

# On the Visualization of Multidimensional Functions using Canonical Decomposition

A.K. Alekseev<sup>1,A,B</sup>, A.E. Bondarev<sup>2,A</sup>, Yu.S. Pyatakova<sup>3,B</sup>

<sup>A</sup> Keldysh Institute of Applied Mathematics RAS

<sup>B</sup> Korolev Rocket and Space Corporation Energia, Korolev, Russia

<sup>1</sup> ORCID: 0000-0001-8317-8688, [aleksey.k.alekseev@gmail.com](mailto:aleksey.k.alekseev@gmail.com)

<sup>2</sup> ORCID: 0000-0003-3681-5212, [bond@keldysh.ru](mailto:bond@keldysh.ru)

<sup>3</sup> ORCID: 0000-0002-8055-7807, [yuliya.pyatakova@rsce.ru](mailto:yuliya.pyatakova@rsce.ru)

## Abstract

The approximation of a multidimensional function by means of tensor decompositions is considered in terms of storage, processing and visualization of the results of parametric calculations in computational aerogas dynamics problems. An algorithm for calculating the canonical decomposition using a combination of the alternative least squares method and stochastic gradient descent is described. Numerical results for interpolation of functions in six-dimensional space obtained using the canonical decomposition are presented, demonstrating the high computational efficiency and quality of results. Visual representations of the results are provided.

**Keywords:** tensor decomposition, canonical decomposition, computational fluid dynamics, visual representation of results.

## Introduction

The treatment and the visualization of the multidimensional data is extremely difficult problem due to the “curse of dimensionality (exponential growth of the required computer memory with the expansion of the problem dimensionality). By this reason we consider the opportunities provided by the tensor form of the multidimensional problems and their approximation by tensor decompositions for the operations with the multidimensional data.

As examples we consider the function  $f(x, y, z, u, v, w)$  defined in the domain  $\Omega \subset R^6$  and corresponding to the probability density in Boltzmann equation and the set of functions  $f_i(x, y, z, \mathcal{G}_1 \dots \mathcal{G}_q)$ , where  $i = 1, \dots, p$  corresponds to the flow variables (density, velocity components, inner energy), and  $(\mathcal{G}_1 \dots \mathcal{G}_q) \subset \Omega_q \subset R^q$  correspond to the parameters of problem (Mach, Reynolds numbers, angle of attack, etc.).

## 1. The tensor form for problems of the aerogas dynamic

Herein we consider the tensor as a multi-way array [1,2] without discussion of the physical sense of tensors. In our case it corresponds to the grid function, defined on the regular (it is important) grid in the multidimensional space. By the gasdynamical variables we always imply their discrete form in this work. Thus, we consider the variables corresponding an unsteady flow-field ( $n$  is the number of time step,  $p$  is the number of the gasdynamical variable) as a tensor

$$\theta_{p;ijk}^n = (\rho_{ijk}^n, u_{ijk}^n, v_{ijk}^n, w_{ijk}^n, e_{ijk}^n). \quad (1)$$

Also, we consider the ensemble of the flow-fields as the tensor

$$\theta_{p;ijk;m_{g_1} \dots m_{g_q}}^n = (\rho_{ijk;m_{g_1} \dots m_{g_q}}^n, u_{ijk;m_{g_1} \dots m_{g_q}}^n, v_{ijk;m_{g_1} \dots m_{g_q}}^n, w_{ijk;m_{g_1} \dots m_{g_q}}^n, e_{ijk;m_{g_1} \dots m_{g_q}}^n), \quad (2)$$

obtained when solving the considered problem in the space of parameters  $(\mathcal{G}_1 \dots \mathcal{G}_q) \subset \Omega_q \subset R^q$ , that corresponds to the statements typical for the generalized computational experiment [3].

Accordingly, the operator of the solution evolution (propagator) is also tensor. In the simplest case (1), the propagator is the tensor of the order 8 that acts on the gasdynamical variables.

$$\theta_{p;ijk}^{n+1} = A_{ps;ijklmz} \theta_{s;lmz}^n. \quad (3)$$

Herein we imply the summation over repeating indexes that is not standard for the operations with tensors, however, it may be convenient for our purposes.

So, the discretizations of both the gasdynamical variables and the corresponding propagators have the form of tensors. Nevertheless, this circumstance is not usually stressed and is not used in applications due to the huge needs for computer memory. The propagators are implicitly applied at the numerical solution of PDE (partial differential equations). They may be easily stated in explicit form for the single time step.

It should be noted that the tensor form of the numerical solutions enables to compress and analyze the results for numerical solutions of the multiparameter problems (defined in the spaces of great (more than three) dimensionality). The compression of the data is performed using the tensor decompositions that are the main topic of present paper.

It is important that the tensor form enables to find nontrivial inner structures both in the solution and in the propagator. The simplest examples concern tensors written in the vectorized and matricized forms. Both vectorization and matricization provide more common and lucid forms of tensor representations and are formally valid. Symmetric matricization (natural for the propagator structure) has the form  $a_{ijklmn} \rightarrow a_{\xi,\eta}, \xi = i + (j-1)I + (k-1)IJ; \eta = l + (m-1)L + (n-1)LM$  for the 6D space. The corresponding vectorization  $a_{\xi\eta} \rightarrow a_{g\xi}, g = \xi + (\eta-1)IJK$ . However, one should remember that the transition from vectors and matrices to tensor (inverse transformation, tensorization) is not always feasible. For example, for the length of vectors equal to the simple number this operation is impossible.

The eigenvectors and the eigenvalues (specific for the matrix algebra) appear in the result of the vectorization and matricization. These objects are not defined in the tensor form. However, they can reflect some inner (hidden) structure of solution and can have the non-trivial physical sense.

For example, in the paper [4] (concerning the atmosphere dynamics) such eigenvectors correspond to the flow disturbances, which maximally grow at the selected time interval (singular vectors). They may be related with the eigenvectors of the operator generated by the product of the forward and adjoint propagators. For this purpose, the gasdynamical variables are vectorized as  $u \in R^N$  and the flow evolution is described by the propagator

$$u(t) = Au_0. \quad (4)$$

The norm of the solution has the appearance

$$\|u(t)\| = (Au_0, Au_0) = (u_0, A^* Au_0). \quad (5)$$

The search for the maximally (in the selected norm) growing linear disturbances  $\|u(t)\| / \|u_0\|$  at time interval  $\Delta t$  is reduced to the search of the eigenvectors of the problem

$$A^* A \eta_{\max} = \sigma_{\max}^2 \eta_{\max} \text{ corresponding to the maximum eigenvalue } \sigma_{\max}^2.$$

The dynamic mode decomposition (DMD) (for the unsteady Euler equations presented by [5,6]) may be provided as another example for implicit vectorization of the solution and the matricization of the propagator. In the DMD frame, the numerical solution of the aero-

gasdynamics problem is vectorized at time step  $i$  and is written as  $u_i \in R^M$  (snapshot). The linear operator  $A(\Delta t) \in R^{M \times M}$  is assumed to exist such that  $u_{i+1} = Au_i$ . Then the snapshots form the Krylov sequence  $Sn_1^{N+1} = \{u_1, Au_1, A^2u_1, \dots, A^Nu_1\}$ . Two sequences  $X = \{u_1 \dots u_N\}$  and  $Y = \{u_2 \dots u_{N+1}\} = AX$ ,  $X, Y \in R^{M \times N}$  are selected from it. Generally (in simplified form) these data enable to construct the approximation of the propagator  $A = YX^+$  ( $X^+$  is the Moore-Penrose pseudoinverse), which is written in compressed form as the product of the rectangular matrices

$$A = \Omega_R^A \Lambda \Omega_L^A. \quad (6)$$

This form enables the radical reduction of the memory necessary for storage of the operator  $A$ . This circumstance enables to use the propagator for the resolution of the set of interesting problems, such as the search for the singular vectors, approximation of the Perron-Frobenius and Koopman operators [5,6].

