

Institute for Computational Mathematics
Hong Kong Baptist University

ICM Research Report
08-10

How to find a good submatrix

S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov,
E. E. Tyrtyshnikov, N. L. Zamarashkin

November 5, 2008

Abstract

Pseudoskeleton approximation and some other problems require the knowledge of sufficiently well-conditioned submatrix in a large-scale matrix. The quality of a submatrix can be measured by modulus of its determinant, also known as volume. In this paper we discuss a search algorithm for the maximum-volume submatrix which already proved to be useful in several matrix and tensor approximation algorithms. We investigate the behavior of this algorithm on random matrices and present some its applications, including maximization of a bivariate functional.

1 Introduction

Several problems in matrix analysis require the knowledge of a good submatrix in a given (supposedly large) matrix. By “good” we mean a sufficiently well-conditioned submatrix. The application that we are particularly interested in is the approximation of a given matrix by a low-rank matrix:

$$A \approx UV^T,$$

where A is $m \times n$ and U and V are $m \times r$ and $n \times r$, respectively. Optimal approximation in spectral or Frobenius norm can be computed via singular value decomposition (SVD) which, however, requires too many operations. A much faster way is to use CGR decompositions [1] (later also referred to as CUR by some authors) which in Matlab notation can be written as:

$$A \approx A(:, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}A(\mathcal{I}, :), \quad (1)$$

where \mathcal{I}, \mathcal{J} are appropriately chosen index sets of length r from $1 : n$ and $1 : m$. It can be seen that the right hand side matrix coincides with A in r rows and r columns. Moreover, if A is strictly of rank r and $\hat{A} = A(\mathcal{I}, \mathcal{J})$ is nonsingular, the exact equality holds. However, in the approximate case the quality of the approximation (1) relies heavily on the “quality” of the submatrix. The question is how to measure this quality and how to find a good submatrix. A theoretical answer (basically, existence theory) [3] is that if \hat{A} has maximal in modulus determinant among all $r \times r$ submatrices of A , then element-wise error estimate is of the form

$$|A - A_r| \leq (r + 1)\sigma_{r+1},$$

where $|A| = \max_{ij} |a_{ij}|$ denotes Chebyshev norm, A_r is the right hand side of (1) and σ_{r+1} is the $r + 1$ -th singular value of the matrix A , i.e. the error of the best rank- r approximation in the spectral norm. That is the theory, but what about a practical algorithm? How to find a good submatrix? That is the topic of the current paper.

As we have seen, the submatrix quality can be measured by its determinant, so we want to find a submatrix with the largest possible determinant. An intermediate step to the solution of that problem is computation of the maximal volume submatrix not in a matrix where both dimensions are large, but in the matrix where only one dimension is large, i.e. in a “tall matrix”. Such procedure (called `maxvol`) plays a crucial role in several matrix algorithms we have developed, and it deserves a special description [2, 4]. In this paper we investigate the behavior of the `maxvol` algorithm on random matrices and present some theoretical results and its application for fast search of the maximum entry of large-scale matrix. We also propose a new approach for maximization of a bivariate functional on the base of `maxvol` algorithm.

1.1 Notation

In this article we use Matlab-like notation for defining rows and columns of matrix. Therefore we write i -th row of matrix A as $a_{i,:}$ and j -th column of A as $a_{:,j}$. We will also use columns and rows of identity matrix, denoting them as e_i and e_j^T respectively, using the same notations for different sizes, but the actual size will be always clear by the context.

1.2 Definitions and basic lemmas

Let us give some formal definitions and prove basic lemmas to rely on.

Definition 1. We refer to the modulus of determinant of square matrix as its volume.

Definition 2. We call $r \times r$ submatrix A_{\blacksquare} of rectangular $m \times n$ matrix A maximum volume submatrix, if it has maximum determinant in modulus among all possible $r \times r$ submatrices of A .

Definition 3. We call $r \times r$ submatrix A_{\square} of rectangular $n \times r$ matrix A of full rank dominant, if all the entries of AA_{\square}^{-1} are not greater than 1 in modulus.

The main observation that lays ground for the algorithms for the construction of `maxvol` algorithm is the following lemma.

Lemma 1. For $n \times r$ matrix maximum volume $r \times r$ submatrix is dominant.

