Institute of Numerical Mathematics
Russian Academy of Sciences
119333, Moscow, Gubkina 8
www.inm.ras.ru

$\begin{bmatrix} \text{P} & \text{U} & \text{B} \\ \text{I} & \text{N} & \text{M} \\ \text{R} & \text{A} & \text{S} \end{bmatrix}$
pub.inm.ras.ru

S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov

Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case

Preprint 2010-01 12-04-2010

Moscow 2010

Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case

S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov

Institute of Numerical Mathematics, Russian Academy of Sciences, Russia, 119333 Moscow, Gubkina 8
[sergei.goreinov,ivan.oseledets,dmitry.savostyanov]@gmail.com

April 12, 2010

Abstract. We propose algorithms for Tucker approximation of 3-tensor, that use it only through tensor-by-vector-by-vector multiplication subroutine. In matrix case, Krylov methods are methods of choice to approximate dominant column and row subspace of sparse or structured matrix given through matrix-by-vector subroutine. However, direct generalization to tensor case, proposed recently by Eldén and Savas, namely minimal Krylov recursion, does not have any convergence theory in background and in certain cases fails to compute an approximation with desired accuracy. Using Wedderburn rank reduction formula, we propose algorithm of matrix approximation that computes Krylov subspaces and allows generalization to tensor case. Several variants of proposed tensor algorithms differ by pivoting strategies, overall cost and quality of approximation. By convincing numerical experiments we show that proposed methods are faster and more accurate than minimal Krylov recursion.

Keywords: Multidimensional arrays, sparse tensors, structured tensors, Tucker approximation, Krylov subspace methods, Wedderburn rank reduction, fast compression
AMS classification: 15A12, 65F10, 65F15.

1. Introduction

In this paper we focus on algorithms for low-rank approximation of large three-dimensional arrays (tensors), that plays increasingly important role in many applications. Throughout the paper, a *tensor* $\mathbf{A} = [a_{ijk}]$ means an array with three indices.

[$]This work was supported by RFBR grants 08-01-00115, 09-01-00565, 09-01-12058, 10-01-00757, RFBR/DFG grant 09-01-91332, Russian Federation Gov. contracts No. П1178 and П940, Rus. President grant MK–127.2009.1 and Priority research program of Dep. Math. RAS No. 3 and 5.

The indices are also called *modes* or *axes*. The number of allowed values for a mode index is called the *mode size*. In numerical work with tensors of large mode size it is crucial to look for data sparse structures. Most common are the following.

The *canonical decomposition* [1, 2, 3] (or *canonical approximation*, if right-hand side does not give $\mathbf{A}$ exactly) of a tensor $\mathbf{A} = [a_{ijk}]$ reads

$$\mathbf{A} = \sum_{s=1}^{R} u_s \otimes v_s \otimes w_s, \qquad a_{ijk} = \sum_{s=1}^{R} u_{is} v_{js} w_{ks}. \tag{C}$$

The minimal possible number of summands is called *tensor rank* or *canonical rank* of given tensor $\mathbf{A}$. However, canonical decomposition/approximation of a tensor with minimal value of R is a rather ill-posed and computationally unstable problem [4], and among several practical algorithms none is known to be absolutely robust.

The (truncated) *Tucker decomposition/approximation* [5] of $\mathbf{A}$ reads

$$\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W, \qquad a_{ijk} = \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{s=1}^{r_3} g_{pqs} u_{ip} v_{jq} w_{ks}. \tag{T}$$

The quantities r_1, r_2, r_3 are referred to as *Tucker ranks* or *mode ranks*, the tensor $\mathbf{G} = [g_{pqs}]$ of size $r_1 \times r_2 \times r_3$ is called the *Tucker core*. In d dimensions, the memory to store $r \times r \times \ldots \times r$ core is r^d, that is usually beyond affordable for large d and even very small r (so-called *curse of dimensionality*). In three dimensions for $r \sim 100$ the storage for core is small and Tucker decomposition can be used efficiently.

Here and below, the symbol $\times_l$ designates the multiplication of a tensor by a matrix along the l-th mode. For example, $\mathbf{B} = \mathbf{A} \times_2 M$ means summation on 2nd index $b_{ijk} = \sum_{j'} m_{jj'} a_{ij'k}$. The notation for tensor operations is not yet standard (for current state of art see review [6]), we use ones proposed by de Lathauwer in the article on *multilinear SVD* [7] (or high-order SVD, HOSVD). In [7] Tucker format arises with additional constrains: orthogonality of factors, that is also assumed in our paper, and all-orthogonality of the core, that we will relax. For tensor given as full array of elements, multilinear SVD provides a quasi-optimal Tucker approximation which further can be refined by different iterative methods such that Tucker-ALS [8, 9], Newton-Graßmann [10, 11], etc. It reduces to SVD for three *unfolding matrices* that costs $\mathcal{O}(n^4)$ operations for $n_1 \times n_2 \times n_3$ tensor.[1] It is clearly too much for large tensors, and we should look for alternatives.

In the matrix case there are two important classes of fast methods: cross algorithms and Krylov-based approaches. Cross methods compute rank-r approximation by interpolating the $n_1 \times n_2$ matrix on a cleverly chosen set of *crosses*, proposed for instance in [12, 13]. That requires $\mathcal{O}(nr)$ evaluations of matrix elements and can be used for matrices implicitly given by a subroutine that allows evaluation of any prescribed element. However, the verification of the approximation requires several heuristics. If the matrix is structured (sparse, Toeplitz or Hankel, low-rank, sum and/or product of

[1]We always assume $n_1 = n_2 = n_3 = n$ and $r_1 = r_2 = r_3 = r$ in complexity estimates

above, etc.) and a fast matrix-by-vector product is available, then Krylov-type methods are methods of choice with established convergence and complexity estimates.

The generalization of cross method to three-dimensional tensors (Cross3D) was presented in [14] and required careful algorithmic implementation and several tweaks to make it really efficient. In generic case, Cross3D interpolates given tensor on $\mathcal{O}(nr+r^3)$ elements and uses $\mathcal{O}(nr^2+r^4)$ additional operations. However, error checking also requires heuristics.

Recently, Eldén and Savas [15] proposed two generalization of Krylov methods to tensors: minimal and maximal Krylov recursion. The minimal recursion requires only $3r$ tensor-by-vector-by-vector multiplications to compute rank-r basis sets U, V, W for Tucker approximation (T), but sometimes it converges slowly and even does not converge (for example, for tensors with sufficiently different mode ranks). The maximal recursion is convergent, but needs definitely unaffordable number of tensor operations. The goal of this paper is to propose algorithms that use tensor only through tensor-by-vector-by-vector multiplication subroutine, has asymptotical complexity equal to one of minimal Krylov recursion whereas has better convergence properties.

In section 2 we propose optimization of minimal Krylov recursion, inspired by idea of maximization of orthogonal component of new vectors. In section 3 we recall Wedderburn rank reduction formula that give nice framework for many matrix factorizations [16], and propose a variant of matrix decomposition similar to Gaussian elimination with pivoting. In section 4 we generalize it to tensor case and propose several 'pivoting' strategies (how to choose vectors of Wedderburn process), that lead to different complexity estimates and convergence properties. Here we also show that with the certain pivoting strategy the proposed method turns into optimized minimal Krylov recursion. In section 5 we apply proposed algorithms for approximation of different structured tensors and give numerical comparison with previous methods.

The *tensor-by-vector-by-vector multiplication*, shortly *tenvec* operation, can be defined via tensor-by-matrix products. For example, tenvec of $n_1 \times n_2 \times n_3$ tensor $\mathbf{A} = [a_{ijk}]$ at modes $2, 3$ reads

$$u = \mathbf{A} \times_2 v^T \times_3 w^T, \qquad u_i = \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} a_{ijk} v_j w_k$$

where v, w are vectors of length n_2 and n_3 respectively and the result is a vector u of length n_1. This is the solely tensor operation in algorithms in this paper and we propose to use extremely simple notation

$$\mathbf{A} \times_2 v^T \times_3 w^T \overset{\text{def}}{=} \mathbf{A}vw, \qquad \mathbf{A} \times_3 w^T \times_1 u^T \overset{\text{def}}{=} \mathbf{A}wu, \qquad \mathbf{A} \times_1 u^T \times_2 v^T \overset{\text{def}}{=} \mathbf{A}uv,$$

relaxing the information on contraction modes, that is always obvious from symbolic notation and mode sizes of vectors. This is a generalization of *Tdd* notation used for instance in [17]. The simplicity of notation helps to illustrate the direct analogy of tensor algorithms with matrix case.

We consider approximation of matrices and tensors in Frobenius norm, that is defined for tensors as follows

$$\|\mathbf{A}\|_F^2 \overset{\text{def}}{=} \langle \mathbf{A},\, \mathbf{A} \rangle\,, \qquad \langle \mathbf{A},\, \mathbf{B} \rangle \overset{\text{def}}{=} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} a_{ijk} b_{ijk}.$$

For theoretical estimates we will also use spectral norm of tensor (cf. [18])

$$\|\mathbf{A}\|_2 \overset{\text{def}}{=} \max_{\|x\|=\|y\|=\|z\|=1} \mathbf{A} \times_1 x^T \times_2 y^T \times_3 z^T = \max_{\|x\|=\|y\|=\|z\|=1} \langle \mathbf{A},\, x \otimes y \otimes z \rangle\,,$$

induced by standard vector norm $\|x\|^2 \overset{\text{def}}{=} \|x\|_2^2 = (x, x) = \sum_{i=1}^n |x_i|^2$.

It is worth to mention that discussed algorithms first aim to construct orthogonal bases U, V and W of the dominant mode subspaces of given tensor, and the core $\mathbf{G}$ of Tucker approximation (T) can be computed afterwards. For fixed U, V, W the most accurate approximation in Frobenius norm is obtained with core

$$\mathbf{G} = \mathbf{A} \times_1 U^T \times_2 V^T \times_3 W^T. \tag{1}$$

In general this core can be computed using r^2 tenvecs, but faster implementation of (1) are available for certain structured tensors and can essentially improve the complexity. We can also be satisfied with sub-optimal but fast formula for $\mathbf{G}$, for example based on interpolation on maximum-volume set of indices [19] as proposed in [20]. If computation of (1) is not a neccesary part of discussed algorithm, this cost is not included in the total complexity.

2. Minimal Krylov recursion and its optimization

The generalization of Krylov recursion to the problem of tensor approximation was first proposed by Eldén and Savas in [15]. Two variants are proposed there: so-called minimal and maximum recursion.

Minimal Krylov recursion (MKR, see Alg. 1) requires 3 tenvecs on each iteration, therefore, basis sets $U = U_r, V = V_r, W = W_r$ are computed in $3r$ tenvecs. However, the 'quality' of these bases, i.e. accuracy of rank-(r, r, r) Tucker approximation $\tilde{\mathbf{A}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ with optimal core (1) may be low, and for some cases $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ will not reduce at all, even for large ranks.

As an example, consider an $n \times n \times n$ tensor $\mathbf{A}$ with only two non-zero slices[2]

$$\mathbf{A}(:, :, 1) = A, \qquad \mathbf{A}(:, :, 2) = B, \qquad \text{and} \quad \mathbf{A}(:, :, 3 : n) = 0.$$

Obviously, if $A \neq B$, mode-3 subspace of $\mathbf{A}$ consists of two vectors $W = [e_1\ e_2]$, and mode-1 and mode-2 subspaces depend on ranks of A and B and can be large. Starting MKR Alg. 1 from some u_1, v_1, we can immediately accumulate $W_2 = [e_1 e_2]$. After that

[2]Here and after for readability we use MATLAB-style notation, with ":" denoting all possible values of certain index, and $a : b$ denoting index values running from a to b

Algorithm 1: [15] Minimal Krylov recursion for tensor approximation (MKR)

```
Input: Tenvec subroutine for tensor $\mathbf{A}$
Output: Mode subspaces $U, V, W$ for Tucker approximation
Initialization: Unit vectors $u_1, v_1$
  $w_1 := \mathbf{A}u_1v_1/\|\mathbf{A}u_1v_1\|, \quad U_1 = [u_1], V_1 = [v_1], W_1 = [w_1]$
  for $k = 1, 2, \dots$ do
    $u := \mathbf{A}v_kw_k; \quad u' := (I - U_kU_k^T)u; \quad u_{k+1} := u'/\|u'\|; \quad U_{k+1} := [U_k\, u_{k+1}]$
    $v := \mathbf{A}w_ku_{k+1}; \quad v' := (I - V_kV_k^T)v; \quad v_{k+1} := v'/\|v'\|; \quad V_{k+1} := [V_k\, v_{k+1}]$
    $w := \mathbf{A}u_{k+1}v_{k+1}; \quad w' := (I - W_kW_k^T)w; \quad w_{k+1} := w'/\|w'\|; \quad W_{k+1} := [W_k\, w_{k+1}]$
  end for
```

all computed vectors w have zero component w' orthogonal to W_2. This situation is referred to as *breakdown*, since mode-3 basis can not be expanded.