The applicability of the tensor statements of the computational fluid dynamics problems is severely restricted by the curse of dimensionality (storage of the tensor with the number of indices above three requires nonrealistic memory) despite their straightforward form.

However, at present, the significant progress may be observed at overcoming these difficulties that is connected with the application of tensor decompositions [7,8,9,10].

In the present paper we consider the feasibility of the approximation of the six-dimensional tensors using such tensor decompositions as the canonical decomposition [7,8] and the tensor train [9,10] and illustrate it by the numerical experiments. The choice of the tensor order is related to the three-dimensional Boltzmann equation approximation, is not principal and does not prevent from the use of the considered algorithms for the parametric problems of aerogasdynamics.

## 2. Some definitions of the tensor algebra

For the further exposition we need to use rather high number of rarely used (if not to use term “exotic”) notations [1,11] that we present below for convenience.

**The tensor space** in accordance with [1] is the tensor (outer) product  $\otimes_{j=1}^d R^{I_j} = R^{I_1} \otimes R^{I_2} \dots \otimes R^{I_d}$  of vector spaces  $R^{I_j}$ . Herein, the symbol  $\otimes$  notes the tensor (outer) product of vectors  $a_i \otimes b_j = a_i b_j$  (sometimes, the outer product is noted by symbol  $\circ$ , in order to distinguish it from the Kronecker product).

**Tensor**  $A = [a_{i_1} \dots a_{i_d}]$  ((d-way array)) is the element of the tensor space.

The tensor is described by following parameters:

**Order**  $d$  of the tensor is equal to the number of it’s indices (number of modes, dimensions). There is  $n_i$  nodes over every index (mode)  $i$ . The order of the tensor is equal to the dimensionality of the space at approximation of functions.

**Size** of the tensor is equal to the product of the number of nodes over all dimensions  $n_1 \times \dots \times n_d$ , corresponds to the number of memory required for the tensor storage, and grows exponentially ( $\sim n^d$ ) in dependence on the tensor order.

**Rank** of tensor  $rank(A) = R$  is defined as the minimum number of the layers of cores  $a_r^{(i)}$  (at fixed  $r$   $a_r^{(i)} \in R^N$  are the normed vectors), which is necessary for the tensor approximation in the following form (canonical decomposition):

$$A = \sum_1^R \lambda_r a_r^{(1)} \otimes \dots \otimes a_r^{(N)}. \quad (7)$$

**Fiber of the tensor** is the vector that is obtained at varying of one index at others fixed .

The following fibers exist for three-dimensional tensor:  $x_{:jk}$  (mode-1 fiber, column),  $x_{i:k}$  (mode-2 fiber, row), and  $x_{ij:}$  (mode-3 fiber, “tunnel”).

The following tensor operations we use or discuss.

**$n$  –mode product of the tensor and the vector** is noted using symbol  $\times_n$  as  $Y = X \times_n v$ . Every mode- $n$  fiber is scalarly multiplied by the vector as

$$(X \times_n v)_{i_1 \dots i_{n-1} i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \dots i_N} v_{i_n}. \text{ The tensor of the less order is obtained in result.}$$

**Product of two tensors** is noted by the symbol  $\times_q^p$  and has the form  $Z = A \times_q^p B = \sum A_{\dots n_{q-1} k n_{q+1} \dots} \times_q^p B_{\dots m_{p-1} k m_{p+1} \dots}$ .

**Kronecker product** ( $\otimes$ ) of the arbitrary size matrices is the generalization of the outer product from vectors to matrices. The element of the first matrix is multiplied by the second matrix in the form that follows:

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \dots & \dots & \dots & \dots \\ a_{i1}B & a_{i2}B & \dots & a_{iJ}B \end{pmatrix}.$$

The Kronecker product of matrices  $A \in R^{I \times J}$ ,  $B \in R^{K \times L}$ ,  $M = IK$ ;  $N = JL$   $C = A \otimes B \in R^{IK \times JL}$  may be written in the index form as follows:

$$c_{mn} = a_{ij} b_{kl}, \quad m = (i-1)K + k, \quad n = (j-1)L + l, \quad m = 1 \dots M; n = 1 \dots N. \quad (8)$$

Unfortunately, as can be seen from (8), the index notation of such operations does not provide the lucidity that is common in the physical applications [12].

**Khatri-Rao product** is usually noted by  $\odot$ , herein, it is more convenient to use the symbol  $\bullet$ . It is used for matrices with the coinciding number of the columns. Every element of the first matrix column is multiplied by the total column of the second one. The result is formed as the column. For  $A = [A_1 A_2 \dots A_R]$ ,  $B = [B_1 B_2 \dots B_R]$

$$A \odot B = [A_1 \otimes B_1 \dots A_R \otimes B_R]. \quad (9)$$

**Hadamard product**

$$C = A * B \quad (c_{ij} = a_{ij} b_{ij}). \quad (10)$$

(no summation over repeating indices) is usually noted as  $*$  and corresponds the element-wise multiplication of same dimension matrices.

Very often it is convenient to roll out the tensor into pancake or stretch into a fiber. The corresponding matricization or vectorization (unfold) are performed by the following operations.

**Mode- $n$  matricization** of the tensor  $X \in R^{I_1 \times \dots \times I_N}$  is denoted by  $X_{(n)}$  and sets the mode- $n$  fibers in the matrix column. In general case, the matricization of the tensor has the

form of the transformation  $X_{i_1, \dots, i_N} \rightarrow m_{i_n, j}$ ,  $j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^N ((i_k - 1) \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m)$ . For

$X_{i,j,k} \in R^{I \times J \times K}$ ,  $i=1 \dots I, j=1 \dots J, k=1 \dots K$ , the matricization, for example, may have the appearance  $X_{(1)} = X_{i,m}$ ,  $m=1 \dots M$ ,  $M = J \cdot K$ ,  $m = j + (k-1)J$ ; ( $j=1 \dots J, k=1 \dots K$ ).

**Vectorization** unfolds the matrix  $X_{i,m}$  into the vector  $Y_j$ ,  $j = I_2(i-1) + m$ .

The matricization and the vectorization of tensors are very popular since they enable to use all spectrum of the linear algebra algorithms. However, their practical application at the tensor order above three is restricted by the curse of dimensionality.

We are interested in the tensor decompositions by the tensors of the less order or size, such as

**Tucker decomposition**  $A = B \times_1 G_1 \dots \times_N G_N$ , in the index form

$$a(i_1, \dots, i_N) = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_N=1}^{r_N} b(i_1, \dots, i_d) g_1(i_1, \alpha_1) \dots g_N(i_N, \alpha_N)$$

**Canonical decomposition**  $A = I_N \times_1 G_1 \dots \times_N G_N$ , in the index form

$$A_{ij \dots N} = \sum_{r=1}^R g_{i,r} g_{j,r} \dots g_{N,r}$$

**Tensor train**  $A = G_1 \times_2^1 G_2 \times_3^1 \dots \times_{N-1}^1 G_{N-1} \times_N^1 G_N$ , in the index form

$$A(i_1, \dots, i_N) = \sum_{\alpha_0, \dots, \alpha_N} g_1(\alpha_0, i_1, \alpha_1) g_2(\alpha_1, i_2, \alpha_2) \dots g_N(\alpha_{d-1}, i_d, \alpha_N)$$

The above notations, as a rule, are cumbersome, not transparent from the intuitive viewpoint, are used in very narrow domain and are unknown for the most specialists. Unfortunately, it is impossible to describe the current state of affairs in the tensor decomposition without these notations. However, we shall try to use more common index notations where it is possible or, in some events, duplicating both approaches.