Proof. Without loss of generality we can consider that A_{\blacksquare} occupies first r rows of A . Let us refer to them as *upper submatrix*. Then

$$AA_{\blacksquare}^{-1} = \begin{bmatrix} I_{r \times r} \\ Z \end{bmatrix} = B. \quad (2)$$

Multiplication by a nonsingular matrix does not change the ratio of determinants of any pair of $r \times r$ submatrices in A . Therefore, the upper submatrix $I_{r \times r}$ is a maximum-volume submatrix in B and it is dominant in B iff A_{\blacksquare} is dominant in A .

Now, if there is some $|b_{ij}| > 1$ in B , then we can construct a new submatrix with a volume larger than volume of the upper submatrix. To see that, swap rows i and j in B , and it is easy to see that a new upper submatrix

$$B_{\blacksquare'} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ * & * & b_{ij} & * & * & \\ & & & \ddots & & \\ & & & & & 1 \end{pmatrix} \quad (3)$$

has

$$|\det(B_{\blacksquare'})| = |b_{ij}| > 1 = |\det(I_{r \times r})|.$$

That means that $I_{r \times r}$ (and hence A_{\blacksquare}) is not the maximum volume submatrix. \square

The volume of a dominant submatrix can not be very much smaller than the maximum volume, as the following lemma shows.

Lemma 2. *For any nonsingular $n \times r$ matrix A*

$$|\det(A_{\square})| \geq |\det(A_{\blacksquare})|/r^{r/2} \quad (4)$$

for all dominant $r \times r$ submatrices of A .

Proof. Suppose that A_{\square} is the upper submatrix and write

$$AA_{\square}^{-1} = \begin{bmatrix} I_{r \times r} \\ Z \end{bmatrix} = B. \quad (5)$$

All entries of B are not greater than 1 in modulus, therefore by Hadamard inequality the volume of any $r \times r$ submatrix $B_{r \times r}$ of B is not greater than

$$|\det(B_{r \times r})| \leq \prod_{i=1}^r |b_{\sigma_i, :}| \leq r^{r/2},$$

where σ_i are indices of rows that contain $B_{r \times r}$. \square

The inequality is sharp. For example, if Z contains Fourier, Hadamard or Walsh matrix as a submatrix, it is easy to see that the equality is attained.

2 Algorithm maxvol

A dominant property of the maximal-volume submatrix allows us to construct a simple and efficient algorithm for the search of maximal volume submatrix.

Algorithm 1. *Given: $n \times r$ matrix A . Find: $r \times r$ dominant submatrix A_{\square} .*

0 *Start with arbitrary nonsingular $r \times r$ submatrix A_{\square} . Reorder rows in A so that A_{\square} occupies first r rows in A .*

1 *Compute*

$$AA_{\square}^{-1} = B$$

and find its maximal in modulus entry b_{ij} .

2 *If $|b_{ij}| > 1$, then*

swap rows i and j in B . Now, upper submatrix in B has the form (3) and the volume $|b_{ij}| \geq 1$. By swapping the rows we have increased the volume of the upper submatrix in B , as well as in A .

Let A_{\square} be the new upper submatrix of A and go to step 1.

If $|b_{ij}| = 1$, return $A_{\square} = A_{\square}$.

On each iterative step of Algorithm 1, volume of A_{\square} increases until the volume of A_{\square} is reached. In practice, we can simplify the stopping criterion in the iterative step to $|b_{ij}| < 1 + \delta$ with sufficiently small parameter δ (we think that $\delta \sim 10^{-2}$ can be a good choice). This dramatically reduces the number of iterative steps but does not change the “good” properties of a submatrix.

If computations proceed in a naive way, then the most expensive part of iterations is step 1, which needs one $r \times r$ matrix inversion and nr^2 operations for the matrix-by-matrix product AA_{\square}^{-1} . We can reduce the complexity of this step by a factor of r if we note that on each iteration, A_{\square} is updated by a rank-one matrix, and apply Sherman-Woodbury-Morrison formula for the matrix inverse. Now we describe this in detail.