Eldén and Savas [15] propose to fix breakdowns by taking an orthogonal vector to previously computed W. However with $w_3 := e_3$ we come to zero vector $u := \mathbf{A}v_3w_3$ on the next iteration and can not continue the process.

Another possible workaround is to use last non-zero vector from sequence of w_k on all subsequent iterations, setting $w_3 := w_2, w_4 := w_2$ and so on. If $w_2 = e_2$, the computation reduces to approximation of matrix B. Final approximation for B will be accurate, but only rank-one approximation is constructed for A, since only one tenvec with $w = e_1$ was used.

In the numerical examples (see section 5.2) we show that poor convergence and even stagnation can be the case also for 'more practical' examples of tensors.

From this example we derive the idea of optimization of Alg. 1. Consider step for u and write it as follows.

$$u := \mathbf{A}(V_k\hat{v}_k)(W_k\hat{w}_k); \quad u' := (I - U_kU_k^T)u; \quad u_{k+1} := u'/\|u'\|; \quad U_{k+1} := [U_k\, u_{k+1}].$$

New direction u is generated by tenvec of $\mathbf{A}$ with *some* vector from span V and *some* vector from span W. In MKR we always take $\hat{v}_k = \hat{w}_k = e_k$, but sometimes that does not lead to convergence. To improve the approximation properties of U_k, we could choose $\hat{v}_k$ and $\hat{w}_k$ to maximize the norm of orthogonal component u'. Since

$$u' = (I - U_kU_k^T)u = \left(\mathbf{A} \times_1 (I - U_kU_k^T) \times_2 V_k^T \times_3 W_k^T\right) \times_2 \hat{v}_k^T \times_3 \hat{w}_k^T,$$

we are to find

$$\hat{v}_k, \hat{w}_k = \arg \max_{\|\hat{v}\|=1, \|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\| \qquad \text{for} \quad \mathbf{B} = \mathbf{A} \times_1 (I - U_kU_k^T) \times_2 V_k^T \times_3 W_k^T. \quad (2)$$

Global maximization is not required, 'sufficiently large' vector u' is enough. It can be found by several iterations of ALS (*alternating least squares*) method (see [8] and Alg. 2), for which the local linear convergence is proved in rank-$(1, 1, 1)$ case [9, 21]. If ALS converges to best rank-one approximation, it also solves (2) as dual problem. To

Algorithm 2: [8] ALS rank-$(1,1,1)$ iteration

```
Input: Tenvec subroutine for tensor $\mathbf{A}$
Output: Best rank-$(1,1,1)$ approximation $\sigma u \otimes v \otimes w$ for tensor
Initialization: Unit vectors $v, w$
  for $k = 1, \dots, p_{als}$ do
    $u := \mathbf{A}vw$;   $\sigma := \|u\|$;   $u := u/\|u\|$
    $v := \mathbf{A}wu$;   $\sigma := \|v\|$;   $v := v/\|v\|$
    $w := \mathbf{A}uv$;   $\sigma := \|w\|$;   $w := w/\|w\|$
  end for
```

see this, consider approximation $\tilde{\mathbf{B}} = bu \otimes \hat{v} \otimes \hat{w}$, with unit $u, \hat{v}, \hat{w}$ and compute optimal $1 \times 1 \times 1$ core by (1)

$$b := \mathbf{B} \times_1 u^T \times_2 \hat{v}^T \times_3 \hat{w}^T = \langle \mathbf{B},\ u \otimes \hat{v} \otimes \hat{w} \rangle = (\mathbf{B}\hat{v}\hat{w}, u).$$

The approximation $\tilde{\mathbf{B}}$ that provides minimal approximation error

$$\|\mathbf{B} - \tilde{\mathbf{B}}\|_F^2 = \|\mathbf{B}\|_F^2 - 2\langle \mathbf{B},\ bu \otimes \hat{v} \otimes \hat{w} \rangle + \|bu \otimes \hat{v} \otimes \hat{w}\|_F^2 = \|\mathbf{B}\|_F^2 - |b|^2$$

also solves (2), since it maximizes $|b|$ and

$$\max_{\|u\|=\|\hat{v}\|=\|\hat{w}\|=1} |b| = \max_{\|\hat{v}\|=\|\hat{w}\|=1} \max_{\|u\|=1} (\mathbf{B}\hat{v}\hat{w}, u) = \max_{\|\hat{v}\|=\|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\|.$$

Each iteration of ALS requires 3 tenvecs with $\mathbf{B}$, that can be expressed by 3 tenvecs with $\mathbf{A}$ and $\mathcal{O}(nk)$ operations for orthogonalization to U_k. Therefore, optimization of minimal Krylov recursion can be summarized in terms of tenvecs operations, see Alg. 3. If p_{als} ALS iterations are used to solve (2) on Step 6, then Alg. 3 requires $(3 + 9p_{als})r$ tenvecs and $\mathcal{O}(nr^2 p_{als})$ additional operations.

In numerical experiments (see section 5) we show that optimized minimal Krylov recursion shows better convergence that minimal Krylov recursion. However, convergence estimates are still missing, even in the exact low-rank case. Is it possible to provide a method that has guaranteed convergence at least in the exact case? We start from matrix case, using Wedderburn rank-one reduction formula. Then, in section 4, we propose generalization of Wedderburn method to tensor case and show that optimized minimal Krylov recursion Alg. 3 arises as a version of Wedderburn process. Then we will discuss the convergence properties of this method and give convincing numerical examples and comparison.

3. Matrix approximation using Wedderburn rank reduction formula

3.1. Preliminaries

Many matrix decomposition algorithms can be represented as a sequence of rank-one *Wedderburn updates* [22]. For a given matrix A and vectors x, y of appropriate sizes,

Algorithm 3: Optimized minimal Krylov recursion for tensor approximation

```
Input: Tenvec subroutine for tensor $\mathbf{A}$, tolerance parameter tol
Output: Mode subspaces $U, V, W$ for approximation $\mathbf{A} \approx \tilde{\mathbf{A}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$
Initialization: Unit vectors $u_1, v_1$
 1: $w_1 := \mathbf{A} u_1 v_1 / \|\mathbf{A} u_1 v_1\|, \quad U_1 = [u_1], V_1 = [v_1], W_1 = [w_1]$
 2: $updU = updV = updW =$ true;  $k = l = m = 1$
 3: while $updU$ or $updV$ or $updW$ do
 u:   {Proceed with new vector $u$ if required}
 4:   if $updU$ then
 5:     Define $\mathbf{B} = \mathbf{A} \times_1 (I - U_k U_k^T) \times_2 V_l^T \times_3 W_m^T$
 6:     Find $\hat{v}_k, \hat{w}_k := \arg\max_{\|\hat{v}\|=1, \|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\|$ by $p_{\mathrm{als}}$ ALS steps, see Alg. 2
 7:     $u := \mathbf{A}(V_l \hat{v}_k)(W_m \hat{w}_k); \quad u' := (I - U_k U_k^T) u$
 8:     if $\|u'\| <$ tol$\|u\|$ then {Breakdown}
 9:       Fix breakdown or set $updU :=$ false
10:     else
11:       $u_{k+1} := u'/\|u'\|; \quad U_{k+1} := [U_k \; u_{k+1}]; \quad k := k+1$
12:     end if
13:   end if
 v:   {Proceed with new vector $v$ if required}
 w:   {Proceed with new vector $w$ if required}
14: end while
```

such that $x^T A y \neq 0$, matrix

$$B = A - \frac{A y x^T A}{x^T A y}. \tag{3}$$

has rank $B = \operatorname{rank} A - 1$. For the rank-$r$ matrix $A_0 = A$ after r updates of form

$$A_k = A_{k-1} - \frac{A_{k-1} y_k x_k^T A_{k-1}}{x_k^T A_{k-1} y_k} \tag{4}$$

with $\omega_k \overset{\mathrm{def}}{=} x_k^T A_{k-1} y_k \neq 0$, the matrix A_r becomes zero and rank-r decomposition of A can be constructed.

In [16] the properties of Wedderburn sequence (4) are studied in much details. We start from a list of elementary facts about (4), written in a compact and 'more matrix' form. The detailed proof can be found in [16].

Statement 1. Matrix A_k reads $A_k = P_k^T A = A Q_k$ with $P_0 = I$, $Q_0 = I$ and

$$P_k = P_{k-1} - \omega_k^{-1} P_{k-1} x_k y_k^T A^T P_{k-1}, \qquad Q_k = Q_{k-1} - \omega_k^{-1} Q_{k-1} y_k x_k^T A Q_{k-1}. \tag{5}$$

Corollary 1. $P_k x_k = 0$ and $Q_k y_k = 0$.

Statement 2. Matrices P_k, Q_k are projectors, since $P_0 = P_0^2 = I$, $Q_0 = Q_0^2 = I$ and from $P_{k-1}^2 = P_{k-1}$ it follows that

$$P_k^2 = P_{k-1}^2 - \omega_k^{-1} P_{k-1}^2 y_k x_k^T A^T P_{k-1} - \omega_k^{-1} P_{k-1} x_k y_k^T A^T P_{k-1}^2 + \\ + \omega_k^{-2} P_{k-1} x_k \underbrace{y_k^T A^T P_{k-1}^2 x_k}_{\omega_k} y_k^T A^T P_{k-1},$$

so $P_k^2 = P_k$ and similarly $Q_k^2 = Q_k$ hold.

Statement 3. $P_k P_m = P_m P_k = P_{\max(m,k)}$, and similarly $Q_k Q_m = Q_m Q_k = Q_{\max(m,k)}$. Start from $Q_k Q_{k-1} = Q_{k-1}^2 - \omega_k^{-1} Q_{k-1} y_k x_k^T A Q_{k-1}^2 = Q_{k-1} - \omega_k^{-1} Q_{k-1} y_k x_k^T A Q_{k-1} = Q_k$ and complete the proof by induction.

Corollary 2. For $m \leqslant k$ it holds $P_k x_m = P_k P_m x_m = 0$ and $Q_k y_m = Q_k Q_m y_m = 0$.

Statement 4. For *biconjugate vectors* $u_k \overset{\text{def}}{=} P_{k-1} x_k$ and $v_k \overset{\text{def}}{=} Q_{k-1} y_k$ it holds

$$u_k^T A v_k = x_k^T A_{k-1} y_k \overset{\text{def}}{=} \omega_k,$$

since $A_{k-1} = A Q_{k-1} = A Q_{k-1}^2 = A_{k-1} Q_{k-1} = P_{k-1}^T A Q_{k-1}$. Also for $m \neq k$ it holds

$$u_m^T A v_k = x_m^T P_{m-1}^T A Q_{k-1} y_k = x_m^T A_{\max(m-1,k-1)} y_k = 0.$$

With $U_k \overset{\text{def}}{=} [u_1 \ \dots \ u_k]$, $V_k \overset{\text{def}}{=} [v_1 \ \dots \ v_k]$ and $\Omega_k \overset{\text{def}}{=} \operatorname{diag}(\omega_1 \ \dots \ \omega_k)$ we conclude

$$U_k^T A V_k = \Omega_k. \tag{6}$$

Corollary 3. For valid Wedderburn process U_k and V_k have full rank.

Corollary 4. For $m \leqslant k$ it holds $P_k u_m = P_k P_{m-1} x_m = P_k x_m = 0$ and $Q_k v_m = 0$.

Statement 5. Each step of rank elimination (4) can be written without multiplication by A_{k-1} as follows

$$A_k = A_{k-1} - \frac{A v_k u_k^T A}{u_k^T A v_k} = A - \sum_{p=1}^{k} A v_p \omega_p^{-1} u_p^T A.$$

Introducing rank-k *Wedderburn approximation*

$$\tilde{A}_k \overset{\text{def}}{=} A V_k \Omega_k^{-1} U_k^T A, \tag{7}$$

we have

$$A = A_k + \tilde{A}_k. \tag{8}$$

Obviously, for rank-r matrix A residual $A_r = 0$ and approximation $\tilde{A}_r$ is exact.

We see that each Wedderburn update (4) adds vector y_k to kernel and x_k to cokernel of 'residual' $A_k = A - \tilde{A}_k$. The approximation $\tilde{A}_k$, as linear operator, interpolates A on subspaces spanned by X_k and Y_k, exactly

$$X_k^T \tilde{A}_k = X_k^T A, \quad \tilde{A}_k Y_k = A Y_k, \qquad U_k^T \tilde{A}_k = U_k^T A, \quad \tilde{A}_k V_k = A V_k. \tag{9}$$

This can be associated with *Gaussian elimination* of certain rows and columns from matrix, and in this respect we refer to process (4) as to *Wedderburn elimination.*

In [16] it is shown, that proper choice of x_k, y_k can reduce Wedderburn elimination to well-known matrix decompositions such as LU, QR, cross factorizations and Lanczos bidiagonalization. We consider another principle for the selection of vectors x_k, y_k at each step in the matrix case, that can be associated with Gaussian elimination with column or row pivoting. It produces a new method for matrix approximation, that is simply generalized to tensors while maintaining convergence.