### 3. Canonical decomposition

The above mentioned canonical decomposition  $A = \sum_1^R Q_r^1 \otimes Q_r^2 \otimes \dots \otimes Q_r^N$  is written in the index form (for non-normed cores) as

$$A_{ij \dots k} = \sum_1^R Q_{i,r} Q_{j,r} \dots Q_{k,r} \quad (11)$$

This expression is unique if not account for the permutation or scaling. The problem for the determination of the set of cores  $Q^n = \{Q_r^1, \dots, Q_r^N\}$  has the appearance (in the variational form)

$$Q^n = \arg \min_{Q^n} \left\| A - \sum_1^R Q_r^1 \otimes Q_r^2 \otimes \dots \otimes Q_r^N \right\|. \quad (12)$$

The rank of the tensor  $R$  is the key parameter at the application of the canonical decomposition. In accordance with [1,13] it is not computable due to the ill-posedness of the problem

$$R = \arg \min_R \left\| A - \sum_1^R Q_r^1 \otimes Q_r^2 \otimes \dots \otimes Q_r^N \right\|. \quad (13)$$

The canonical decomposition suffers from instabilities and requires a regularization [1,13]. However, in accordance with [14] the approximation of the positive functions (such as the probability density) by the canonical decomposition engenders the well-posed stable

statement. The oscillating behaviour of the cores was observed in the present work, also. However, it is not principal from the standpoint of the multidimensional functions' approximation.

The canonical decomposition implies the compression  $N^n \rightarrow N \cdot n \cdot R$ , where  $N$  is the space dimension,  $n$  is the number of nodes over single direction,  $R$  is the tensor rank.

The canonical decomposition is equivalent to DMD [5,6]  $A = \Omega_R^A \Lambda \Omega_L^A$  in the two-dimensional case.

The canonical decomposition is some extension of the Principal Components Analysis (PCA) at the tensor order expansion over two (transition from matrices to the multi-way arrays) and also enables to reduce the dimensionality of the problem.

## 4. The Tensor train

The canonical decomposition rather often suffers from instabilities [13]. By this reason, the attempts to find alternative decompositions are natural. The tensor train (TT) format [8] is one of such attempts. The works exist ([9]) that state that the tensor train is more stable if compare with the canonical decomposition.

**The Tensor train (TT)** enables to write the  $d$ -way  $n_1 \times n_2 \times \dots \times n_d$  tensor  $A$  in the form

$$A(i_1, \dots, i_d) = \sum_{\alpha_0, \dots, \alpha_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \dots G_d(\alpha_{d-1}, i_d, \alpha_d), \quad (14)$$

where  $G_k$  are the cores of the size  $r_{k-1} \times n_k \times r_k$ ,  $k = 1, \dots, d$ ,  $r_0 = 1$ ,  $r_d = 1$ .

The tensor train is not unique transformation since it is invariant regarding the transformation  $G'_k(i_k) = G_k(i_k)S$ ,  $G'_{k+1}(i_{k+1}) = S^{-1}G_{k+1}(i_{k+1})$ .

The tensor train provides less compression  $nNr^2$ , if compared with the canonical decomposition.

The tensor train is an interesting alternative to the canonical decomposition. The comparison of the tensor train and the canonical decomposition is interesting both from the viewpoint of the stability of results and the computational efficiency. We hope to perform the corresponding comparison in future works.

## 5. The methods for the tensor decomposition calculation

The methods for the tensor decomposition calculation may be divided into two subclasses: the methods which are based on the linear algebra, for example [9] and the variational methods [6,7].

The linear algebra based methods significantly apply tensor matricization, singular decomposition and contain a lot of interesting and original algorithms that enable to execute operations on cores without appealing to approximated functions.

The variational statements, as a rule, are based on alternating least squares (ALS) [15,16], however, the matricization of tensors is also used.

The utilization of the tensor matricization is, by our opinion, the weak point of both approaches, since it requires a huge memory.

However, we believe that the variational methods may be relieved from this drawback. By this reason, herein we use certain combination of alternating least squares and the stochastic gradient descent (SGD) [17,18,19], which will be described below. Roughly speaking, we minimize the discrepancy on the single randomly selected fiber using ALS, that enables us to resolve one-dimensional problem with the moderate requirements to memory at every step.

## 5.1 The method of alternating least squares

The alternating least squares (ALS) method [15,16] is commonly used at the search for the tensor decompositions and enables optimization of the single parameter while others are fixed. For the canonical decomposition ALS is realized by Khatri-Rao product, usually, for three-dimensional problems. For the case of our interest (multidimensional) [19,20] cores  $Q_k$  are determined by the consequent solution of the following problem

$$Q_k = \arg \min_{Q_k} \left\| A_{(k)} - Q_k (Q_1 \bullet \dots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \dots \bullet Q_N)^T \right\|^2. \quad (15)$$

Herein  $A_{(k)}$  is the mode- $k$  matricization of the tensor,  $Q_k (Q_1 \bullet \dots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \dots \bullet Q_d)^T$  is the auxiliary matrix of the same dimensionality. One may obtain the elegant expression for the minimum of (15)

$$Q_k = A_{(k)} ((Q_1 \bullet \dots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \dots \bullet Q_d)^T)^+, \quad (16)$$

which enables the estimation of the core by the matrix algebra methods.

In the general case (at a variation of all parameters), the convexity is not guaranteed and the gradient descent is not obliged to converge. The fixation of the main part of parameters and the variation of the single core enable to obtain the convex goal functional and to optimize it with success. The low rate of the ALS convergence is the cost of the relative universality of the method.

The approach to the canonical decomposition calculation using expressions of Eq. (16) kind dominates at present. Unfortunately, this approach is not applicable for our purposes, since it uses the tensor matricization, which requires the same memory as the tensor itself. The memory, which is necessary for the problems of considered class, is above the range of modern computers parameters (in this paper we use tensors, formally containing  $10^{12}$  numbers), that excludes the application of the tensor matricization.

## 5.2 Stochastic gradient descent

The stochastic gradient descent (SGD) is widely used in the problems of high dimensionality [17,18,19]. Paradoxically, it enables to find the point of the functional minimum in the space of the control parameters for less time than it takes to fully calculate this functional to. This occurs since the local functional (on single random point or small set of points (minibatch)) is computed instead the global (batch) functional (12) that requires huge time for computation. Rather often SGD is used in a combination with ALS in order to overcome difficulties caused by huge memory requirements at the Khatri-Rao product application [17,18].

In our case we use the functional computed on the single randomly selected fiber  $i$  ( $n_i = 1 \dots N_i$  at other fixed indices) or a small set of such fibers. Corresponding algorithm is described in the Section that follows in details.

## 6. The numerical algorithm for the canonical decomposition

As we stated above, our approach corresponds to some combination of the stochastic gradient descent (in minibatch variant) and alternating least squares method. We present it in details for the single fiber case (the case for the set of fibers may be obtained by simple summation of the discrepancy and the gradient) since we failed to find the description of this algorithm in publications. We consider the six-dimensional case and the following approximation

$$f_{ijklmp} = \sum_{\alpha=1}^R Q^x(\alpha, i) \cdot Q^y(\alpha, j) \cdot Q^z(\alpha, k) \cdot Q^u(\alpha, l) \cdot Q^v(\alpha, m) \cdot Q^w(\alpha, p). \quad (17)$$

At the beginning, we determine the first core  $Q^x(\beta, i)$  related with the coordinate  $x$ . Other cores are determined consequently and the corresponding expressions may be found by the cyclic change.

We select a fibre along  $x$  ( $i = 1 \dots N_x$ ) by the random uniformly distributed choice of other indices  $j, k, l, m, p$ .