Swapping of rows i and j of matrix A is equivalent to the following rank-one update.

$$A := A + e_j(\mathbf{a}_{i,:} - \mathbf{a}_{j,:}) + e_i(\mathbf{a}_{j,:} - \mathbf{a}_{i,:}) = A + (e_j - e_i)(\mathbf{a}_{i,:} - \mathbf{a}_{j,:}) = A + p\mathbf{v}^T. \quad (6)$$

For the upper submatrix, this update is

$$A_{\square} := A_{\square} + e_j(\mathbf{a}_{i,:} - \mathbf{a}_{j,:}) = A_{\square} + q\mathbf{v}^T. \quad (7)$$

For the inverse of the upper submatrix, we use the SWM formula

$$A_{\square}^{-1} := A_{\square}^{-1} - A_{\square}^{-1}q(1 + \mathbf{v}^T A_{\square}^{-1}q)^{-1}\mathbf{v}^T A_{\square}^{-1}. \quad (8)$$

Note that

$$\mathbf{v}^T A_{\square}^{-1}q = (\mathbf{a}_{i,:} - \mathbf{a}_{j,:})A_{\square}^{-1}e_j = ((AA_{\square}^{-1})_{i,:} - (AA_{\square}^{-1})_{j,:})e_j = b_{ij} - b_{jj} = b_{ij} - 1.$$

We proceed with the formula of fast update of $B = AA_{\square}^{-1}$,

$$\begin{aligned} B = AA_{\square}^{-1} &:= (A + p\mathbf{v}^T)(A_{\square}^{-1} - A_{\square}^{-1}q\mathbf{v}^T A_{\square}^{-1}/b_{ij}) = \\ &= AA_{\square}^{-1} - AA_{\square}^{-1}q\mathbf{v}^T A_{\square}^{-1}/b_{ij} + p\mathbf{v}^T A_{\square}^{-1} - p\mathbf{v}^T A_{\square}^{-1}q\mathbf{v}^T A_{\square}^{-1}/b_{ij} = \\ &= B - (Bq - b_{ij}p + p\mathbf{v}^T A_{\square}^{-1}q)\mathbf{v}^T A_{\square}^{-1}/b_{ij}. \end{aligned}$$

Using $v^T A_{\square}^{-1} = b_{i,:} - b_{j,:}$ and $v^T A_{\square}^{-1} q = b_{ij} - 1$, we have

$$B := B - (b_{:,j} - b_{ij}p + (b_{ij} - 1)p)(b_{i,:} - b_{j,:})/b_{ij},$$

and finally

$$B := B - (b_{:,j} - e_j + e_i)(b_{i,:} - e_j^T)/b_{ij}. \quad (9)$$

Note also that the upper $r \times r$ submatrix of B remains to be identity after each update, because $b_{1:r,j} = e_j$ for $j \leq r$ and $(e_i)_{1:r} = 0$ for $i > r$ that is always the case. So we need to update only the submatrix Z . This can be also done by a rank-one update:

$$Z := Z - (b_{:,j} + e_i)(b_{i,:} - e_j^T)/b_{ij}. \quad (10)$$

Note that in the last formula we use “old” indexing, i.e. rows of Z are numbered from $r + 1$ to n .

Therefore, each iterative step of the algorithm reduces to a rank-one update of Z which can be done in $(n - r)r$ operations, and a search for a maximum-modulus element in Z , which is of the same complexity. Overall complexity for the algorithm 1 is therefore $\mathcal{O}(nr^2)$ for initialization of Z and $\mathcal{O}(c nr)$ for iterative part, where c is the number of iterations. We can write a rather rough estimate for c as follows. Each iteration step increases volume of A_{\square} by a value $|b_{ij}| \geq 1 + \delta$. After k steps

$$|\det(A_{\square}^{[k]})| \geq |\det(A_{\square}^{[0]})|(1 + \delta)^k,$$

therefore

$$c \leq \left(\log |\det(A_{\blacksquare})| - \log |\det(A_{\square}^{[0]})| \right) / \log(1 + \delta). \quad (11)$$

This shows that good initial guess for A_{\square} can reduce the number of iterations. If no “empirical” guesses are available, it is always safe to apply Gaussian elimination with pivoting to A and use the set of pivoted rows as an initial approximation to the maximal volume submatrix.

3 maxvol-based maximization methods

As an application consider the following simple and interesting problem: find maximum in modulus element of a low-rank matrix $A = UV^T$, given by U and V . This problem arises for example in maximization of two-dimensional separable function on a grid, or as an essential part of the Cross3D algorithm for computation of Tucker approximation of three dimensional tensor in linear time [4]. Direct comparison of all elements requires nm operations for $m \times n$ matrix of rank r . Is it possible to devise an algorithm with complexity linear in matrix size?