3.2. Pivoting in Wedderburn elimination

The idea behind the proposed choice of x_k, y_k is minimization of Frobenius norm of residual, that is important when we deal with full-rank matrices, that have low ε-rank, i.e. can be approximated by low-rank matrix with accuracy ε. As shown in [16], minimization of residual w.r.t. unit x_k, y_k gives k-th singular vectors of A, that are not known in advance. We propose another minimization strategy, based on the following theorem.

Theorem 1. Consider Wedderburn step (3). Then

$$\text{for fixed } y, \qquad x_{\text{opt}} \overset{\text{def}}{=} \arg \min_{\|x\|=1} \left\| A - \frac{Ayx^T A}{x^T A y} \right\|_F = \frac{Ay}{\|Ay\|}; \tag{10}$$

$$\text{for fixed } x, \qquad y_{\text{opt}} \overset{\text{def}}{=} \arg \min_{\|y\|=1} \left\| A - \frac{Ayx^T A}{x^T A y} \right\|_F = \frac{A^T x}{\|A^T x\|}. \tag{11}$$

Proof. Since valid Wedderburn step does not depend on scaling of x, y, for any fixed y we can constrain x to satisfy $x^T A y = 1$. Then

$$x_{\text{opt}} = \min_{x:\, x^T A y = 1} \left\| (Ay)(A^T x) - A \right\|_F^2 = \min_{x:\, x^T A y = 1} \left\| (A^T x)(Ay)^T - A^T \right\|_F^2.$$

This least squares problem solves by $(A^T x)_{\text{opt}} = A^T A y / \|Ay\|^2$ and $x_{\text{opt}} = Ay/\|Ay\|^2$. After normalization we have (10). Statement (11) follows by substituting $A = A^T$. $\square$

Eqs. (10) and (11) show how to reach fast decay of residual in Wedderburn process (4) either by choosing optimal x_k for given y_k or by choosing optimal y_k for given x_k. This can be associated with column and row pivoting in Gaussian elimination. Thus, we refer to Wedderburn process (4) with arbitrary y_k and optimal x_k chosen by (10) as to *Wedderburn elimination with column pivoting* (WCP) and to Wedderburn process with arbitrary x_k and optimal y_k chosen by (11) as to *Wedderburn elimination with row pivoting* (WRP). The steps of WCP and WRP shortly reads

$$\text{choose } y_k, \text{ set} \quad x_k = \frac{A_{k-1}y_k}{\|A_{k-1}y_k\|}, \quad A_k = A_{k-1} - \frac{x_k x_k^T A_{k-1}}{x_k^T x_k} = \left(I - x_k x_k^T\right) A_{k-1}; \quad \text{(WCP)}$$

$$\text{choose } x_k, \text{ set} \quad y_k = \frac{A_{k-1}^T x_k}{\|A_{k-1}^T x_k\|}, \quad A_k = A_{k-1} - \frac{A_{k-1} y_k y_k^T}{y_k^T y_k} = A_{k-1}\left(I - y_k y_k^T\right). \quad \text{(WRP)}$$

Theorem 2. For Wedderburn elimination with column pivoting (WCP) it holds

1. $X_k = [x_1 \dots x_k]$ has orthonormal columns $X_k^T X_k = I$;
2. $P_k = P_k^T = I - X_k X_k^T$ is projector on subspace orthogonal to $\operatorname{span} X_k$;
3. biconjugate vectors $u_k \overset{\text{def}}{=} P_{k-1} x_k = x_k$.

For Wedderburn elimination with row pivoting (WRP) and $Y_k = [y_1 \dots y_k]$ it holds

1. $Y_k^T Y_k = I$;
2. $Q_k = Q_k^T = I - Y_k Y_k^T$;
3. $v_k \overset{\text{def}}{=} Q_{k-1} y_k = y_k$.

Proof. Let us prove the statements for WRP. It is easy to check them for $k = 1$. Suppose they hold at step $k-1$. Optimal choice of y_k by (11) reads

$$y = A_{k-1}^T x_k = Q_{k-1}^T A^T x_k = (I - Y_{k-1} Y_{k-1}^T) A^T x_k, \qquad y_k = \frac{y}{\|y\|}.$$

This shows $\|y_k\| = 1$ and $Y_{k-1}^T y_k = 0$ and first statement follows for $Y_k := [Y_{k-1}\, y_k]$. By substituting $y_k = A_{k-1}^T x_k / \|A_{k-1}^T x_k\|$ in (5) we prove the second statement

$$Q_k = Q_{k-1} - \frac{Q_{k-1} y_k x_k^T A_{k-1}}{x_k^T A_{k-1} y_k} = Q_{k-1}\left(I - \frac{y_k y_k^T}{y_k^T y_k}\right) =$$
$$= \left(I - Y_{k-1} Y_{k-1}^T\right)\left(I - y_k y_k^T\right) = I - Y_{k-1} Y_{k-1}^T - y_k y_k^T = I - Y_k Y_k^T.$$

Finally, last statement reads

$$v_k \overset{\text{def}}{=} Q_{k-1} y_k = Q_{k-1} \frac{Q_{k-1}^T A^T x_k}{\|A_{k-1}^T x_k\|} = \frac{Q_{k-1}^2 A^T x_k}{\|A_{k-1}^T x_k\|} = \frac{Q_{k-1}^T A^T x_k}{\|A_{k-1}^T x_k\|} = y_k.$$

The statements for WCP are proven in the same way. □

Based on this theorem, we propose the Wedderburn elimination algorithm with column pivoting (WCP, see Alg. 4). The approximation is sought in the form $\tilde{A}_k = X_k B_k^T$ with $B_k = A^T X_k$, that follows from (9) and orthogonality of biconjugate vectors $U_k = X_k$, result of Theorem 2. To explain stopping criteria of Alg. 4, write

$$\mathtt{nrm} \overset{\text{def}}{=} \|\tilde{A}_k\|_F = \|X_k X_k^T A\|_F = \|A^T X_k\|_F = \|B_k\|_F, \qquad \mathtt{err} \overset{\text{def}}{=} \|\tilde{A}_k - \tilde{A}_{k-1}\|_F = \|b_k\|_F,$$

Algorithm 4: Wedderburn elimination with column pivoting (WCP)

```
Input: Matvec subroutine for matrix $A$, tolerance parameter tol, accuracy $\varepsilon$
Output: Approximation $\tilde{A}$ with accuracy $\|A - \tilde{A}\|_F \lesssim \varepsilon \|\tilde{A}\|_F$
Initialization: $k = 0$, $X_0 = [\varnothing]$, $B_0 = [\varnothing]$, nrm $= 0$
 1: repeat
 2:   $k := k + 1$, choose unit vector $y_k$
 3:   $x := Ay_k$;   $x' := (I - X_{k-1}X_{k-1}^T)x$
 4:   if $\|x'\| < \mathtt{tol}\|x\|$ then {Breakdown}
 5:     return $\tilde{A} = X_{k-1}B_{k-1}^T$ {or repeat current iteration step with another $y_k$}
 6:   else
 7:     $x_k := x'/\|x'\|$
 8:   end if
 9:   $b_k := A^T x_k$,   err $:= \|b_k\|$,   nrm$^2 :=$ nrm$^2 + \|b_k\|^2$
10:   $X_k := [X_{k-1}\ x_k]$,   $B_k := [B_{k-1}\ b_k]$
11: until err $\leqslant \varepsilon$ nrm
12: return $\tilde{A} = X_k B_k^T$
```

where first part estimates norm of matrix by norm of approximation and second estimates error of approximation by internal convergence at current step. The *breakdown* can occur if new vector x have small component x' orthogonal to previously computed X_{k-1}. This is regulated by tolerance parameter tol, that could be chosen close to machine precision. We can try to fix breakdown by repeating the current step with another selection of y_k, or choose to terminate the algorithm.

The WRP version of algorithm follows by substituting $A = A^T$.

3.3. Relation to SVD and Lanczos bidiagonalization

On each step of WCP Alg. 4 we perform Wedderburn elimination, choosing optimal 'pivot' x_k for given y_k. It is also important to select proper 'leading vectors' y_k to obtain faster convergence of approximation and avoid breakdowns. We could think about maximization of $\omega_k = x_k^T A_{k-1} y_k = \|A_{k-1}y_k\| = \|x'\|$. By solving maximization problem

$$y_k = \arg\max_{\|y\|=1} \|A_{k-1}y\| = \arg\max_{\|y\|=1} \left\|(I - X_{k-1}X_{k-1}^T)Ay\right\| \tag{12}$$

exactly, we reduce the Wedderburn process to a sequence of best rank-one approximations with ω_k and x_k, y_k being singular values of A (sorted descending) and corresponding left and right singular vectors. Since SVD provides best rank-r approximation in Frobenius norm, WCP with selection of leading vector by (12) leads to fastest possible convergence of $\tilde{A}_k$ to A. This approach can be associated with *full pivoting* in Gaussian elimination. Each maximization problem (12) can be accurately solved by power iterations using only matvec operations, but in matrix case this approach generally is considered as quite expensive and faster alternatives are used.

Algorithm 5: [23] Lanczos bidiagonalization

```
Initialization: $y_0 = 0, \beta_0 = 0$, unit vector $x_0$
  for $k = 1, 2, \dots$ do
    $y := A^T x_{k-1}$;   $y' := y - \beta_{k-1} y_{k-1}$,   $\alpha_k := \|y'\|$,   $y_k := y'/\|y'\|$
    $x := A y_k$;   $x' := x - \alpha_k x_k$,   $\beta_k := \|x'\|$,   $x_k := x'/\|x'\|$
  end for
```

One of them in Lanczos bidiagonalization, see [23] and Alg. 5. It generates bases $X_k = [x_1 \dots x_k]$ and $Y_k = [y_1 \dots y_k]$ such that $X_k^T X_k = I$, $Y_k^T Y_k = I$ and matrix $X_k^T A Y_k$ is bidiagonal.

The following theorem shows that Lanczos bidiagonalization is similar to Alg. 4 if leading vector is selected as follows

$$y_{k+1} = \frac{A_{k-1}^T x_k}{\|A_{k-1}^T x_k\|} = \frac{A^T x_k}{\|A^T x_k\|}. \tag{13}$$

This value is well defined, since

$$A_{k-1}^T x_k = A^T P_{k-1} x_k = A^T (I - X_{k-1} X_{k-1}^T) x_k = A^T x_k = b_k,$$

and $\|A_{k-1}^T x_k\| \overset{\text{def}}{=} \mathtt{err}$ vanishes only when stopping criteria $\mathtt{err} < \varepsilon\, \mathtt{nrm}$ is met and we do not need to select next leading vector y_{k+1}.

Theorem 3. Vectors $X^{\{lnc\}} = [x_1^{\{lnc\}}, \dots, x_k^{\{lnc\}}]$ of Lanczos process (Alg. 5) that starts from vector x_0, coincide with vectors $X_k^{\{wcp\}} = [x_1^{\{wcp\}}, \dots, x_k^{\{wcp\}}]$ generated by WCP Alg. 4 that starts from $y_1 = A^T x_0$ and chooses y_k as proposed by (13), providing both algorithms do not meet breakdowns.