Let's consider the discrepancy along this fibre obtained by summation over  $i$  of the local (point-wise) discrepancies

$$\varepsilon(Q^x) = \sum_i^{N_x} \left\{ \sum_{\alpha} Q^x(\alpha, i) \cdot Q^y(\alpha, j) \cdot Q^z(\alpha, k) \cdot Q^u(\alpha, l) \cdot Q^v(\alpha, m) \cdot Q^w(\alpha, p) - \tilde{f}_{ijklmp} \right\}^2 / 2. \quad (18)$$

Here  $\tilde{f}_{ijklmp}$  is the exact magnitude of the function at the point  $i, j, k, l, m, p$  that is known beforehand (in the present work from the analytic expressions for the test functions). In accordance with the ALS approach, the discrepancy is considered to depend on the single core  $Q^x(\alpha, i)$ . Let's disturb this core by  $\Delta Q^x(\beta, i)$ . The corresponding disturbance of the discrepancy has the form

$$\Delta_x \varepsilon = \sum_i \left\{ \left( \sum_{\alpha} (Q^x(\alpha, i) \cdot Q^y(\alpha, j) \cdot Q^z(\alpha, k) \cdot Q^u(\alpha, l) \cdot Q^v(\alpha, m) \cdot Q^w(\alpha, p) - \tilde{f}_{ijklmp}) \cdot \left( \sum_{\beta} \Delta Q^x(\beta, i) \cdot Q^y(\beta, j) \cdot Q^z(\beta, k) \cdot Q^u(\beta, l) \cdot Q^v(\beta, m) \cdot Q^w(\beta, p) \right) \right) \right\}. \quad (19)$$

Let's choose  $\Delta Q^x(\beta, i)$ , which is not equal to zero only at single point  $i$ , that free us from summation over  $i$  in (19). Then, one may extract the corresponding value of the gradient in form:

$$\nabla_{x,i,\beta} \varepsilon = \left( \sum_{\alpha} (Q^x(\alpha, i) \cdot Q^y(\alpha, j) \cdot Q^z(\alpha, k) \cdot Q^u(\alpha, l) \cdot Q^v(\alpha, m) \cdot Q^w(\alpha, p) - \tilde{f}_{ijklmp}) \cdot (Q^y(\beta, j) \cdot Q^z(\beta, k) \cdot Q^u(\beta, l) \cdot Q^v(\beta, m) \cdot Q^w(\beta, p)) \right). \quad (20)$$

The defect of expression (20) if compared with ALS methods, using the Khatri-Rao product (16), is the impossibility of the direct use of condition  $\nabla_{x,i,\beta} \varepsilon = 0$  (since the sought value  $Q^x(\alpha, i)$  is summated over  $\alpha$ ) for calculation of cores. The merit of expression (20) if compared with (16) is the economy of the memory (matrization is not used).

The regularized term  $-\gamma Q^x(\beta, i)$  should be added to (20) if the zero order Tikhonov regularization ( $\gamma(Q^x(\beta, i))^2$ ) is used.

The steepest descent iterations that minimize the functional (18) over the core  $Q^x$  element at point  $\beta, i$  have the form

$$\{Q^x(\beta, i)\}^{n+1} = \{Q^x(\beta, i)\}^n - \tau \nabla_{x,i,\beta} \varepsilon, \quad (21)$$

where  $\tau$  is the iteration step.

Iterations on the single core and selected fibre are performed until relaxation. The discrepancy over selected fibre (18) was used as the stopping criterion. Iterations terminated at  $\varepsilon \leq \varepsilon_1 = 10^{-9} \div 10^{-13}$ .

We proceed to the next core using a new randomly selected fibre past stopping iterations on the current core. The form of the gradient is obtained by permutations in the second term of (19) while the sum in (18) depends on the chosen fibre.



We proceed to the next step of the global iteration (from the new values of cores) and again start the search for the optimal cores from  $Q^x$  past all cores are locally optimized. Formally, the quality of the function approximation by the canonical decomposition at every step of the global iteration may be estimated via the discrepancy that follows

$$\tilde{\varepsilon}_{total} = \sum_{i,j,k,l,m,p} \left\{ \sum_{\alpha} Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp} \right\}^2 / 2. \quad (22)$$

Unfortunately, this functional is not directly computable (at least on usual personal computers) due to the problems with dimensionality and the great demands for computer time. We numerically estimated its value using the Monte-Carlo method used in the form

$$\varepsilon_{total} = \frac{1}{2MC} \sum_{s=1}^{s=MC} \left\{ \sum_{\alpha} Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp} \right\}^2, \quad (23)$$

where at every step of summation  $s$  every index from  $i, j, k, l, m, p$  was chosen as the random uniformly distributed number. In result, we computed the averaged over the ensemble sum of the approximation error squares. The number of trials in the ensemble was in the range  $MC = 1000 \div 100000$ . The results varied rather weakly at the change of  $MC$ .

The optimization of all cores (and the total process of the canonical decomposition generation) stopped at  $\varepsilon_{total} \leq \varepsilon_2$ ,  $\varepsilon_2 = 10^{-5} \div 10^{-7}$ .

In general, the combinations of ALS and SGD are rather widespread [17,18], however, in all known to authors events they use the Khatri-Rao product. The approach, presented here, does not use the Khatri-Rao product in any form, but is based on a direct numerical differentiation of the discrepancy, which is defined on the single fibre, and the gradient descent. By this reason, neither tensor matricization nor cores product in the Khatri-Rao form are not used, that enables the radical reduction of the memory requirements. The increase of the number of iterations necessary for the problem solution is the cost of this success. Fortunately, it does not lead to the sensible consequences for the considered problems, since the computation time for the considered problems on the personal computer (Intel I5, 2.66 GHz) remains in the limits of the several minutes.

## 7. The results of numerical tests

In the results of calculations we obtain the approximation of the six-dimensional function  $\tilde{f}$  (more correctly, the tensor  $\tilde{f}_{ijklmp}$ , corresponding the values of the function in the nodes of the regular grid) using the canonical decomposition and the corresponding set of cores  $Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p)$ .

The grid containing 100 nodes on every coordinate was used in numerical experiments. Formally, the storage of  $\tilde{f}_{ijklmp}$  on such grid requires  $10^{12}$  cells of memory that is not realistic neither from the viewpoint of storage nor from the viewpoint of the visualization. We mark that the memory necessary for cores with the rank 100 requires  $6 \cdot 100 \cdot 100 = 60000$  cells, which illustrates super high compression of the information ( $\sim 10^7$ ) at application of the canonical decomposition.

The comparison of the numerical (obtained by the direct numerical differentiation) and analytical gradients (obtained by expression (20)) was performed during debugging and demonstrate their practically complete coincidence.

Formally, the quality of the approximation of the function by the canonical decomposition may be estimated using discrepancy (22), but it was estimated using Monte-Carlo method (23).

The results of computations provide sufficiently stable and reproducible error estimations.

The numerical tests were performed using the authors' codes written in Fortran-95 specifically for the considered problems.

## 7.1 The test problems

The tests of the approximation of different functions by the canonical decomposition is performed. The quality of the approximation (17) is of interest at testing both from the viewpoint of visual presentation and from the value of the discrepancy (23). The determination of the real rank of the function and the convergence rate are of interest. The corresponding data are presented in this Section. The six-dimensional functions of the different rank of the tensor presentation are selected. This enables to estimate not only the quality of the approximation but also the possibilities for the estimation of the rank for the functions under consideration. The following multidimensional functions are considered that are situated in the order of the complexity (canonical decomposition rank) increasing.

### 1. The product of vectors

$$f = x \cdot y \quad (25)$$

It is the simplest function with the rank of the tensor equal unit. Single fibre, approximating rank of cores in the range from 1 to 10, 30 iterations are used. The dependence of discrepancy on the rank is provided in Table 1.

Table 1. The dependence of discrepancy (23) on the rank in (17) for the function (25)

rank	1	2	3	4	5	10
discrepancy (23)	$3.27 \cdot 10^{-14}$	$2.78 \cdot 10^{-6}$	$4.61 \cdot 10^{-5}$	$3.34 \cdot 10^{-6}$	$2.44 \cdot 10^{-4}$	$1.63 \cdot 10^{-4}$

These results show that the magnitude of the discrepancy may serve as the indicator of the true rank of the tensor. The noise in the results increases as the rank rises, obviously, this reflects the instabilities occurring at the rank estimation arising due to the ill-posedness of this problem for the canonical decomposition [1,13]. The results of the computations ( $f(x, y)$ ) are presented in Fig. 1 (exact function) and Fig. 2 (approximation, rank 5).