3.1 Theoretical estimates

Our idea is not to search for maximum element in the whole submatrix, but only in the submatrix of maximal volume. Though looking not very natural at the first glance,

this algorithm actually works well in many cases. Often the maximal element in the maximal volume submatrix is not necessarily the same as the true maximal element, but it can not be very much smaller (for example, if it zero, then the submatrix of maximal volume is zero and the matrix is also zero, which we hope is not true). But are there any quantitative estimates? In fact, we can replace maximal-volume submatrix by an arbitrary dominant submatrix, which yields the same estimate. But first we need to extend the definition of the dominant submatrix to the case of $m \times n$ matrices. It is done in a very simple manner.

Definition 4. We call $r \times r$ submatrix A_{\square} of rectangular $m \times n$ matrix A dominant, if it is dominant in columns and rows that it occupies in terms of Definition 3.

Theorem 1. If A_{\square} is a dominant $r \times r$ submatrix of a $m \times n$ matrix A of rank r , then

$$|A_{\square}| \geq |A|/r^2. \quad (12)$$

Proof. If maximum in modulus element b of A belongs to A_{\square} , the statement is trivial. If not, consider $(r+1) \times (r+1)$ submatrix, that contains A_{\square} and b ,

$$\hat{A} = \begin{bmatrix} A_{\square} & c \\ d^T & b \end{bmatrix}. \quad (13)$$

Elements of vectors c and d can be bounded as follows

$$|c| \leq r|A_{\square}|, \quad |d| \leq r|A_{\square}|. \quad (14)$$

This immediately follows from $c = A_{\square}(A_{\square}^{-1}c) = A_{\square}\tilde{c}$, where all elements of \tilde{c} are not greater than 1 in modulus. Bound for elements of d is proved in the same way.

Now we have to bound $|b|$. Since A has rank r and A_{\square} is nonsingular,

$$b = d^T A_{\square}^{-1} c, \quad (15)$$

and it immediately follows that

$$|A| = |b| \leq |d|r \leq |A_{\square}|r^2,$$

which completes the proof. □

The restriction $\text{rank } A = r$ may be removed with almost no change in the bound (12). However, one has to replace A_{\square} by A_{\blacksquare} .

Theorem 2. If A_{\blacksquare} is maximum-volume $r \times r$ (nonsingular) submatrix of $m \times n$ matrix A , then

$$|A_{\blacksquare}| \geq |A|/(2r^2 + r).$$

Proof. Again, consider submatrix \hat{A} that contains A_{\blacksquare} and b , see (13). Bound (14) follows immediately, because the maximum-volume submatrix is dominant, see Lemma 1.

Since $\text{rank } A$ is now arbitrary, the equality (15) is no longer valid. Instead, we use an inequality from [3],

$$|\mathbf{b} - \mathbf{d}^\top \mathbf{B}^{-1} \mathbf{c}| \leq (r+1) \sigma_{r+1}(\hat{A}), \quad (16)$$

where $\sigma_1(\hat{A}) \geq \sigma_2(\hat{A}) \geq \dots \geq \sigma_{r+1}(\hat{A})$ are singular values of \hat{A} . That gives

$$\begin{aligned} |\mathbf{b}| &\leq (r+1) \sigma_{r+1}(\hat{A}) + |\mathbf{d}^\top \mathbf{A}_{\blacksquare}^{-1} \mathbf{c}| \leq (r+1) \sigma_{r+1}(\hat{A}) + |\mathbf{d}^\top \tilde{\mathbf{c}}| \leq \\ &\leq (r+1) \sigma_{r+1}(\hat{A}) + |\mathbf{d}| r \leq (r+1) \sigma_{r+1}(\hat{A}) + |\mathbf{A}_{\blacksquare}| r^2. \end{aligned} \quad (17)$$

We need an estimate for $\sigma_{r+1}(\hat{A})$ in terms of values of its elements. Note that

$$\hat{A}^\top \hat{A} = \begin{bmatrix} \mathbf{A}_{\blacksquare}^\top & \mathbf{d} \\ \mathbf{c}^\top & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{\blacksquare} & \mathbf{c} \\ \mathbf{d}^\top & \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\blacksquare}^\top \mathbf{A}_{\blacksquare} + \mathbf{d} \mathbf{d}^\top & \mathbf{A}_{\blacksquare}^\top \mathbf{c} + \mathbf{b} \mathbf{d} \\ \mathbf{c}^\top \mathbf{A}_{\blacksquare} + \mathbf{b} \mathbf{d}^\top & \mathbf{c}^\top \mathbf{c} + \mathbf{b}^2 \end{bmatrix}.$$