Proof. If no breakdowns are met, then

$$\operatorname{span} X_k^{\{wcp\}} = \operatorname{span}\{Ay_1, (AA^T)Ay_1, \dots, (AA^T)^{k-1}Ay_1\},$$
$$\operatorname{span} X_k^{\{lnc\}} = \operatorname{span}\{(AA^T)x_0, (AA^T)^2 x_0, \dots, (AA^T)^k x_0\}.$$

If $y_1 = A^T x_0$ then for every k it holds $\operatorname{span} X_k^{\{wcp\}} = \operatorname{span} X_k^{\{lnc\}}$. Since $X_k^{\{wcp\}}$ consists of orthonormal columns (Theorem 2), as well as $X_k^{\{lnc\}}$, we conclude $X_k^{\{wcp\}} = X_k^{\{lnc\}}$. □

Thus, in terms of x_k, WCP give the same result as Lanczos bidiagonalization. However, properties of sequence y_k in WCP differ from those of Lanczos bidiagonalization. First, we note that

$$y = A_{k-1}^T A_{k-1} y_k = A^T P_{k-1} P_{k-1}^T A y_k = A^T P_{k-1} A y_k = Q_{k-1}^T A^T A y_k, \quad y_{k+1} = \frac{y}{\|y\|}$$

$$\operatorname{span} Y_k = \operatorname{span}\{y_1, (A^T A) y_1, \dots, (A^T A)^{k-1} y_1\}.$$

Sequence y_k is 'almost orthogonal', i.e. for $m \leqslant k-2$ it holds

$$y_k^T y_m = \frac{y_{k-1}^T A^T A Q_{k-2} y_m}{\|A^T A y_{k-1}\|} = 0$$

Algorithm 6: WCP with Lanczos-like selection of leading vector

```
Input: Matvec subroutine for matrix $A$, tolerance parameter tol, accuracy $\varepsilon$
Output: Approximation $\tilde{A}$ with accuracy $\|A - \tilde{A}\|_F \lesssim \varepsilon \|\tilde{A}\|_F$
Initialization: $k = 0$,  $x_0 = x_{-1} = 0$,  $\tilde{A}_0 = 0$,  $\mathrm{nrm} = 0$, unit vector $y_1$
 1: repeat
 2:   $k := k + 1$,  $x := Ay_k$;  $x' := (I - x_{k-2}x_{k-2}^T - x_{k-1}x_{k-1}^T)x$
 3:   if $\|x'\| < \mathtt{tol}\|x\|$ then {Breakdown}
 4:     return $\tilde{A} = \tilde{A}_{k-1}$
 5:   else
 6:     $x_k := x'/\|x'\|$
 7:   end if
 8:   $y_{k+1} := A^T x_k$,  $\mathrm{err} := \|y_{k+1}\|$,  $\mathrm{nrm}^2 := \mathrm{nrm}^2 + \|y_{k+1}\|^2$
 9:   $\tilde{A}_k = \tilde{A}_{k-1} + x_k y_{k+1}^T$
10: until $\mathrm{err} \leqslant \varepsilon\, \mathrm{nrm}$
11: return $\tilde{A} = \tilde{A}_k$
```

since $Q_{k-2}y_m = 0$, see Corollary 2. Also, for X_k and Y_k generated by WCP, matrix $X_k^T A Y_k$ is tridiagonal, i.e.

$$x_m^T A y_k = y_{m+1}^T y_k = 0, \qquad \text{for} \quad m \notin \{k-2, k-1, k\}.$$

That leads to the following remark.

Remark 1. For $k \geqslant 3$ vector $X_{k-1}^T x = X_{k-1}^T A y_k$ on Step 3 of Alg. 4 has only 2 nonzero components, the last ones. Hence, the orthogonalization step can be simplified to

$$x := Ay_k, \quad x' := (I - x_{k-2}x_{k-2}^T - x_{k-1}x_{k-1}^T)x.$$

This allows us to use short recursion for orthogonalization of x_k in WCP Alg. 4, that is also the case in Lanczos bidiagonalization Alg. 5. This version of WCP is summarized in Alg. 6. However, in machine arithmetics the orthogonality can be violated by rounding errors and reorthogonalization is required to stabilize the computations. The convergence of Lanczos bidiagonalization method is well studied and although this method can have breakdowns, it is considered to converge from 'almost every' initial vector [24, 25, 26].

4. Tensor approximation using Wedderburn rank reduction

4.1. Computing dominant mode subspaces by WCP algorithm

Now we are ready to propose the extension of Wedderburn elimination algorithm to tensor case. For Tucker approximation we need to find dominant subspaces U, V, W that contain most part of information about mode vectors of tensor. In [7] this is done by SVD applied to the unfoldings of an $n_1 \times n_2 \times n_3$ tensor $\mathbf{A} = [a_{ijk}]$. They are matrices

$$A^{(1)} = [a^{(1)}_{i,jk}], \quad A^{(2)} = [a^{(2)}_{j,ki}], \quad A^{(3)} = [a^{(3)}_{k,ij}], \qquad a^{(1)}_{i,jk} = a^{(2)}_{j,ki} = a^{(3)}_{k,ij} = a_{ijk}, \tag{14}$$

of size $n_1 \times n_2n_3$, $n_2 \times n_1n_3$ and $n_3 \times n_1n_2$, that consist of columns, rows and tube fibres of $\mathbf{A}$, respectively. Left singular vectors after appropriate truncation give Tucker factors U, V, W, and the core is found by convolution (1) with initial tensor. Since SVD is applied to tensors that are given as full array of elements and costs $\mathcal{O}(n^4)$ operations, it can not be used efficiently for large structured tensors. We propose to apply Wedderburn elimination to unfoldings to approximate dominant subspaces of mode vectors.

We are looking for algorithms that compute Tucker rank-(r_1, r_2, r_3) approximation of $n_1 \times n_2 \times n_3$ tensor using $\mathcal{O}(r)$ tenvecs and *linear* in mode size number of additional operations. This is asymptotically the same cost, as one of MKR Alg. 1. With this restriction for $n_1 \times n_2n_3$ unfolding $A = A^{(1)} = [a_{i;jk}]$ we generally can not compute $x = Ay$ and $y = A^T x$ for arbitrary vectors y of size n_2n_3 and x of size n_1, since these operations require $\mathcal{O}(n^2)$ storage and n tenvecs. To stay within the desired complexity bounds, we restrict operations with unfolding to those, that can be accomplished by small number of tenvecs, namely

$$x = A(y \otimes z) = \mathbf{A} \times_2 y^T \times_3 z^T = \mathbf{A}yz, \qquad y \otimes z :\approx A^T x,$$

where x is of size n_1, y of size n_2, z of size n_3, first operation assumes rank-one tensor product vector on input and second assumes that certain rank-one output is returned as approximation of $A^T x$.

In matrix case difference between WCP and WRP algorithms is not significant, since the properties of column basis in WCP coincide with ones of row basis in WRP and vice versa. In tensor case with imposed restriction multiplication $A(y \otimes z)$ is accurate, but multiplication $A^T x = \mathbf{A} \times_1 x^T$ includes error of truncation to rank one. Considering this difference, we use WCP algorithm for tensor case, since good properties of X_k given by Theorem 2 persist here. That is not the case for properties of Y_k in WRP algorithm, since every $y_k := A_{k-1}^T x_k / \|A_{k-1}^T x_k\|$ is computed approximately and orthogonality is perturbed.

Using the direct analogy with WCP Alg. 4 for matrices, we propose a method to derive dominant mode-1 subspace U, see Alg. 7. There we can use one or both of stopping criteria:

- fix maximum number of iterations, i.e. desired size of basis U by $r_{\max}$,
- find basis U that allows approximation of $\mathbf{A}$ with relative accuracy ε.

To compute estimate of error $\mathrm{err} \overset{\mathrm{def}}{=} \|\mathbf{A} \times_1 x_k^T\|_2$, we use power method for $n_2 \times n_3$ matrix $\mathbf{A} \times_1 x_k^T$, starting from some unit vector z of size n_3. During power iteration σ converges to maximum singular value of matrix $B = \mathbf{A} \times_1 x_k^T$ and $\tilde{B} = \sigma y z^T$ converges to best rank-1 approximation of B. Since high precision is not neccesary for estimation of error, we can satisfy with fixed small number p_{pow} of power iteration steps. If only fixed-rank stopping criteria is desired, power iterations on Steps 9-13 of Alg. 7 can be safely omitted.

As well as in matrix case, we can meet with *breakdown* if new vector x have small component x' orthogonal to previously computed X_{k-1}. We can try to fix it by repeating the current step with another selection of y_k, z_k, or choose to terminate the algorithm.

Algorithm 7: Wedderburn elimination for dominant subspace computation

```
Input: Tenvec subroutine for tensor $\mathbf{A}$, tolerance tol, accuracy $\varepsilon$, maximum size $r_{\max}$.
Output: Mode subspace $U$ for Tucker approximation $\tilde{\mathbf{A}}$ such that $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F \lesssim \varepsilon \|\tilde{\mathbf{A}}\|_F$
Initialization: $X_0 = [\varnothing]$,  nrm $= 0$,  $k = 0$
1:  repeat
2:    $k := k + 1$, choose unit vectors $y_k, z_k$
3:    $x := \mathbf{A} y_k z_k$;   $x' := (I - X_{k-1} X_{k-1}^T) x$
4:    if $\|x'\| < \mathtt{tol} \|x\|$ then {Breakdown}
5:      return $U = X_{k-1}$ {or repeat current iteration step with another $y_k, z_k$}
6:    else
7:      $x_k := x' / \|x'\|$;   $X_k := [X_{k-1}\ x_k]$
8:    end if
9:    Choose unit $z$
10:   for $k = 1, \dots, p_{\mathrm{pow}}$ do {Power iterations to approximate $\mathbf{A} \times_1 x_k^T \approx: \sigma y z^T$}
11:     $y := (\mathbf{A} \times_1 x_k^T)\ z = \mathbf{A} \times_1 x_k^T \times_3 z^T = \mathbf{A} z x_k$,   $\sigma := \|y\|$,   $y := y / \|y\|$
12:     $z := (\mathbf{A} \times_1 x_k^T)^T y = \mathbf{A} \times_1 x_k^T \times_2 y^T = \mathbf{A} x_k y$,   $\sigma := \|z\|$,   $z := z / \|z\|$
13:   end for
14:   err $:= \sigma$,   $\mathrm{nrm}^2 := \mathrm{nrm}^2 + \mathrm{err}^2$
15: until err $\leqslant \varepsilon\,$nrm or $k = r_{\max}$
16: return $U = X_k$
```

Dominant mode-2 and mode-3 bases V and W can be computed by the same algorithm after obvious permutation of modes. Directly from Statement 5 in matrix case we derive the following lemma on convergence of tensor method.

Lemma 1. For tensor $\mathbf{A}$ with mode ranks r_1, r_2, r_3 Alg. 7 applied to the unfoldings computes bases U, V, W that allow exact representation $\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ in Tucker form with core $\mathbf{G}$ given by (1), providing that computations of U, V, W are not terminated by breakdowns.

4.2. Selecting leading vector

Considering the choice of leading vector $y_k \otimes z_k$ we can propose four strategies that lead to different maximization problems and result in different complexity estimates and convergence properties.

4.2.1. SVD-like strategy (Wsvd). We can apply Alg. 7 to unfoldings $A^{(1)}, A^{(2)}, A^{(3)}$ and compute U, V, W in three completely independent processes, even using three processors on distributed memory system. In algorithm for $U = X_k$ the best way to keep from breakdowns is to choose $y_k \otimes z_k$ that maximizes norm of orthogonal component $\|x'\|$ on current iteration, that reads

$$\begin{aligned} y_k, z_k &= \arg \max_{\|y\|=\|z\|=1} \left\|(I - X_{k-1} X_{k-1}^T)(\mathbf{A} y z)\right\| = \arg \max_{\|y\|=\|z\|=1} \|\mathbf{B} y z\|, \\ \mathbf{B} &= \mathbf{A} \times_1 (I - X_{k-1} X_{k-1}^T). \end{aligned} \tag{15}$$

This is direct analogy with SVD approach (12) in matrix case. The main difference is that vectors x_k and $y_k \otimes z_k$ are not singular vectors of unfolding $A = A^{(1)}$ anymore, since maximization is done w.r.t. 'long' right vectors of special tensor product structure. This is the reason why best tensor rank-(r, r, r) approximation can not be computed by simultaneous elimination of best rank-$(1, 1, 1)$ approximations from residual.

However, with this choice we are safe from breakdowns. Zero-valued orthogonal component $\|x'\|$ appears only if $\max_{\|y\|=\|z\|=1} \|\mathbf{B}yz\| = 0$ and hence $\mathbf{B} = 0$. This means that *exact* representation $\tilde{\mathbf{A}}_k \overset{\text{def}}{=} \mathbf{A} \times_1 (X_k X_k^T)$ with mode-1 rank $r_1 = k$ is computed for $\mathbf{A}$. The following theorem show that 'machine precision breakdown', regulated by tolerance `tol`, also can happen only when approximation $\tilde{\mathbf{A}}_k$ is accurate up to machine precision `tol`.

Theorem 4. If on step $k + 1$ of Alg. 7 we meet the breakdown, i.e. $\|x'\| < \mathtt{tol}\|x\|$ satisfies, than computed subspace X_k provides Tucker approximation $\tilde{\mathbf{A}}_k = \mathbf{A} \times_1 (X_k X_k^T)$ with mode-1 rank $r_1 = k$ and accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}_k\|_2 < \mathtt{tol}\|\mathbf{A}\|_2$ in spectral norm.