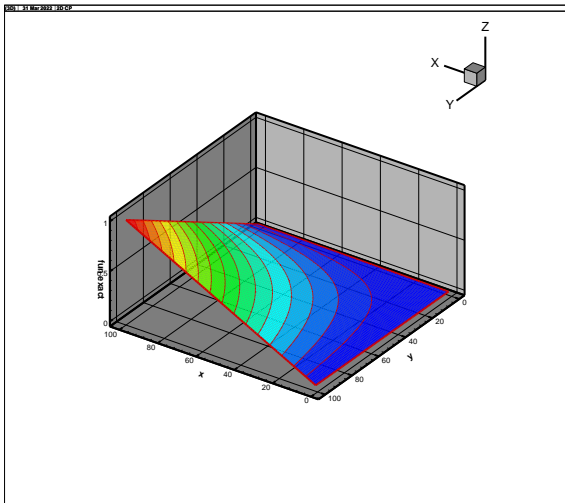


Fig. 1 Exact function (25)

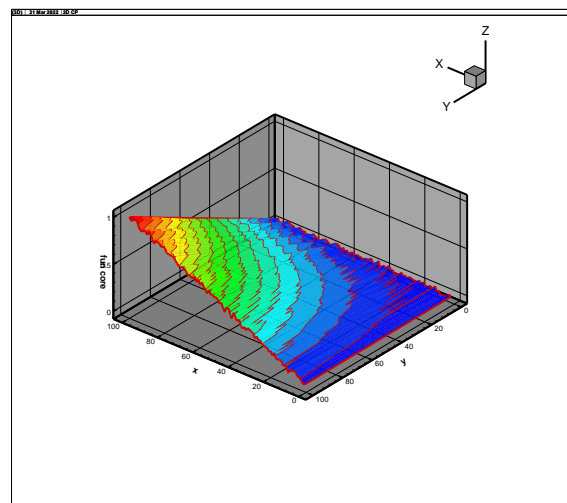


Fig. 2. Approximation of function (25), rank 5

### 2. The sum of vectors

$$f = x + y \quad (26)$$

It is also very simple function, but it's rank a priori is unknown and it would be desirable to estimate it in computations. Single fibre, rank in the range from 1 to 10, 30 iterations are used. The dependence of the discrepancy on the rank is presented in Table 2.

Table 2. The dependence of discrepancy (23) on the rank in (17) for the function (26)

rank	1	2	3	4	5	10
discrepancy (23)	$4.87 \cdot 10^{-2}$	$1.12 \cdot 10^{-2}$	$8.67 \cdot 10^{-5}$	$8.02 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$	$2.03 \cdot 10^{-4}$

This function has the rank  $3 \div 4$  if the minimum of the discrepancy is analyzed. Fig. 3 presents the exact function, Fig. 4 presents its approximation (rank 5), one may mark the sufficient coincidence.

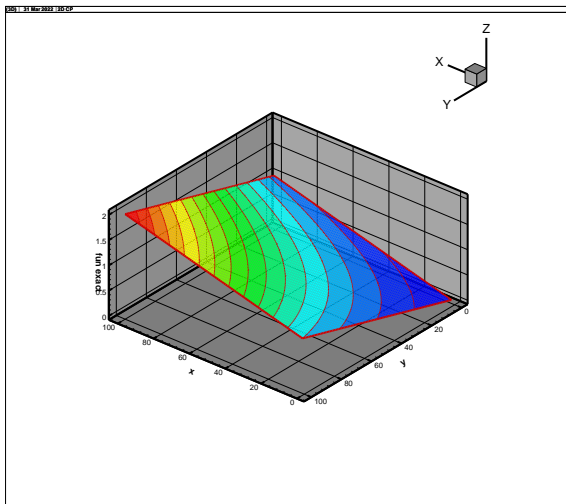


Fig. 3. The exact function (26)

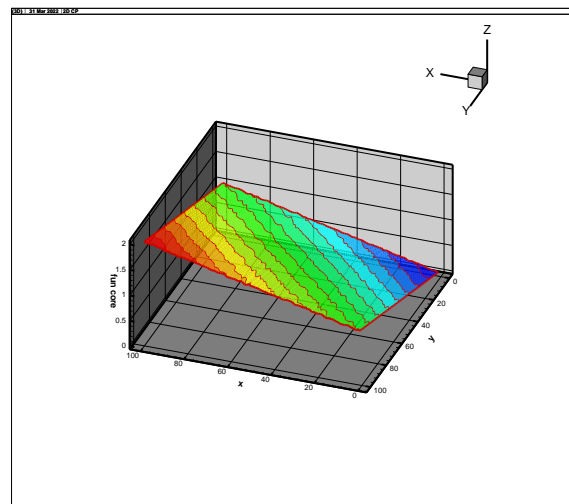


Fig. 4. The approximation of function (26)

### 3. The sum of sines

$$f = \sin(x / 20) + \sin(y / 20) \quad (27)$$

It is the two-dimensional function in the six-dimensional space that is visually significantly more complex if compare with (25) and (26). The calculations demonstrates the rank of this function to be about 10. The results of computations are presented in Fig. 5 (exact function) and Fig. 6 (approximation, rank 10, 1 fibre,  $\varepsilon = 5 \cdot 10^{-4}$ , 13 iterations).

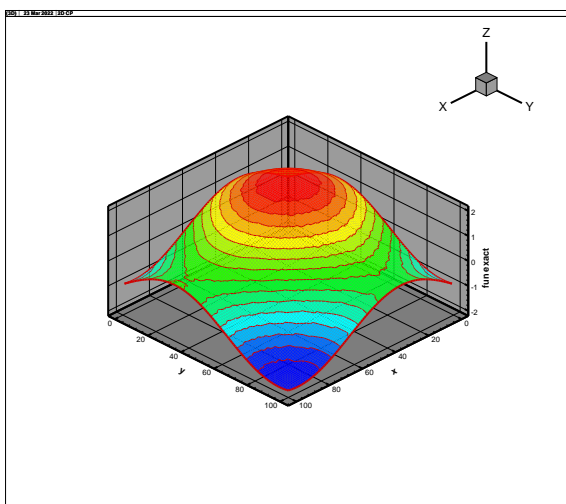


Fig. 5. Exact function (27)

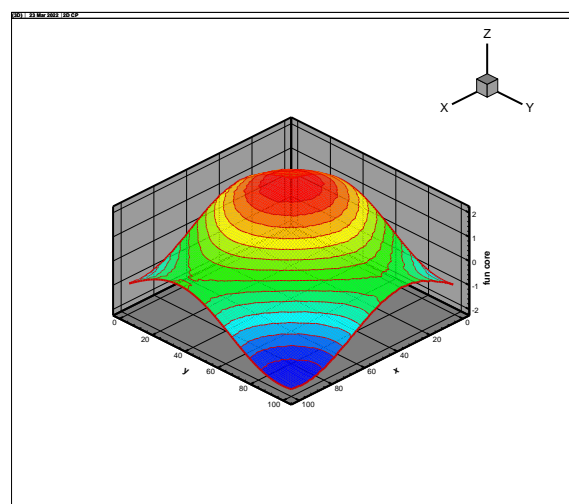


Fig. 6. The approximation of function (27)

### 4. Two-dimensional product of sines

$$f = \sin(x / 20) \cdot \sin(y / 20) \quad (28)$$

It is also two-dimensional functions in the six-dimensional space, it is the multiplicative analogue of (27). The rank of this function is about 10. Results of computations are provided in Fig. 7 (exact function) and Fig. 8 (approximation, rank 10, 1 fibre,  $\varepsilon = 7 \cdot 10^{-5}$ , 30 iterations).

