Linear Algebra Appl.*, V. 379, P. 423–437 (2004).

[20] E. E. Tyrtyshnikov, Tensor approximations of matrices generated by asymptotically smooth functions, *Sbornik: Mathematics* **194**, No. 5-6, 941–954 (2003).

[21] I. Ibraghimov, Application of the three-way decomposition for matrix compression, *Numer. Linear Algebra Appl.*, V.9, No. 6-7. P. 551–565 (2002).

[22] V. Olshevsky, I. V. Oseledets, E. E. Tyrtyshnikov, Tensor properties of multilevel Toeplitz and related matrices, *Linear Algebra Appl.* 412, P. 1–21 (2006).

[23] L. R. Tucker, Some mathematical notes on three-mode factor analysis, *Psychometrika*, V. 31, P. 279–311 (1966).

[24] P. Drineas, R. Kannan, and M. W. Mahoney, *SIAM J. Computing*, 36, 158-183 (2006).