Proof. On step $k + 1$ we choose $y_{k+1}, z_{k+1} = \arg\max_{\|y\|=\|z\|=1} \|\mathbf{B}yz\|$. Then norm of orthogonal component $x' = \mathbf{B}y_{k+1}z_{k+1}$ reads

$$\|x'\| = \max_{\|y\|=\|z\|=1} \|\mathbf{B}yz\| = \max_{\|y\|=\|z\|=1} \max_{\|x\|=1} (\mathbf{B}yz, x) = \max_{\|x\|=\|y\|=\|z\|=1} \langle \mathbf{B},\, x \otimes y \otimes z\rangle \overset{\text{def}}{=} \|\mathbf{B}\|_2.$$

Also for $x = \mathbf{A}y_{k+1}z_{k+1}$ it holds

$$\|x\| = \|\mathbf{A}y_{k+1}z_{k+1}\| = \max_{\|x\|=1} (\mathbf{A}y_{k+1}z_{k+1}, x) \leqslant \max_{\|x\|=\|y\|=\|z\|=1} \langle \mathbf{A},\, x \otimes y \otimes z\rangle \overset{\text{def}}{=} \|\mathbf{A}\|_2.$$

Now breakdown criteria $\|x'\| < \mathtt{tol}\|x\|$ gives $\|\mathbf{B}\|_2 < \mathtt{tol}\|\mathbf{A}\|_2$. Finally we write

$$\mathbf{B} = \mathbf{A} \times_1 (I - X_k X_k^T) = \mathbf{A} - \mathbf{A} \times_1 (X_k X_k^T) = \mathbf{A} - \tilde{\mathbf{A}}_k,$$

that completes the proof since mode-1 rank of $\tilde{A}_k$ is equal to $r_1 = \operatorname{rank} X_k = k$. □

Corollary 5. Alg. 7 with SVD-like strategy of choice of leading vectors (15) applied to unfoldings of tensor $\mathbf{A}$ with mode ranks r_1, r_2, r_3 computes bases U, V, W that allow exact representation $\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ after r_1, r_2 and r_3 iterations, respectively.

Note that since x_k is orthogonal to X_{k-1}, it holds

$$\mathbf{A} \times_1 x_k^T = \mathbf{A} \times_1 \left((I - X_{k-1}X_{k-1}^T)x_k\right)^T = \left(\mathbf{A} \times_1 (I - X_{k-1}X_{k-1}^T)\right) \times_1 x_k^T = \mathbf{B} \times_1 x_k^T.$$

The maximization problem (15) can be solved by ALS Alg. 2 applied for $\mathbf{B}$. In this case power iterations for matrix $\mathbf{A} \times_1 x_k^T$ on Step 9-13 of Alg. 7 can be omitted, since the error estimate $\mathrm{err} = \|\mathbf{A} \times_1 x_k^T\|_2 = \|\mathbf{B} \times_1 x_k^T\|_2$ is actually computed in ALS iterations.

On step k of Wedderburn method ALS costs $3p_{\mathrm{als}}$ tenvecs and $\mathcal{O}(p_{\mathrm{als}}nk)$ operations for orthogonalization. This summarizes to $3p_{\mathrm{als}}r$ tenvecs and $\mathcal{O}(p_{\mathrm{als}}nr^2)$ additional operations for approximation of one dominant subspace.

4.2.2. Lanczos-like strategy (Wlnc). We can also use analogy with Lanczos choice (13) by taking unit $y_k \otimes z_k \approx \mathbf{A} \times_1 x_k^T / \|\mathbf{A} \times_1 x_k^T\|$. This leads to dual maximization problem

$$\begin{aligned} y_k, z_k &= \arg \max_{\|y\|=\|z\|=1} \left\|\mathbf{A} \times_1 x_k^T \times_2 y^T \times_3 z^T\right\| = \arg \max_{\|y\|=\|z\|=1} \left\|y^T B z\right\| \\ B &= \mathbf{A} \times_1 x_k^T. \end{aligned} \tag{16}$$

The solution can be accomplished by p_{pow} steps of power iteration method applied to matrix B. On step k of Wedderburn method that requires $2p_{\text{pow}}$ tenvecs. Note that maximization problem (16) is actually solved on Steps 9-13 of Alg. 7 by power iteration that computes best rank-1 approximation of $\mathbf{A} \times_1 x_k^T$ to estimate norm of residual. Thus, after power iterations we can simply choose $y_k := y$ and $z_k := z$ as new vectors in Wedderburn elimination with Lanczos-like pivoting strategy.

4.2.3. Restricted SVD-like strategy (WsvdR). Another strategy is to run three Wedderburn elimination algorithms to extend all mode bases *simultaneously*. Suppose $k-1, l$ and m steps were done to compute mode subspaces X_{k-1}, Y_l, Z_m. Then on step k of Wedderburn process for mode-1 subspace we can make use of $V = Y_l$ and $W = Z_m$ by *restricting* maximization (15) to tensor product of these subspaces. Therefore, we take $y_k = Y_l \hat{y}_k, z_k = Z_m \hat{z}_k$ and solve

$$\begin{aligned} \hat{y}_k, \hat{z}_k &= \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \left\|(I - X_{k-1}X_{k-1}^T)\,(\mathbf{A}(Y_l\hat{y})(Z_m\hat{z}))\right\| = \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \|\mathbf{B}\hat{y}\hat{z}\| \\ \mathbf{B} &= \mathbf{A} \times_1 (I - X_{k-1}X_{k-1}^T) \times_2 Y_l^T \times_3 Z_m^T. \end{aligned} \tag{17}$$

The resulted method exactly matches optimized minimal Krylov recursion Alg. 3. We conclude that optimized MKR is a variant of Wedderburn process for tensors with restricted SVD-like strategy of pivoting. This approach can lead to slow convergence or breakdowns at first steps, but when X_k, Y_l, Z_m are large, chances to meet breakdown vanish, since (17) becomes very close to unrestricted maximization (15). The numerical results (see section 5.2) show that sometimes restricted SVD strategy leads even to better results than unrestricted SVD strategy, probably because with imposed restrictions ALS iterations are not caught in local minima.

Complexity on step k for mode-1 subspace is $3p_{\text{als}}$ tenvecs and $\mathcal{O}(p_{\text{als}} nk)$ operations for orthogonalization, total complexity is $9p_{\text{als}} r$ tenvecs and $\mathcal{O}(p_{\text{als}} n r^2)$ additional operations.

4.2.4. Restricted Lanczos-like strategy (WlncR). Finally, we can combine Lanczos-like selection of leading vectors (16) and restricted maximization (17) and solve

$$\begin{aligned} \hat{y}_k, \hat{z}_k &= \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \left\|(Y_l\hat{y})^T(\mathbf{A} \times_1 x_k^T)(Z_m\hat{z})\right\| = \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \left\|\hat{y}^T B \hat{z}\right\| \\ B &= \mathbf{A} \times_1 x_k^T \times_2 Y_l^T \times_3 Z_m^T, \qquad y_k = Y_l \hat{y}_k, \quad z_k = Z_m \hat{z}_k. \end{aligned} \tag{18}$$

If we need only bases for approximation, the maximization can be accomplished by p_{pow} steps of power iteration method applied to matrix B, and at every step of Wedderburn

Algorithm 8: Wedderburn with restricted Lanczos-like pivoting strategy (WlncR)

```
Input: Tenvec subroutine for tensor $\mathbf{A}$, tolerance $\mathtt{tol}$, accuracy $\varepsilon$
Output: Approximation $\tilde{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ with accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F \lesssim \|\tilde{\mathbf{A}}\|_F$
Initialization: Unit vectors $u, v, w$, $updX = updY = updZ = $ true.
 1: $x_1 = \mathbf{A}vw/\|\mathbf{A}vw\|$;   $y_1 = \mathbf{A}wu/\|\mathbf{A}wu\|$;   $z_1 = \mathbf{A}uv/\|\mathbf{A}uv\|$,   $k = l = m = 1$
 2: $X_1 = [x_1], Y_1 = [y_1], Z_1 = [z_1]$,   $\mathbf{G} = \mathbf{A} \times_1 x_1^T \times_2 y_1^T \times_3 z_1^T$;   $\mathtt{nrm} = \|\mathbf{G}\|_F$
 3: while $updX$ or $updY$ or $updZ$ do
 x:   {Proceed with new vector $x$ if required}
 4:   if $updU$ then
 5:     For $B = \mathbf{G}(k, :, :)$ solve $B \approx$: $\tilde{B} = b\hat{y}_k\hat{z}_k^T$ {best rank-one approximation}
 6:     $y_k := Y_l\hat{y}_k$,   $z_k := Z_m\hat{z}_k$;   $x := \mathbf{A}y_kz_k$;   $x' := (I - X_kX_k^T)x$
 7:     if $\|x'\| < \mathtt{tol}\|x\|$ then {Breakdown}
 8:       $updX :=$ false {or repeat current iteration step with another $y_k, z_k$}
 9:     else
10:       $x_{k+1} := x'/\|x'\|$;   $X_{k+1} := [X_k \, x_{k+1}]$
11:       Enlarge $\mathbf{G}$ by $\mathbf{G}(k+1, :, :) := \mathbf{A} \times_1 x_{k+1}^T \times_2 Y_l^T \times_3 Z_m^T$
12:       $\mathtt{err} := \|\mathbf{G}(k+1, :, :)\|_F$,   $\mathtt{nrm}^2 := \mathtt{nrm}^2 + \mathtt{err}^2$,   $k := k+1$
13:       if $\mathtt{err} < \varepsilon\,\mathtt{nrm}$ then {Convergence}
14:         $updX :=$ false
15:       end if
16:     end if
17:   end if
 y:   {Proceed with new vector $y$ if required}
 z:   {Proceed with new vector $z$ if required}
18: end while
19: return  $U = X_k, V = Y_l, W = Z_m$,   $\tilde{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$
```

method that requires $2p_{\text{pow}}$ tenvecs and some operations for orthogonalization. But if approximation with core is desired, the complexity can be highly improved by precomputing B as $l \times m$ matrix. By comparing (18) and (1) we note that B is exactly last mode-1 slice from optimal core for approximation of $\mathbf{A}$ in bases $U = X_k, V = Y_l, W = Z_m$

$$\mathbf{G}(k, :, :) = \mathbf{A} \times_1 (U(:, k)) \times_2 V^T \times_3 W^T = \mathbf{A} \times_1 x_k^T \times_2 Y_l^T \times_3 Z_m^T = B.$$

Thus if we need computation of $\mathbf{G}$, we prefer to do it slice-by-slice in Wedderburn process. Then we can apply standard matrix tools like SVD to find best rank-one approximation of precomputed B to solve (18) without any additional tensor operations. This version of Wedderburn elimination algorithm for tensors is summarized by Alg. 8.

4.3. Comparison of algorithms

To compare the versions of the proposed algorithm, we give their complexities in Table 1. If only subspaces U, V, W for Tucker rank-(r_1, r_2, r_3) approximation are required, than minimal Krylov recursion [15] is fastest in terms of number of tenvecs used, since it uses

Table 1. Complexity of algorithms for rank-(r_1, r_2, r_3) tensor approximation

name	description	output	tenvecs
MKR	Min. Krylov recursion [15] Alg. 1	U, V, W	$3r$
Wsvd	Alg. 7 with strategy (15)	U, V, W	$9p_{\mathrm{als}}r + 3r$
Wlcz	Alg. 7 with strategy (16)	U, V, W	$6p_{\mathrm{pow}}r + 3r$
WsvdR	Alg. 7 with strategy (17), Alg. 3	U, V, W	$9p_{\mathrm{als}}r + 3r$
WlczR	Alg. 7 with strategy (18), Alg. 8	U, V, W	$6p_{\mathrm{pow}}r + 3r$
WlczR	— // —	$U, V, W, \mathbf{G}$	$r^2 + 3r$

only $3r$ tensor operations. All versions of Wedderburn elimination Alg. 7 also require *linear in* r number of tenvecs, with factor depending only on selected number of ALS or power iterations. Note that for $p_{\mathrm{als}} = p_{\mathrm{pow}}$, Lanczos-like pivoting strategy and SVD-like pivoting take roughly the same time, but SVD-like pivoting is guaranteed to be free from breakdowns. This differ tensor algorithms from the matrix case.

If core $\mathbf{G}$ for Tucker approximation is also required, we generally need additional r^2 tenvecs to evaluate it by (1). In this case, the fast version of Wedderburn elimination algorithm is one with restricted Lanczos-like pivoting given by Alg. 8. It uses *exactly* the same number of tenvecs as MKR, but shows much better convergence in numerical experiments.

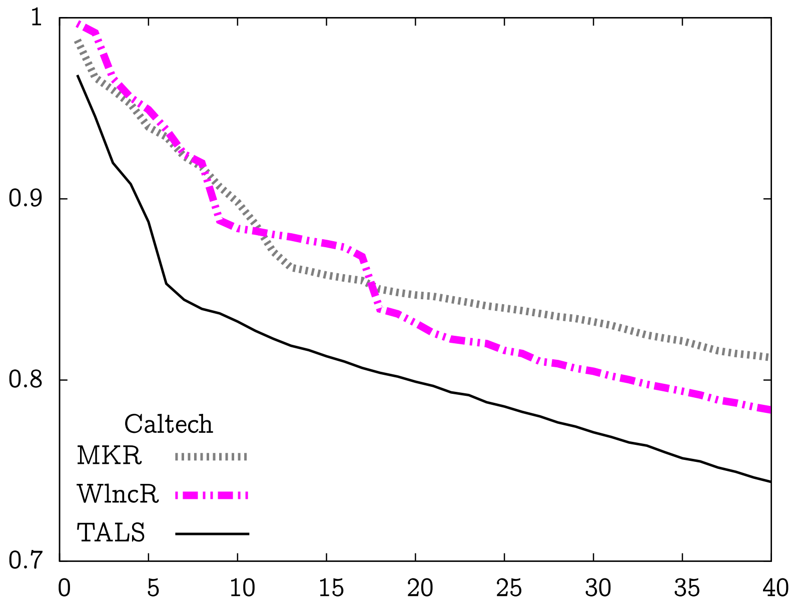
5. Numerical examples

The numerical experiments presented in this section were performed on Intel Xeon Quad-Core E5504 CPU running at 2.00 GHz. In section 5.1 we use MATLAB version 7.7.0, in sections 5.2, 5.3 use Intel Fortran compiler version 11.1 and BLAS/LAPACK routines provided by MKL library.