From the singular value interlacing theorem,

$$\sigma_r(\mathbf{A}_{\blacksquare}^\top \mathbf{A}_{\blacksquare} + \mathbf{d} \mathbf{d}^\top) \geq \sigma_{r+1}^2(\hat{A}),$$

and for $r > 1$

$$\sigma_{r-1}(\mathbf{A}_{\blacksquare}^\top \mathbf{A}_{\blacksquare}) \geq \sigma_r(\mathbf{A}_{\blacksquare}^\top \mathbf{A}_{\blacksquare} + \mathbf{d} \mathbf{d}^\top) \geq \sigma_{r+1}^2(\hat{A}).$$

Finally we have $\sigma_1(\mathbf{A}_{\blacksquare}) \geq \sigma_{r+1}(\hat{A})$ and $|\mathbf{A}_{\blacksquare}| \geq \sigma_1(\mathbf{A}_{\blacksquare})/r$. Plugging this into (17), we get

$$|\mathbf{b}| \leq (r+1)r|\mathbf{A}_{\blacksquare}| + r^2|\mathbf{A}_{\blacksquare}| = (2r^2 + r)|\mathbf{A}_{\blacksquare}|,$$

which completes the proof. \square

Now it is clear that we can reduce the search to only r^2 elements of the dominant matrix. Then the search time does not depend on matrix size, and the total complexity is just the complexity of finding A_{\square} , which is $\mathcal{O}(nr^2 + mr^2)$ operations. Maximum element in A_{\square} is “sufficiently good” in the sense of proven theorems.

In practical cases, the ratio $|A|/|A_{\square}|$ is sufficiently smaller than r^2 . Consider two examples, which illustrate this fact.

3.2 Search for the maximum element in random low-rank matrices

In order to see how good is the maximal element in our “good” submatrix, we tested it first on random matrices. Given n, m, r , two matrices U and V were generated with elements uniformly distributed in $[-1 : 1]$. Then U, V were replaced with Q -factors of their QR-decompositions and a matrix $A = UDV^\top$ was generated with random positive diagonal D with elements uniformly distributed on $[0, 1]$. We generated a large set of trial matrices, for each of these matrices we computed maximal element using the proposed algorithm.

The actual degradation of the maximal element is presented on the Figure 1, where the histogram of the ratio of maximal element in A_{\square} over the true maximal element is given. Note that this ratio for certain is not lower than 0.5 for all trials (smooth humps in the middle part of histograms), and in some 5% of cases (sharp peaks in the right part of histograms) we even found a true maximal element, which was much less probable for a random choice.

