Cross approximation in electron density computations*

I. V. Oseledets[†], D. V. Savostyanov[‡], E. E. Tyrtyshnikov[§]

February 2, 2009

Abstract

We propose new tensor approximation algorithms for certain discrete functions related with the Hartree-Fock equation. Given a canonical tensor representation for the electron density function (for example, produced by some packages such as MOLPRO), we obtain its Tucker approximation with much fewer parameters than the input data and Tucker approximation for the cubic root of this function, which is part of the Kohn-Sham exchange operator. The key idea is in the fast and accurate prefiltering of possibly large-scale factors of the canonical tensor input. The new algorithms are based on the incomplete cross approximation method applied to matrices and tensors in higher dimensions and outperform other tools for the same purpose.

First, we show that the cross approximation method is robust and much faster than the SVD-based approach. As a consequence, it becomes possible to increase the resolution of grid and the complexity of molecules that can be handled by the Hartree-Fock chemical models. Second, we propose a fast mimic approximation method for $f^{1/3}(x, y, z)$, based on the factor prefiltering method for $f(x, y, z)$. Third, we conclude that the Tucker format has advantages in storage and computation time compared to the ubiquitous canonical format.

Keywords: Hartree-Fock/Kohn Sham, electron density, canonical decomposition, Tucker approximation, cross approximation.

1 Introduction

For large-scale problems in three and higher dimensions, approximation of functions and operators in a suitable structured format with a reduced number of involved parameters is merely mandatory. In this respect, tensor formats have proved to be most useful. Tensors as a new computational paradigm with nice general approximation properties

*This work was partially supported by RFBR grant 08-01-00115 and RFBR/DFG grant 09-01-91332.

[†]Institute of Numerical Mathematics RAS, 119333 Moscow, Gubkina 8; ivan.oseledets@gmail.com.

[‡]Institute of Numerical Mathematics RAS, 119333 Moscow, Gubkina 8; drraug@gmail.com.

[§]Institute of Numerical Mathematics RAS, 119333 Moscow, Gubkina 8; tee@inm.ras.ru were presented in [16, 17]. A series of recent papers is devoted to fast tensor algorithms and applications in computational physics [2, 12, 13], and in particular in electronic structure calculations based on the Hartree-Fock/Kohn-Sham equations.

In a certain extent, tensors are already incorporated in modern chemical modelling programs, e.g. GAMESS, MOLPRO. In particular, the *electron density function* $f(x, y, z)$ in the MOLPRO output appears as a sum of gaussians

$$f(x, y, z) = \sum_{i=1}^{m} \sum_{j=1}^{m} \sigma_{ij} \Phi_{ij}^{(1)}(x) \Phi_{ij}^{(2)}(y) \Phi_{ij}^{(3)}(z).$$

The right-hand side is the sum of $R = m(m+1)/2$ functions (since $\Phi_{ij}^{(\mu)}(t) = \Phi_{ji}^{(\mu)}(t)$) with separated variables. Upon discretization on a tensor grid $\{x_i\} \times \{y_j\} \times \{z_k\}$, we obtain a data array (*tensor*) that inherits the same structure. This type of structure is known as *canonical decomposition* of a tensor. The number of summands is referred to as *canonical rank* of the canonical decomposition. However, even for simple molecules, canonical ranks obtained by MOLPRO are too large for practically feasible evaluation of the Hartree potential. In order to make computations effective and efficient, we propose to perform further approximation (recompression) using the Tucker format [14].

We are interested as well in a low-parametric approximation of some functions of $f(x, y, z)$. Here we focus on the cubic root $f^{1/3}(x, y, z)$ which is used in the Kohn-Sham model for approximation of the nonlocal exchange potential. For such functions the canonical input for $f(x, y, z)$ cannot be handled in a direct way. Nevertheless, we propose a fast *mimic tensor approximation* method with the Tucker format output that provides a significant reduction of the input data.