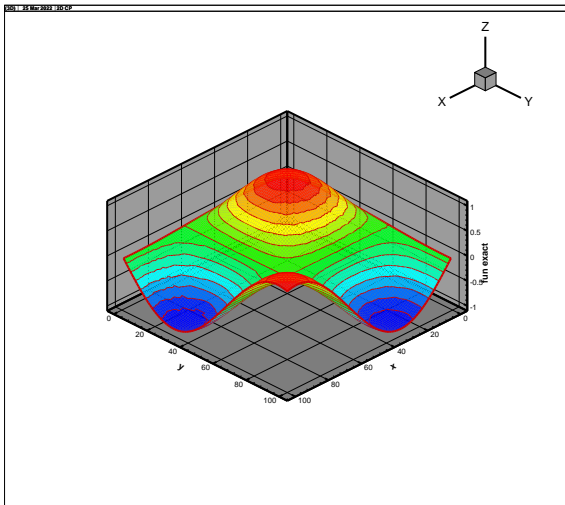


Fig. 7. The exact function (28)

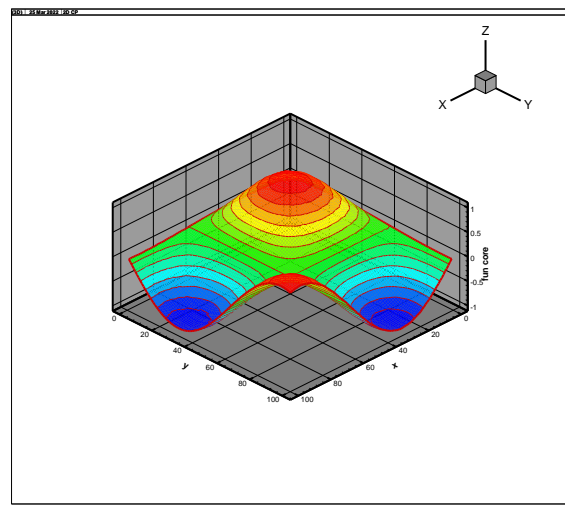


Fig. 8. The approximation of function (28)

The previous tests were performed in the six-dimensional space over the two-dimensional functions and demonstrated enough fast convergence (10-30 iterations) and low values of the discrepancy. Further we consider truly multidimensional problems that require greater number of iterations and demonstrate greater discrepancies.

#### 5. The Gaussian in the multidimensional space

Let's consider the function in the multidimensional space that is described by the following equation

$$rad = 0.001 \cdot ((ix - 50) + (iy - 50) + (iz - 50) + (iu - 50) + (iv - 50) + (iw - 50))$$

$$f = \exp(-rad^2) \tag{29}$$

Formally, this function is defined in the six-dimensional space, but, really, it is one-dimensional (depends only on the radius) and is determined by the product of vectors, so it's rank equals unit. Table 3 presents the dependence of discrepancy on the rank for the function (29)

Table 3. The dependence of discrepancy (23) on the rank in (17) for the function (29)

rank	1	2	3	4	5	10
discrepancy (23)	$5.31 \cdot 10^{-13}$	$6.92 \cdot 10^{-7}$	$2.78 \cdot 10^{-6}$	$6.74 \cdot 10^{-4}$	$3.91 \cdot 10^{-5}$	$5.13 \cdot 10^{-3}$

30 iterations and 1 fibre are used. The complete coincidence of function and its approximation is observed for the rank 1. The noise in results grows and the accuracy decreases (discrepancy grows) as the rank increases.

Thus, the estimation of the exact rank of function is necessary and the calculations for the greater rank ("with the reserve") may not provide the necessary quality.

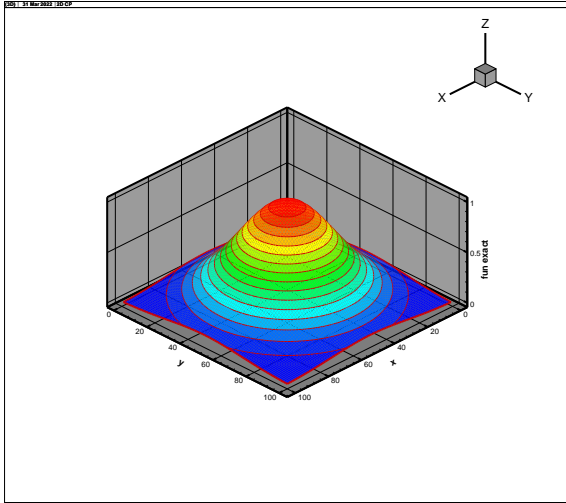


Fig. 9. The exact function (29)

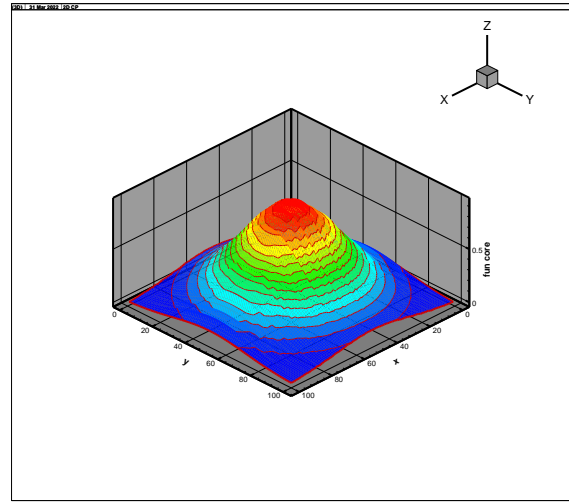


Fig. 10. The approximation of function (29), rank 5.

### 6. The sum of sines

$$f = \sin(x / 20) + \sin(y / 20) + \sin(z / 20) + \sin(u / 20) + \sin(v / 20) + \sin(w / 20) \quad (30)$$

It is the most difficult for calculations variant that is truly six-dimensional. The rank of this function is about 200 as will be demonstrated in the next Section. The calculations for this variant converge rather slow (about 300 iterations) and require a great enough rank. Figs. 11 and 12 demonstrate results for 10 fibres and rank 200 in the plane  $x, y$ , the discrepancy  $\varepsilon = 3 \cdot 10^{-4}$ . Other variables correspond to the centres of the intervals on the grid 100 ( $iz = iu = iv = iw = 50$ ).

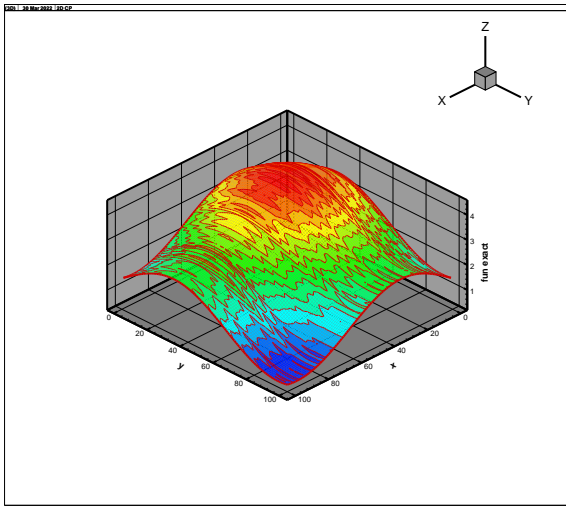


Fig. 11. Exact function (30)

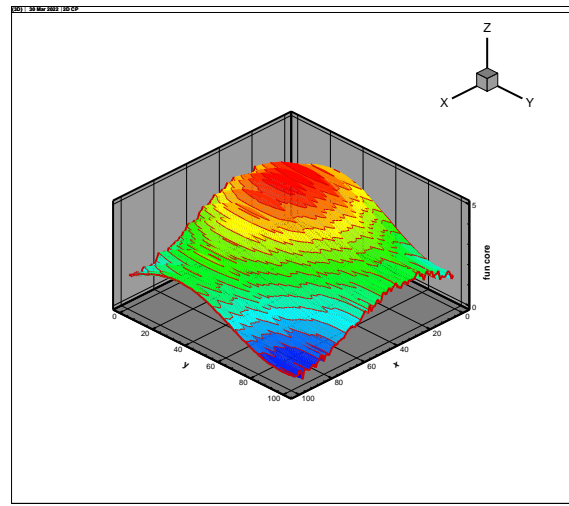


Fig. 12. The approximation of function (30), rank 200

## 7.2. Iteration convergence criteria

Fig. 13 presents the behaviour of different convergence criteria in the dependence on the number of iterations for the function (27). The discrepancy over the fibre (eps\_fibre), global discrepancy estimated by Monte-Carlo method (eps\_MC), and the norm of the gradient of discrepancy (grad norm) are provided. The increasing of the discrepancy at transition to the next global step (rank values refreshing) was permitted in the variant of the optimization, illustrated by Fig. 13