5.1. Sparse tensors

The tensor decomposition of sparse large-dimensional arrays is an important tool in network analysis, that is widely used now in information science, sociology, and many other disciplines. We consider one example from [27], where a very nice introduction to network analysis is given. The example is a graph of Facebook social network from Caltech University, coupled with dormitory information. The graph represent relations between n people living in d dorms, with $\mathbf{A}(i, j, p, q) = 1$ meaning that person number i links to person number j, and p, q are dorms numbers of persons i and j. Other elements of $\mathbf{A}$ are zeroes. In our example $n = 597$, $p = 8$ and $\mathbf{A}$ is considered as $597 \times 597 \times 8^2$ tensor with only 25646 nonzeros. The accuracy of approximation is shown on Fig. 1. We compare accuracy of approximation given by Tucker-ALS [8, 9] (we use MATLAB implementation from Tensor Toolbox [28]) with accuracy approximations given by of

Figure 1. Accuracy of approximation of Caltech graph, $597 \times 597 \times 8^2$



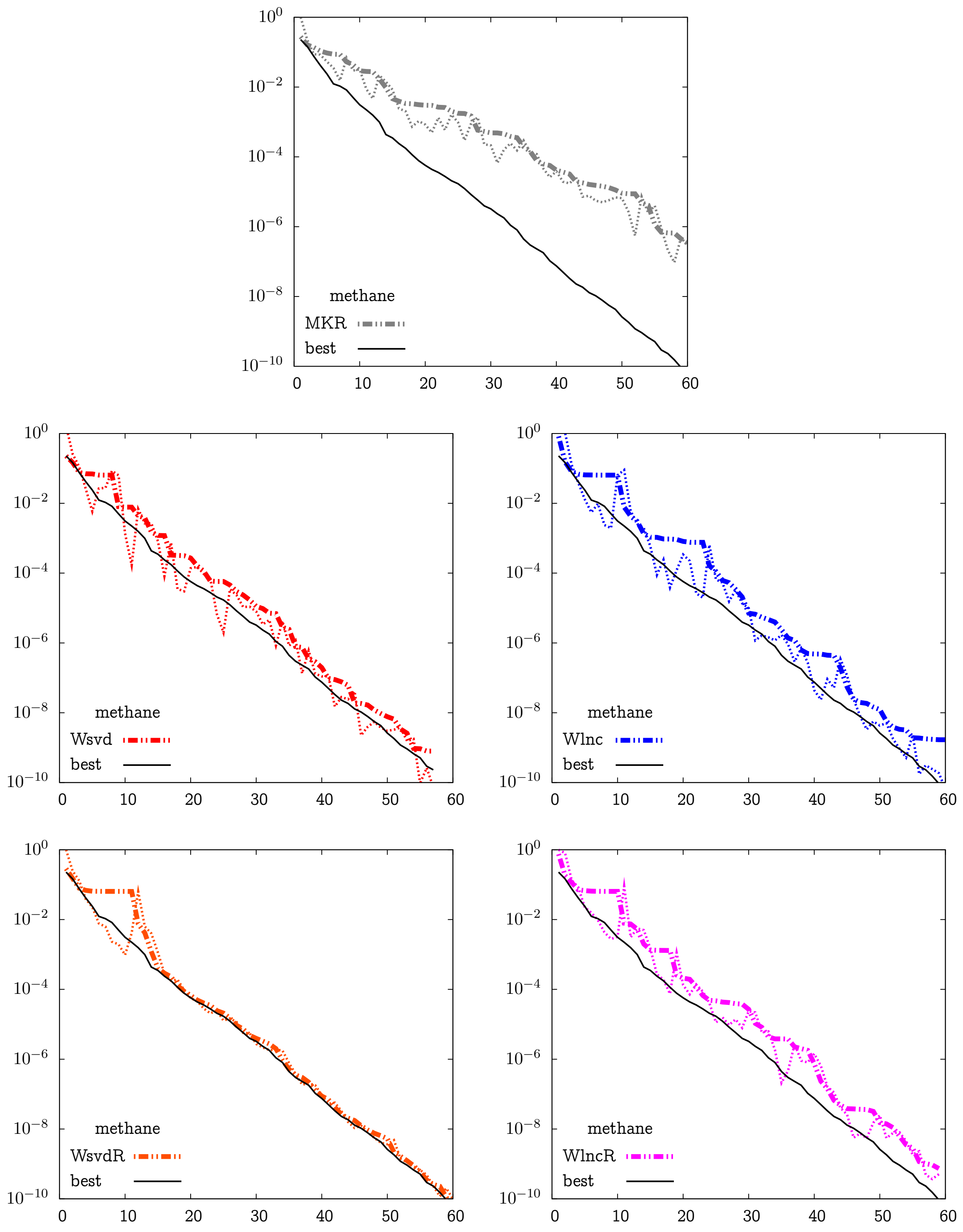
minimal Krylov recursion Alg. 1 and Wedderburn elimination method with Lanczos-like restricted pivoting Alg. 8. We note that WlncR Alg. 1 converges slowly on the first iterations, when accumulated bases are small and this imposes serious restrictions in maximization (18). However, on the next steps of iterations, WlncR become more accurate than MKR method. Tucker-ALS algorithm gives most accurate results, but requires $3r^2$ tenvecs on each iteration, that is much more than the complexity of MKR and WlncR. We do not provide timings here, because MATLAB-based computations are usually far from being highly optimized. This will be done in the next sections for Fortran implementation of discussed algorithms.

5.2. Compression from canonical to Tucker format

Multidimensional data often appear in modern modelling programs in canonical form (C). For example, in chemical modelling programs, e.g. PC GAMESS, MOLPRO, the *electron density function* is given as a sum of tensor product of Gaussians. However, even for simple molecules, number of terms in decomposition obtained by MOLPRO may be too large for practically feasible computations. In order to make computations efficient, further approximation (recompression) to the Tucker format can be performed. This was done in [29] using Tucker-ALS algorithm [8, 9], in [30] by Tucker-ALS with initial guess obtained from the coarser grids, in [31] by Cross3D algorithm [14], in [20] by individual cross approximation of canonical factors and in [32] by cross approximation of Gram matrices of unfoldings. For an $n \times n \times n$ tensor given in canonical form with R terms, each tenvec costs $3nR$ operations, thus proposed versions of Alg. 7 can be applied to compute its Tucker approximation, even for large n and R.

We apply the discussed algorithm for Tucker approximation of electron density of

Figure 2. Accuracy of approximation of methane electron density, $n = 5121$



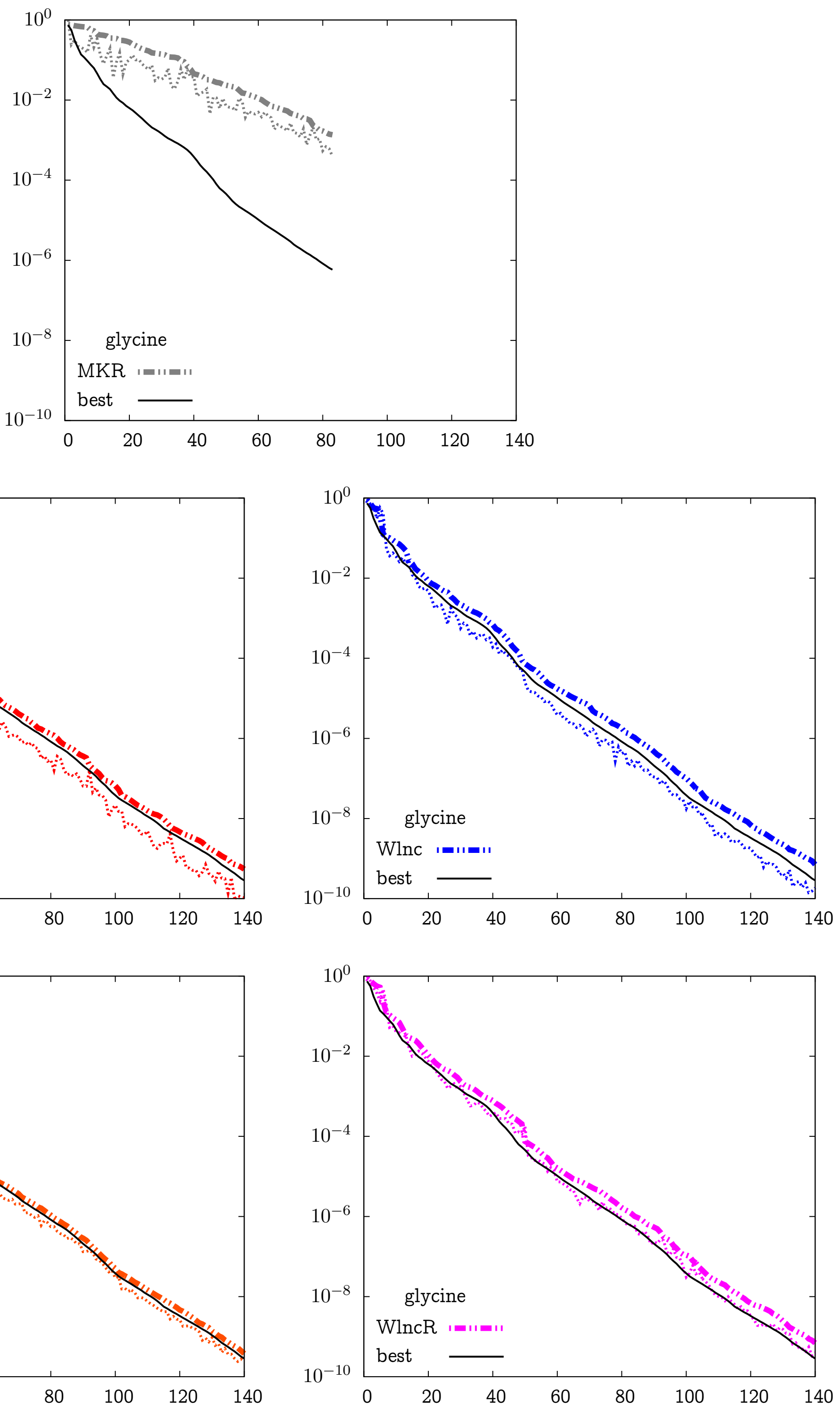


Figure 3. Accuracy of approximation of glycine electron density, $n = 5121$

Table 2. Time to approximate the electron density, seconds, $n = 5121$

molecule	accuracy	MKR	Wsvd	Wlnc	WsvdR	WlncR	cross	TALS(1)
methane	10^{-4}	1.0	6.1	4.4	2.0	0.5	1.0	2.6
$R = 1334$	10^{-6}	1.7	10.7	8.2	3.7	0.8	1.4	9.6
	10^{-8}	—	14.5	11.1	5.1	1.4	1.9	30
	10^{-10}	—	20.4	16.1	6.9	2.0	2.9	59
ethane	10^{-4}	2.7	17	15	6.7	1.6	3.0	4.6
$R = 3744$	10^{-6}	5.2	32	25	12	2.8	4.9	17
	10^{-8}	—	45	35	17	3.9	6.3	42
	10^{-10}	—	61	50	23	5.3	8.2	83
ethanol	10^{-4}	8.0	54	45	17	4.7	8.1	22
$R = 6945$	10^{-6}	14	89	74	30	8.4	13	81
	10^{-8}	—	135	108	45	13	17	194
	10^{-10}	—	180	145	61	18	22	391
glycine	10^{-4}	—	85	69	37	7.5	24	32
$R = 9208$	10^{-6}	—	131	112	57	13	33	96
	10^{-8}	—	200	160	80	18	43	211
	10^{-10}	—	268	211	114	24	60	412

some simple molecules, discretized on uniform tensor $n \times n \times n$ grid with $n = 5121$. The convergence of algorithms, i.e. accuracy of rank-(r, r, r) approximation for different r is shown on Figs. 2, 3. On each graph we compare accuracy of best approximation computed by Tucker-ALS [8, 9] (thin solid line) with accuracy of certain approximation method. For each method, the internal estimate of error $\mathtt{err}$ is shown by thin dashed line and real accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ of algorithm is shown by thick dashed line. The core of Tucker approximation was chosen by (1). Since evaluation of whole $n \times n \times n$ array for $n = 5121$ requires one terabyte of memory and a lot of computational resources, we verify the accuracy of algorithms by comparison of result with Tucker approximation computed by individual cross approximation of canonical factors proposed in [20] with accuracy set to $\varepsilon = 10^{-12}$. The verification of cross method was done in [20] by exhaustive computation of residual on parallel memory platforms, and thus Tucker approximation computed by cross method is considered as reliable answer. The residual between two Tucker formats is computed as proposed in [33].