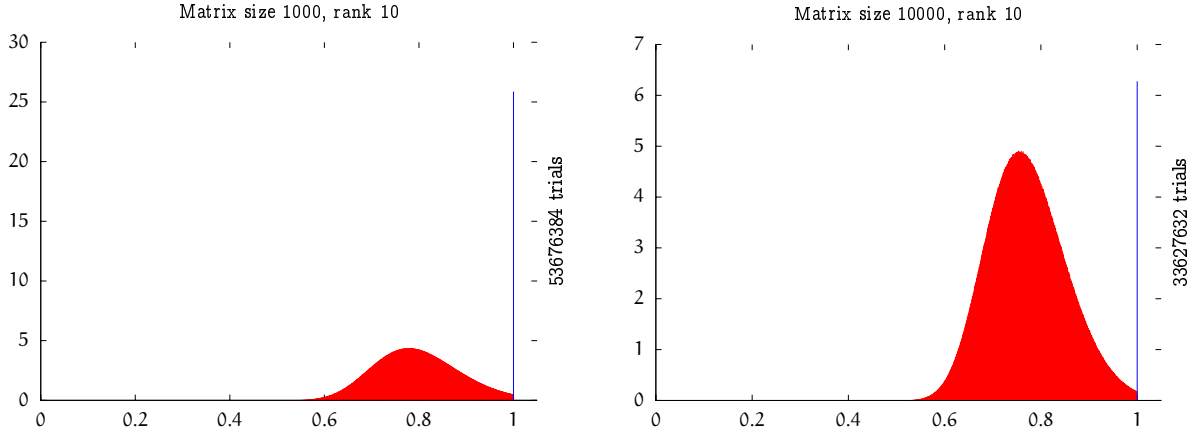


Figure 1: Distribution of the ratio of maxvol over true maximal element

3.3 Maximization of bivariate functions

There is an interesting application of our algorithm. It can be applied to the problem of global optimization of bivariate functions. Suppose we want to find a maximum of $|f(x, y)|$ in some rectangle $(x, y) \in \Pi = [a_0, a_1] \times [b_0, b_1]$, and f is some given function. “Discretizing” the problem on some sufficiently fine grid (x_i, y_j) , $i = 1, 2, \dots, m$, $j = 1, 2, \dots, n$ we obtain an $m \times n$ matrix $A = [f(x_i, y_j)]$ to find the maximal in modulus element in. Assume additionally that the function $f(x, y)$ can be sufficiently well approximated by a sum of separable functions:

$$f(x, y) \approx \sum_{\alpha=1}^r u_{\alpha}(x)v_{\alpha}(y).$$

Then it easy to see that in this case the matrix A admits a rank- r approximation of the form

$$A = f(x_i, y_j) \approx UV^T,$$

where U, V are $n \times r$ and $m \times r$ matrices, respectively, with elements $U = [u_{\alpha}(x_i)]$, $V = [v_{\alpha}(y_j)]$. Thus the “discretized” problem is equivalent to the problem of finding maximal in modulus element in a large low-rank matrix A , so we can apply our method. We have no guarantee that we will find the exact maximum, but we will have an estimate of it. As an example we considered a standard banana function minimization problem:

$$b(x, y) = 100(y - x)^2 + (1 - x)^2.$$

This function has minimum in $(1, 1)$ equal to 0 and is positive in all other points. In order to reformulate the problem as a maximization problem, we introduce an auxiliary function

$$f(x, y) = \frac{1}{b(x, y) + 10^{-6}},$$

the maximum of which is located at the same point $(1, 1)$. A rectangle $[-2, 2] \times [2, 2]$ was chosen and discretized on a 500×500 uniform grid, the corresponding matrix A was approximated by a matrix of rank 10 for which the maximum was found by our

maxvol algorithm. The extremal point was contained in the grid, and the maxvol returned the exact position of the minimum: $(1, 1)$. For other choices of grids the situation was the same, and the approximations to the extremum were very good (the error was $\mathcal{O}(h)$, where h is a grid size).

This result is very encouraging. However, it should not be treated as a universal optimization method, but it can be very useful in global optimization methods, because it gives us an estimate of the value of the global optimum — this can be efficiently used, for example, in branch-and-bound methods, with maxvol estimates for the maximal value in a particular domain. Another possibility is to use “local” separable approximations to functions and then minimize this local part by the maxvol algorithm. Incorporation of our method into robust optimization methods will be the subject of future research.

4 Conclusion and future work

In this paper we presented a simple iterative method for the search of a submatrix of maximal volume in a given rectangular matrix. This submatrix plays an important role in the theory and algorithms for the approximation by low (tensor) rank matrices. As an application, we constructed an algorithm for the computation of maximal in modulus element in a given low-rank matrix and proved, that the element can not be much smaller than the “true” maximal element. Experiments on random matrices prove that our algorithm performs very good, as well as the experiment with the minimization of the banana function. A future work will be focused on maximizing separable functions by using branch-and-bound method and maxvol estimates of the maximal element in each subdomain and by using “local” approximations by separable functions.

References

- [1] S. A. GOREINOV, E. E. TYRTYSHNIKOV, and N. L. ZAMARASHKIN, A theory of pseudo-skeleton approximations, *Linear Algebra Appl.*, **261**: 1–21, 1997.
- [2] E. E. TYRTYSHNIKOV, Incomplete cross approximation in the mosaic-skeleton method. *Computing*, **64**(4): 367–380, 2000.
- [3] S. A. GOREINOV and E. E. TYRTYSHNIKOV, The maximal-volume concept in approximation by low-rank matrices, *Contemporary Mathematics*, **208**: 47–51, 2001.
- [4] 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**(3): 939–956, 2008.