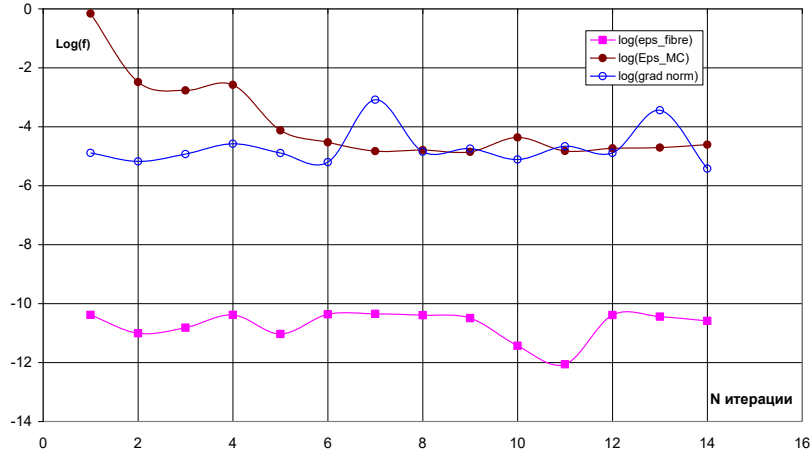


Fig. 13. Different criteria of convergence in dependence on the number of iterations.

The present variant of the optimization does not provide the monotonic convergence, although, the minimization is observed for the simple enough functions. The local convergence (summation on the fiber) occurs at every step of the global iteration. The convergence of the gradient norm is not observed. Slow and nonmonotonic convergence of the global discrepancy (the difference between the exact and approximate solutions in  $L_2$  norm, estimated by Monte-Carlo method (23)) is observed.

The transition to the variant of the minimization, which prohibits the increase of discrepancy at the transition to the next (random) set of fibres (at the next step of the global iteration) was used for more complex functions. Actually, the gradient optimization at certain moment is replaced by the stochastic search over fibres in this version of the algorithm.

One may conclude from the intuitive viewpoint that the utilization of several fibres instead single one (minibatch) should improve the monotonicity of the convergence. However, the numerical experiments demonstrated that, starting from certain moment, it spoils the convergence rate and the achievable value of discrepancy. Fig. 14 presents the results of the convergence for 1 and 5 fibres for function (27).

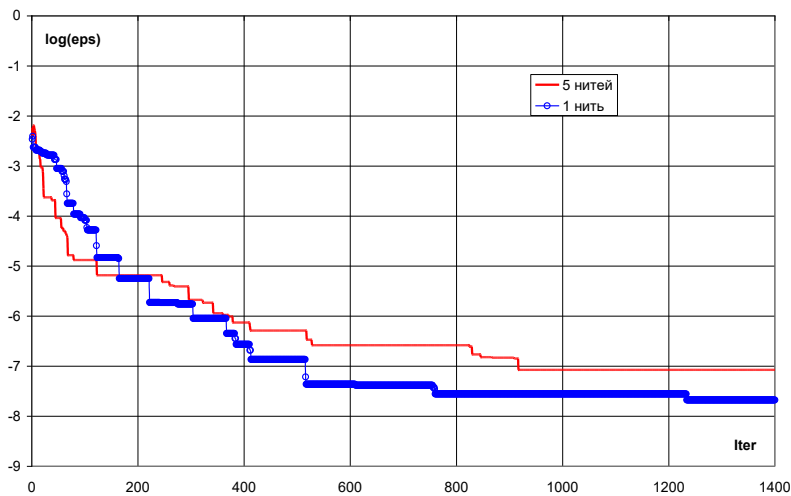


Fig. 14. The logarithm of the discrepancy (23) in dependence on number of iterations for 1 and 5 fibres.

### 7.3 Rank of the decomposition

The numerical estimation of the tensor rank was obtained by viewing the results over the rank magnitude at the solution of the variational problem (23). It was assumed, that the discrepancy should decrease as the rank expands as it is usually observed in calculations. However, for some simple functions the opposite behaviour is demonstrated. For example, the minimum of the discrepancy occurs at unit rank for the multidimensional Gaussian (29). The expansion of the rank increases the discrepancy.

The dependence of discrepancy logarithm on the rank is presented in Fig. 15 (1 fibre, 300 iterations) for the function described by the Eq. (30).

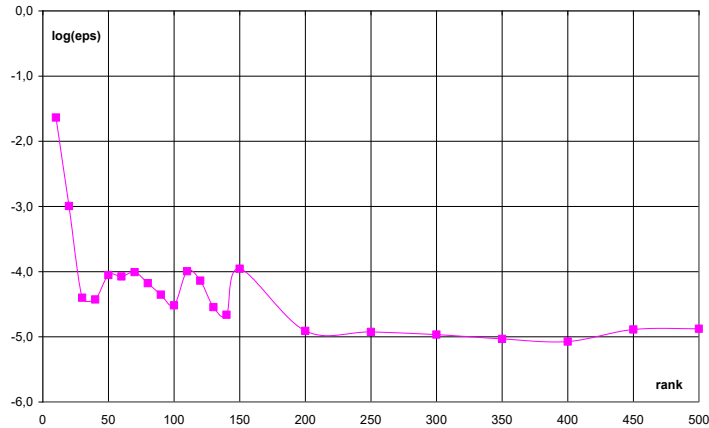


Fig. 15. The dependence of the discrepancy on the rank for function (30)

The dependence of the discrepancy on the rank value is enough nonmonotonic for this function. Nevertheless, the rank expansion decreases the discrepancy. Thus, the rank should be adapted for discrepancy diminishing at calculations. The simplest variant (the run over expanding number of the ranks) is used herein. However, the solution of the problem in the consequently expanding space is rather costly. So, the search for more suitable algorithm for the rank estimation is necessary despite the available principal difficulties [13].

Fig. 16 presents the core  $Q^x(\alpha, i)$  for function (30), rank is equal 100. Fig. 17 presents core  $Q^x(\alpha, i)$  for function (30) for rank 250. The oscillating character of the core is observed. No damping at the expansion of the rank is observed, although, the accuracy of the approximation increases.

The difficulties with the determination of the rank in canonical decomposition stimulate the investigation of other tensor decompositions, the tensor train in particular.

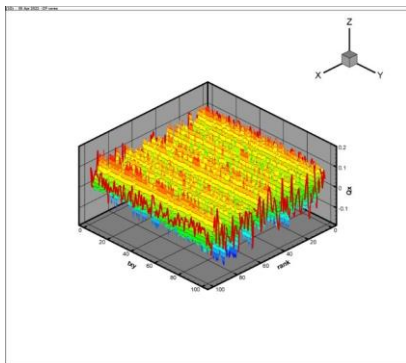


Fig. 16  $Q^x(\alpha, i)$ , rank 100

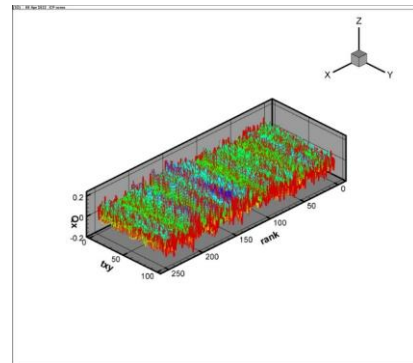


Fig. 17  $Q^x(\alpha, i)$ , rank 250

## Discussion

If the initial approximation of the cores is chosen as constants, for example  $Q^m(\alpha, i) = 1$ , all magnitudes of gradients for different  $\alpha$  automatically coincides (19) and the optimization process fails. So, the expression  $Q^m(\alpha, i) = 1 + N(\sigma)$  was used for the initial approximation of cores (the random normally distributed value with dispersion  $\sigma$  was used). Naturally, there is no convergence at  $\sigma = 0$ . The instabilities were observed at the great  $\sigma \approx 10$ . The optimal magnitude is  $\sigma = 0.1$ , which was used in all above described tests.