We note slow convergence of minimal Krylov recursion for methane electron density and breakdown for glycine density. All versions of Wedderburn elimination converge much better, and for larger glycine molecule the convergence is even more regular, than for methane. Accuracy of methods with Lanczos-like pivoting is close to the optimal one, and accuracy of method with restricted SVD-like pivoting is almost equal to optimal, except for first steps of the process, when accumulated subspaces are small and impose significant restrictions on selection of leading vectors. Also, the internal error value, estimated by norm of $\mathbf{A} \times_1 x_k^T$ (and the same for other modes), is less regular than real accuracy, that decays monotonically. In methods with unrestricted pivoting (see graph for Wlnc) the internal error value is computed by spectral norm of matrix $\mathbf{A} \times_1 x_k^T$ and appears to be 'more optimistic' than real error, that is measured in Frobenius norm. In methods with restricted pivoting the Frobenius norm of $\mathbf{A} \times_1 x_{k+1}^T \times_2 Y_l^T \times_3 Z_m^T$ is used to estimate the internal error, which matches the real error more closely.

Timings for approximation for different molecules and accuracy parameters are given in Tab. 2. We compare all methods listed in Tab. 1 and method proposed in [20] based on incomplete cross approximation [12] of unfoldings. For Wedderburn elimination methods we set $p_{als} = p_{pow} = 3$. For reference we also provide time for one iteration of Tucker-ALS method [8, 9], that uses output of Alg. 8 as initial guess. In our implementation of Tucker-ALS we compute dominant subspaces by cross approximation, that is usually several $(2 \div 20)$ times faster than SVD-based computations. However, even one iteration of modified Tucker-ALS seems to be quite expensive. We note that for this example the Wedderburn elimination with restricted Lanczos-like pivoting (Alg. 8) is faster than all other versions of Wedderburn elimination algorithms. We also note that it outperforms minimal Krylov recursion, since Alg. 1 converges slowly (or even does not converge) and uses more iterations to reach the same accuracy level. WlncR also outperforms method based on cross approximation of unfoldings [20]. We conclude that for this problem Alg. 8 outperforms previously proposed methods.

5.3. Recompression in operations with structured tensors

The efficient operations with matrices and vectors in compressed tensor formats is crucial in construction of efficient iterative methods for solving equations and eigenproblems in three and more dimensions. The approach to such highly-efficient tensor linear algebra subroutines was discussed in [33, 34], where it is shown that efficient evaluation of all basic linear algebra subroutines with tensor-structured data is based on fast recompression of certain structured tensor. As an example, consider Hadamard (elementwise) multiplication between $n_1 \times n_2 \times n_3$ tensors $\mathbf{A}$ and $\mathbf{B}$ given in Tucker form

$$\mathbf{A} = \mathbf{G} \times_1 U^{(A)} \times_2 V^{(A)} \times_3 W^{(A)}, \qquad \mathbf{B} = \mathbf{H} \times_1 U^{(B)} \times_2 V^{(B)} \times_3 W^{(B)}.$$

Let mode ranks of $\mathbf{A}$ be r_1, r_2, r_3 and mode ranks of $\mathbf{B}$ be p_1, p_2, p_3. The result reads

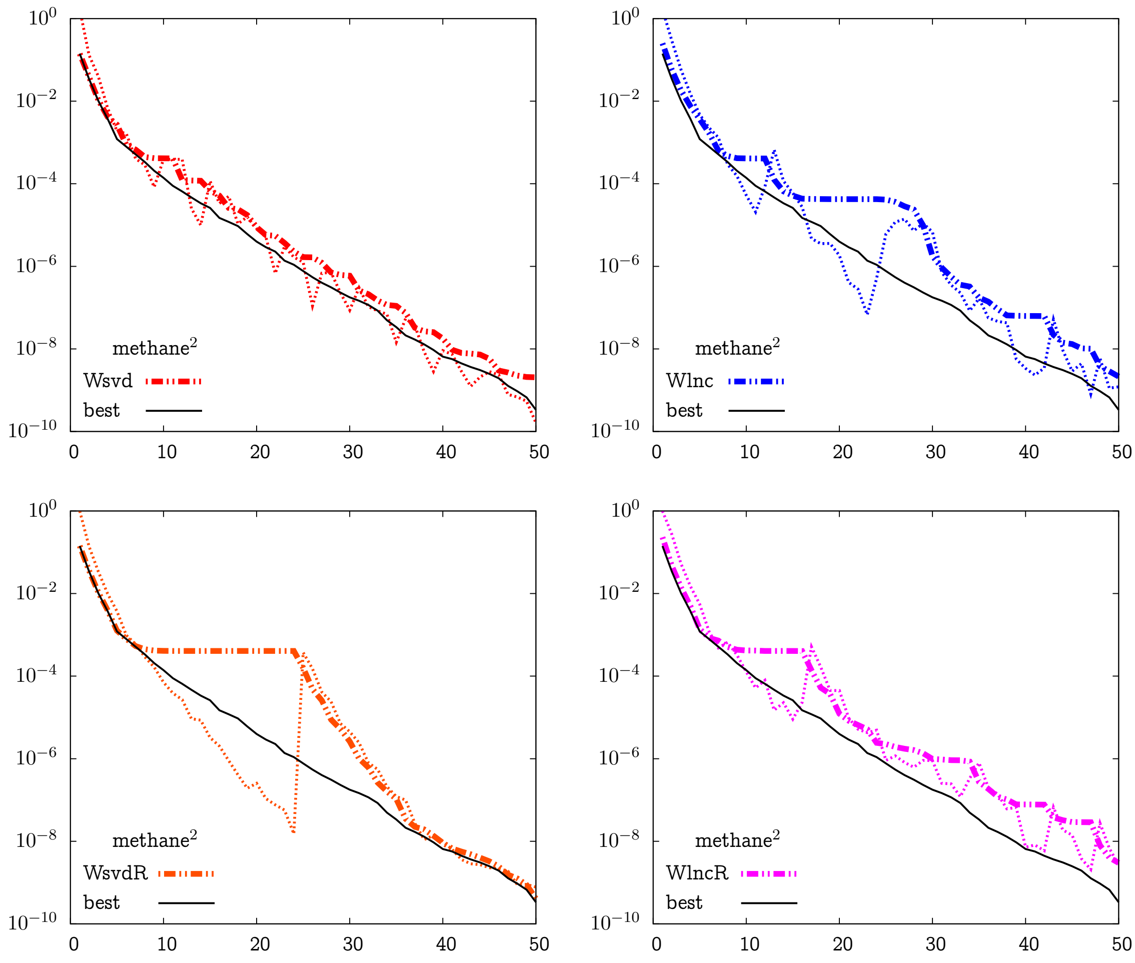
$$\mathbf{C} = \mathbf{F} \times_1 U \times_2 V \times_3 W \tag{19}$$

with $p_1 r_1 \times p_2 r_2 \times p_3 r_3$ core $\mathbf{F} \overset{\text{def}}{=} \mathbf{Kron}(\mathbf{G}, \mathbf{H})$ and *non-orthogonal* factors U, V, W of sizes $n_1 \times p_1 r_1$, $n_2 \times p_2 r_2$ and $n_3 \times p_3 r_3$, respectively. Formally this again the Tucker format with core and factors given by

$$\mathbf{F}(ap, bq, cs) \overset{\text{def}}{=} \mathbf{G}(p, q, s)\mathbf{H}(a, b, c), \qquad U(i, ap) = U^{(A)}(i, p)U^{(B)}(i, a),$$

and so on for V, W. However, mode ranks of $\mathbf{C}$ are products of correspondent mode ranks of $\mathbf{A}$ and $\mathbf{B}$, and recompression is required to reduce the storage size.

Figure 4. Accuracy of Hadamard square of methane electron density, $n = 5121$



In [34] fast method if recompression based on individual filtering of factors were proposed. Numerical examples in [34] includes evaluation of Hartree potential for electron density of molecules, discussed in section 5.2. This problem writes as multiplication between three-level matrix in canonical format (with diagonal core tensor **G**) and three-dimensional vector of electron density given in Tucker format. The convolution with approximation of result to Tucker format requires $10 \div 120$ seconds depending on complexity of molecule and desired accuracy of evaluation. However, no examples of Tucker to Tucker multiplication were presented in [34], since this operation appears to be sufficiently more expensive for individual filtering method and for complex molecules it requires up to an hour, that we consider not affordable.

We show that fast and accurate multiplication between two tensors in Tucker format can be done using Wedderburn-based Alg. 7. Each tenvec operation with tensor (19) can be done in $\mathcal{O}(q^4 + nq^2)$, where $q = \max(p, r)$, that is fast enough even for n up to hundred thousands and p, r up to several hundreds. We apply the discussed algorithm for Hadamard multiplication of discretized electron density of simple molecules to themselves. This operation can be a building block for algorithms that compute

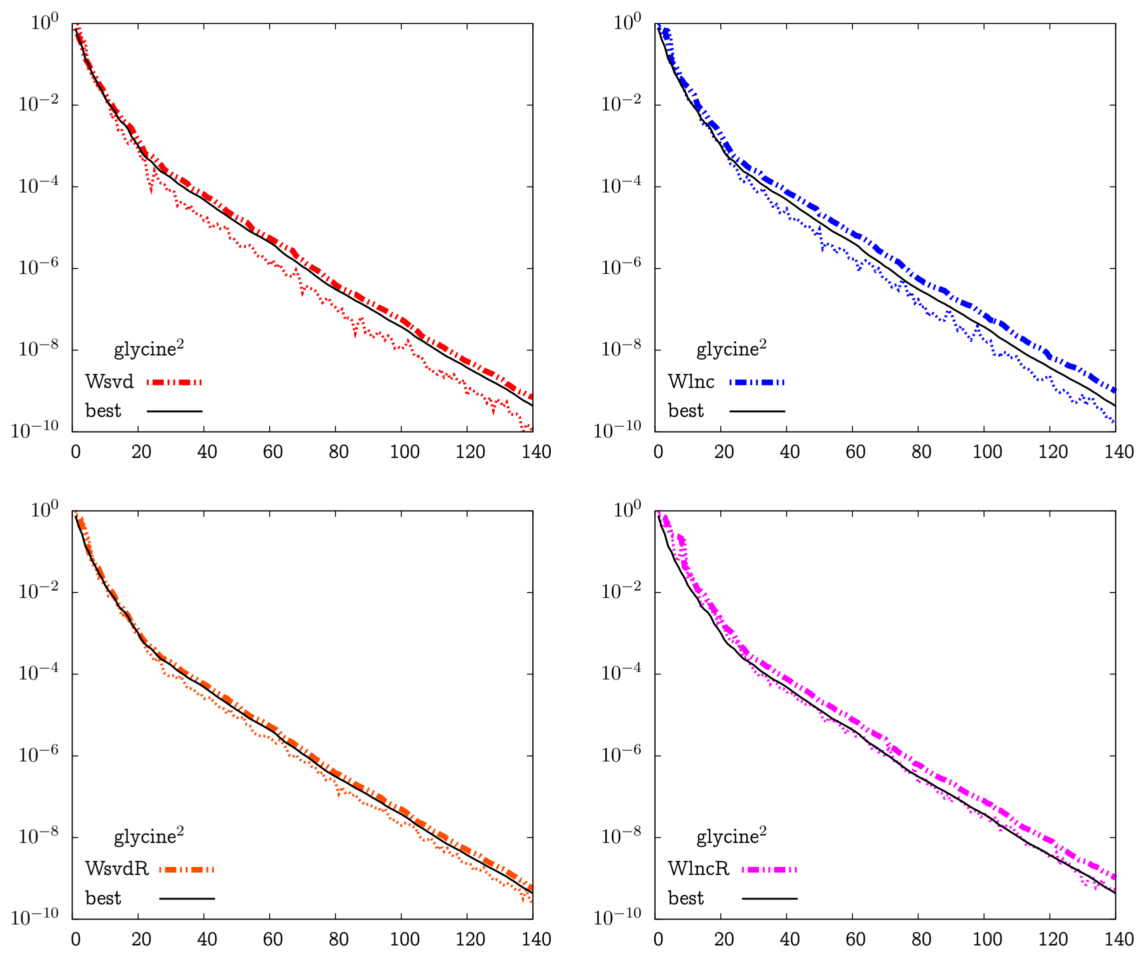


Figure 5. Accuracy of Hadamard square of glycine electron density, $n = 5121$

pointwise nonlinear functions of large tensors with complexity linear in mode size, for example cubic root of electron density, that is important in Kohn-Sham model. A good initial guess for such methods can be evaluated by mimic algorithm [20].

The convergence of algorithms, i.e. accuracy of rank-(r, r, r) approximation for different r is shown on Figs. 4, 5. On each graph we compare accuracy of best approximation computed by Tucker-ALS [8, 9] (thin solid line) with accuracy of certain approximation method. The real accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ was verified by comparing the result with Tucker approximation computed by Cross3D algorithm [14] with accuracy set to $\varepsilon = 10^{-12}$. The Cross3D algorithm was verified in [14, 31] by exhaustive check on parallel memory platforms, and can be considered as reliable answer. The residual between two Tucker formats is computed as proposed in [33].

As well as for recompression of electron density from canonical form, in this problem all versions of Wedderburn elimination demonstrate good convergence, and for larger glycine molecule it is even more regular, than for methane. Comparing the different versions of Alg. 7, we note the same behavior that is already described in section 5.2.

Timings for approximate computation of Tucker-to-Tucker multiplication for differ-

Table 3. Time to compute Hadamard square of electron density, seconds, $n = 5121$

molecule	accuracy	Wsvd	Wlnc	WsvdR	WlncR	TALS(1)
methane	10^{-4}	15.5	13.8	8.1	2.8	2.4
$(74, 74, 74)$	10^{-6}	34	33	14.8	9.7	14.3
	10^{-8}	63	46	43	18.6	42
	10^{-10}	93	74	68	40	97
ethane	10^{-4}	22	20	11.8	4.2	4.2
$(67, 94, 83)$	10^{-6}	46	41	33	14.0	16.7
	10^{-8}	82	68	72	27	45
	10^{-10}	125	105	127	56	117
ethanol	10^{-4}	120	101	106	45	52
$(128, 127, 134)$	10^{-6}	281	228	293	176	257
	10^{-8}	493	419	635	441	678
	10^{-10}	736	653	1100	808	1370
glycine	10^{-4}	179	170	177	60	64
$(62, 176, 186)$	10^{-6}	442	380	600	217	270
	10^{-8}	732	600	1033	500	646
	10^{-10}	1010	850	1530	888	1223

ent methods and accuracy parameters are given in Tab. 3. We compare all methods listed in Tab. 1 and provide time for one iteration of Tucker-ALS method [8, 9], that uses output of Alg. 8 as initial guess. For Wedderburn elimination methods we set $p_{\text{als}} = p_{\text{pow}} = 3$. Again, we note that for this example the WlncR Alg. 8 is faster than all other versions of Wedderburn elimination algorithms and also than one iteration of Tucker-ALS method. In this case the difference between versions of Wedderburn elimination algorithms is small, because time for evaluation of core (r^2 tenvecs) dominates over $\mathcal{O}(nr^2)$ time for additional operations. Nevertheless, we conclude that for this problem Alg. 8 can be method of choice for fast approximate evaluation of operations with data in tensor formats.

6. Conclusion and further work

We presented the family of algorithms for approximation of large three-dimensional tensors that are used only through tenvec (tensor-by-vector-by-vector multiplication) operation. The new approach is based on Wedderburn rank-reduction formula and admits different strategies to select vectors of Wedderburn elimination ('pivoting') that lead to algorithms with rather different convergence and complexity estimates. The fastest algorithm from presented family, namely WlncR Alg. 8, may converge slowly or stagnate on the first steps of iterations, however, one can easily propose an efficient algorithm combining slow pivoting strategy on the first steps (for example, Wsvd, that is free from breakdowns) with fast WlncR pivoting strategy on the next steps of algorithm.

The presented methods can be applied for approximation of dominant subspaces of large structured tensors. In examples discussed we show that proposed algorithms are as fast as minimal Krylov recursion [15], but more accurate in certain cases. We also show that Wedderburn-based methods are much faster than Tucker-ALS [8, 9] approximation algorithm. The proposed methods can be applied directly for fast computation of bilinear operations between structured tensors, but the efficiency can be further improved by combining the proposed methods with individual factor filtering proposed in [34].

Canonical and Tucker formats can be straightforwardly generalized to d dimensions, however each of them has serious drawbacks, and another decomposition should be used for high dimensions, for example recently introduced tensor-train (TT decomposition, see [35, 36, 37]), that are based on the same SVD techniques, but is free from the curse of dimensionality. Therefore, it is very natural to extend the proposed ideas to TT format. In the sequel we will show how to construct algorithm for TT format that use tensor through tensor-by-vectors multiplication, i.e. Krylov-type methods in d dimensions.

Acknowledgements

This work was partially supported by the RFBR grants 08-01-00115, 09-01-00565, 09-01-12058, 10-01-00757, joint RFBR/DFG grant 09-01-91332, Russian Federation Gov. contracts No. П1178 and П940, President Grant MK–127.2009.1 and Priority research program of Dep. Math. RAS No. 3 and 5. Part of this work was done during the stay of the second and third authors in Max-Plank Institute for Mathematics in Sciences in Leipzig (Germany). Authors are grateful to Heinz-Juergen Flad and Rao Chinnamsettey for providing input data for the electron density function. Authors are grateful to Lars Eldén for inspiring paper [15], heartful attention to results of this paper, presented on ICSMT (Hong Kong, January 2010) and GAMM seminar (Leipzig, February 2010) and introducing us to the field of approximation of sparse tensors in network analysis.

References

[1] *Hitchcock F. L.* The expression of a tensor or a polyadic as a sum of products // *J. Math. Phys.* 1927. V. 6, № 1. P. 164–189.

[2] *Caroll J. D., Chang J. J.* Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition // *Psychometrika*. 1970. V. 35. P. 283-319.

[3] *Harshman R. A.* Foundations of the Parafac procedure: models and conditions for an explanatory multimodal factor analysis // *UCLA Working Papers in Phonetics*. 1970. V. 16. P. 1-84.

[4] *de Silva V., Lim L.* Tensor Rank and the Ill-Posedness of the Best Low-Rank Approximation Problem // *SIAM J. Matrix Anal. Appl.* 2008. V. 30, № 3. P. 1084–1127. doi:10.1137/06066518x.

[5] *Tucker L. R.* Some mathematical notes on three-mode factor analysis // *Psychometrika*. 1966. V. 31. P. 279-311.

[6] *Kolda T. G., Bader B. W.* Tensor Decompositions and Applications // *SIAM Review*. 2009. V. 51, № 3. P. 455–500. doi:10.1137/07070111X.

[7] *de Lathauwer L., de Moor B., Vandewalle J.* A multilinear singular value decomposition // *SIAM J. Matrix Anal. Appl.* 2000. V. 21. P. 1253–1278. doi:10.1137/s0895479896305696.

[8] *Kroonenberg P., de Leeuw J.* Principal component analysis of three-mode data by means of alternating least squares algorithms // *Psychometrika*. 1980. V. 45, № 1. P. 69–97.

[9] *de Lathauwer L., de Moor B., Vandewalle J.* On best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of high-order tensors // *SIAM J. Matrix Anal. Appl.* 2000. V. 21. P. 1324-1342.

[10] *Savas B., Eldén L.* A Newton-Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor // *SIAM J. Matrix Anal. Appl.* 2009. V. 31, № 2. P. 248-271.

[11] *Ishteva M., De Lathauwer L., Absil P. A., Van Huffel S.* Differential-geometric Newton method for the best rank-(r_1, r_2, r_3) approximation of tensors // *Numerical Algorithms*. 2009. V. 51, № 2. P. 179–194.

[12] *Tyrtyshnikov E. E.* Incomplete cross approximation in the mosaic–skeleton method // *Computing*. 2000. V. 64, № 4. P. 367-380. doi:10.1007/s006070070031.

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

[14] *Oseledets I. V., Savostianov D. V., Tyrtyshnikov E. E.* Tucker dimensionality reduction of three-dimensional arrays in linear time // *SIAM J. Matrix Anal. Appl.* 2008. V. 30, № 3. P. 939-956. doi:10.1137/060655894.

[15] *Savas B., Eldén L.* Krylov subspace methods for tensor computations: Preprint LITH-MAT-R-2009-02-SE. — Linköping: Dep. Math. Linköpings Univ., 2009.

[16] *Chu M. T., Funderlic R. E., Golub G. H.* A rank-one reduction formula and its applications to matrix factorizations // *SIAM Review*. 1995. V. 37, № 4. P. 512–530.

[17] *Bouaricha A.* Tensor-Krylov methods for large nonlinear equations // *Comp. Optimization Appl.* 1996. V. 5, № 3. P. 207–232.

[18] *Defant A., Floret K.* Tensor norms and operator ideals. — North Holland, 1993.

[19] *Goreinov S., Oseledets I., Savostyanov D. et al.* How to find a good submatrix // Matrix Methods: Theory, Algorithms, Applications / Ed. by V. Olshevsky, E. Tyrtyshnikov. — World Scientific Publishing, 2010. — P. 247-256.

[20] *Oseledets I. V., Savostyanov D. V., Tyrtyshnikov E. E.* Cross approximation in tensor electron density computations // *Numer. Lin. Alg. Appl.* 2009. doi:10.1002/nla.682.

[21] *Zhang T., Golub G. H.* Rank-one approximation to high-order tensors // *SIAM J. Matrix Anal. Appl.* 2001. V. 23. P. 534-550.

[22] *Wedderburn J. H. M.* Lectures on matrices, colloquim publications. — New York: AMS, 1934. — V. XVII.

[23] *Golub G., Kahan W.* Calculating the singular values and pseudo-inverse of a matrix // *SIAM J. Numer. Anal.* 1965. V. 2, № 2. P. 205–224.

[24] *Paige C. C.* Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix // *IMA J. Appl. Math.* 1976. V. 18, № 3. P. 341.

[25] *Golub G., Van Loan C.* Matrix computations. — Johns Hopkins University Press, 1996.

[26] *Greenbaum A.* Iterative methods for solving linear systems. — Society for Industrial Mathematics, 1997.

[27] *Traud A. L., Kelsic E. D., Mucha P. J., Porter M. A.* Community Structure in Online Collegiate Social Networks: Tech. rep.: 2008. — arXiv:0809.0960.

[28] *Bader B. W., Kolda T. G.* Efficient MATLAB computations with sparse and factored tensors // *SIAM J. Sci. Comp.* 2007. V. 30, № 1. P. 205–231. doi:10.1137/060676489.

[29] *Chinnamsetty S. R., Flad H.-J., Khoromskaia V., Khoromskij B. N.* Tensor decomposition in electronic structure calculations on 3D Cartesian grids // *J. Comp. Phys.* 2009. V. 228, № 16. P. 5749-5762. doi:10.1016/j.jcp.2009.04.043.

[30] *Khoromskij B. N., Khoromskaia V.* Multigrid accelerated tensor approximation of function related multidimensional arrays // *SIAM J. Sci. Comp.* 2009. V. 31, № 4. P. 3002-3026. doi:10.1137/080730408.

[31] *Flad H.-J., Khoromskij B. N., Savostyanov D. V., Tyrtyshnikov E. E.* Verification of the cross 3D algorithm on quantum chemistry data // *Rus. J. Numer. Anal. Math. Model.* 2008. V. 23, № 4. P. 329-344. doi:10.1515/RJNAMM.2008.020.

[32] *Savostyanov D. V.* Fast revealing of mode ranks of tensor in canonical formal // *Numer. Math. Theor. Meth. Appl.* 2009. V. 2, № 4. P. 439-444. doi:10.4208/nmtma.2009.m9006s.

[33] *Oseledets I. V., Savostyanov D. V., Tyrtyshnikov E. E.* Linear algebra for tensor problems // *Computing*. 2009. V. 85, № 3. P. 169-188. doi:10.1007/s00607-009-0047-6.

[34] *Savostyanov D. V., Tyrtyshnikov E. E.* Approximate multiplication of tensor matrices based on the individual filtering of factors // *J. Comp. Math. Math. Phys.* 2009. V. 49, № 10. P. 1662-1677. doi:10.1134/s0965542509100029.

[35] *Oseledets I. V., Tyrtyshnikov E. E.* Breaking the curse of dimensionality, or how to use SVD in many dimensions // *SIAM J. Sci. Comp.* 2009. V. 31, № 5. P. 3744-3759.

[36] *Oseledets I. V.* Compact matrix form of the d-dimensional tensor decomposition: Preprint 2009-01. — Moscow: INM RAS, 2009. http://pub.inm.ras.ru.

[37] *Oseledets I. V.* Approximation of $2^{\mathrm{d}} \times 2^{\mathrm{d}}$ matrices using tensor decomposition // *SIAM J. Matrix Anal. Appl.* 2010.