The obtained results depend on the random selection of the fibres and on the noise in the initial approximation of cores that partly make difficult comparison at debugging and the search for the optimal parameters for the algorithm installation. The fixation of the start point of the random number generator and storing the sequences of fibre coordinates (used at optimization) enables the stabilization of results.

Due to the problem ill-posedness [13] the zero order Tikhonov [21] quadratic regularization is provided. However, the dumping of the oscillations at calculation of cores caused the break of the approximation in numerical tests. So, the positive influence of the regularization was not observed and the above provided numerical tests correspond to the absence of the regularization. The ill-posedness of the canonical decomposition does not have an influence in the frame of approximation of functions. In the frame of the solution of the evolutional problems in partial differential equations (PDE) [10] the stability of the tensor decomposition is useful since it enables to perform some part of evolution in the space of cores without returning to the space of functions. The possibility of the evolution in the space of cores is provided in the frame of tensor train [9]. There exists the set of operations including special TT-rounding operation, which enables to reduce the rank of the approximation past several steps. It explains the significant interest to the tensor train format. So, the instability at the level of the determination of cores is not insurmountable obstacle at the simulation of PDE at spaces of the high dimensionality. Nevertheless, the transition to the more stable statements (tensor train) may enable to obtain more fast algorithms.

There exists the coincidence of numerical and analytical gradients, but the analytical calculation is faster about two order of the magnitude at the considered problem. It is related with the circumstance, that disturbance of the discrepancy (18) should be computed for any element of the core at the numerical calculation of the gradient. The need for this operation (and in summation over the coordinate) is absent at the analytical computation (20).

The plausible idea that the transition from the single fibre to several ones provides more stable optimization failed in calculations. This transition slows down the convergence rate and deteriorates the quality of optimization (increases the value of the reachable discrepancy).

The numerical tests show that the tensor rank strongly depends on the kind of the approximated function.

As far the rank increases the noise in the result rises, so the choice of the rank value “from above”, which partly simplifies the algorithm, may cause the deterioration of the results.

The tensor decompositions are actively used for the visualization purposes since they enable to build the easily computable model, which approximate the difficult data set in the space of parameters. For example, papers [22,23] use tensor train format and the cross approximation for these purposes.

The tensor decompositions (canonical decomposition, tensor train, hierarchical Tucker) are used for the economic solution of the multidimensional problems of the Boltzmann equation type [7,8,10].



The canonical decomposition enables to efficiently approximate and store the multi-dimensional functions. The computer time cost for the operations with the functions in the six-dimensional space (at using 100 nodes along each coordinate that formally requires storage and operations with  $10^{12}$  numbers) takes 2-3 minutes of the personal computer (processor Intel I5, 2.66 GHz) at memory requirements for cores store about  $10^5$  numbers.

## Conclusion

The applicability of the tensor statements of the computational fluid dynamics problems is restricted by the “curse of dimensionality”. Fortunately, the application of the tensor decompositions provides some hope for its overcoming.

The algorithm is offered that unifies the alternate least squares and the stochastic gradient descent and requires the memory, which is far less if compared with the methods using Khatri-Rao product.

The numerical experiments demonstrate that the application of this algorithm for the canonical decomposition enables storing and visualization of functions in the multidimensional space with the very moderate costs from the standpoint of the memory and the time of computation. The results of the visualization of the performed numerical experiments are provided.

## References

1. W. Hackbusch. Tensor Spaces and Numerical Tensor Calculus. Springer, 2012.
2. H. Yorick, S. Willi-Hans, Matrix Calculus, Kronecker Product and Tensor Product: A Practical Approach to Linear Algebra, Multilinear Algebra and Tensor Calculus with Software Implementation, Singapore: World Scientific Publishing Co. Pte. Ltd., 2019.
3. Alekseev, A.K., Bondarev, A.E., Galaktionov, V.A., Kuvshinnikov A.E. Generalized Computational Experiment and Verification Problems. Program Comput Soft 47, 177–184 (2021). <https://doi.org/10.1134/S0361768821030026>
4. Farrell B.F. and A.M. Moore, An adjoint method for obtaining the most rapidly growing perturbation to oceanic flows, J. Phys. Oceanogr, 22 338-349, 1992
5. Alekseev A.K., Bondarev A.E., On the application of the dynamic mode decomposition in problems of computational fluid dynamics, KIAM Preprints. 2018, N 154. p. 30
6. A.K. Alekseev, D.A. Bistrián, A.E. Bondarev, I.M. Navon, On Linear and Nonlinear Aspects of Dynamic Mode Decomposition, Int. J. Numer. Meth. Fluids, 2016, V. 82, Issue 6, p. 348–371
7. A. M. P. Boelens, D. Venturi, D. M. Tartakovsky, Parallel tensor methods for high-dimensional linear PDEs, J. Computat. Phys. 375 (2018) 519-539.
8. Arnout M. P. Boelens, Daniele Venturi, Daniel M. Tartakovsky, Tensor methods for the Boltzmann-BGK equation, arXiv:1911.04904v2 2020
9. Oseledets I. V., Tensor-train decomposition, SIAM J. Sci. Comput. , 33 (2011), pp. 2295–2317
10. A.V. Chikitkin, E.K. Kornev, V.A. Titarev, Numerical solution of the Boltzmann equation with S-model collision integral using tensor decompositions, arXiv:1912.04582v1 2019
11. Kolda T. G. and Bader B. W., Tensor Decompositions and Applications, *SIAM Review*, 51(3):455–500, 2009.
12. Korenev G.V., Tensor calculus, M. MIPT, 1995
13. V. D. Silva and L.-H. Lim, Tensor rank and the ill-posedness of the best low rank approximation problem, SIAM J. Matrix Anal. Appl., 30(3) 1084-1127, 2008.
14. L-H. Lim and Pierre Comon, Nonnegative approximations of nonnegative tensors, Chemometrics 2009; 23: 432–441.

15. P. Comon, X. Luciani, and A. L.F. De Almeida, Tensor decompositions, alternating least squares and other tales, *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 393–405, 2009.
16. A. Uschmajew, Local convergence of the alternating least squares algorithm for canonical tensor approximation, *SIAM J. Matrix Anal. Appl.* 33 (2) (2012) 639–652.
17. C. Battaglino, G. Ballard, T.G. Kolda, A practical randomized CP tensor decomposition, arXiv:1701.06600, 2017.
18. X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, Block-randomized stochastic proximal gradient for low-rank tensor factorization, *IEEE Transactions on Signal Processing*, 2020.
19. Ioanna Siaminou, Athanasios P. Liavas, An Accelerated Stochastic Gradient for Canonical Polyadic Decomposition, arXiv:2109.13964v1 2021.
20. Guoxu Zhou, Andrzej Cichocki, and Shengli Xie, Accelerated Canonical Polyadic Decomposition by Using Mode Reduction, arXiv:1211.3500v2, 2013
21. Tikhonov, A.N., Arsenin, V.Y.: *Solutions of Ill-Posed Problems*. Winston and Sons, Washington DC (1977).
22. R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola, A Surrogate Visualization Model Using the Tensor Train Format, SA '16: SIGGRAPH ASIA 2016 Symposium on Visualization November 2016 Article No. 13 Pages 1–8 <https://doi.org/10.1145/3002151.3002167>
23. Susanne K. Suter, *Tensor Approximation in Visualization and Graphics: Background Theory*, PhD thesis, University of Zurich, Switzerland, April 2013
24. Oseledets I., Tyrtshnikov E., TT-cross approximation for multidimensional arrays, *Linear Algebra Appl.*, 432 (2010), pp. 70–88.