2 Basic definitions

Throughout the paper, a *tensor* means an array with many indices. The indices are also called *modes*, *directions* or *axes*. The number of indices is referred to as *dimension* of the given tensor.

The number of allowed values for a mode index is called the *mode size*. If every mode is of size n, then the total number of entries of a d-dimensional tensor is n^d. Hence, in numerical work with tensors it is crucial to look for some data reduction structures rather than full arrays. Most useful are the following.

The *canonical format* of a tensor $\mathbf{F} = [f_{ij\ldots k}]$ reads

$$\mathbf{F} = (A, B, \ldots, C), \qquad f_{ij\ldots k} = \sum_{t=1}^{R} a_{it} b_{jt} \ldots c_{kt}.$$

It is also referred to as *canonical decomposition* or *multilinear form/decomposition*. If the right-hand side does not give $\mathbf{F}$ exactly, it is considered as *canonical approximation* of $\mathbf{F}$. The number of summands is called *tensor rank* or *canonical rank* of the corresponding canonical decomposition/approximation.

The *Tucker decomposition* of a d-dimensional tensor $\mathbf{F} = [f_{ij\dots k}]$ reads

$$\mathbf{F} = \mathbf{G} \times_1 U \times_2 V \dots \times_d W, \qquad f_{ij\dots k} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \cdots \sum_{k'=1}^{r_d} g_{i'j'\dots k'} u_{ii'} v_{jj'} \dots w_{kk'}.$$

The right-hand side can be also regarded as *Tucker approximation* of $\mathbf{F}$. Another name used for this kind of data structure is *Higher-Order SVD (HO-SVD)* decomposition/approximation. The quantities $r_1, r_2, \dots, r_d$ are referred to as *Tucker ranks* or *mode ranks*, the tensor $\mathbf{G}$ of size $r_1 \times r_2 \times \dots \times r_d$ is called the *Tucker core* tensor.

Here and below, the symbol $\times_l$ denotes the multiplication of a tensor by a matrix along the l-th mode. For example, $\mathbf{H} = \mathbf{F} \times_2 M$ means $h_{ij\dots k} = \sum_{j'} m_{jj'} f_{ij'\dots k}$.

3 Tucker approximation algorithms

In this section we briefly overview some classical Tucker approximation algorithms and recently proposed efficient cross approximation methods. On input, tensors are given in the canonical format with possibly large canonical rank, as in the quantum chemistry applications. We remind a recent SVD-based factor filtering algorithm [3] and then propose a new, much faster algorithm using the Cross2D approximation of canonical factors. On several chemical examples we show that the new method is the same accurate as and 500 times faster than the SVD-based methods.

In the general case, a large-scale tensor may have no eminent structure and be defined in a certain implicit way. Whatever this way is, we assume that a procedure which allows us to compute any entry on demand is available. In these cases we have to revoke more complicated algorithms.

The widely used Tucker approximation method is based on three SVDs for the unfolding matrices. The cost amounts to $\mathcal{O}(n^4)$ operations for a 3-dimensional tensor of size $n \times n \times n$, which is what we cannot afford for $n \gtrsim 1000$. Instead, we can apply the recently developed Cross3D algorithm [13, 11] and its multilevel version [3]. Whenever possible, we give numerical results for the aforementioned algorithms applied to the electronic density approximation for a sample of molecules, with timings and reliably estimated error values.

3.1 Tensors in the canonical form

The tensor canonical form

$$f_{ijk} = f(x_i, y_j, z_k) = \sum_{t=1}^{R} a_{it} b_{jt} c_{kt}, \qquad \mathbf{F} = (A, B, C), \tag{1}$$

is standard as input and output for many chemical modelling programs (e.g. MOLPRO or GAMESS). In [3] we have proposed a few algorithms exploiting this kind of input and yielding a data reduction output in the Tucker format with a prescribed accuracy to be kept. In chemical examples, the mode ranks r_1, r_2, r_3 turn out to be remarkably smaller than R. When seeking a low-rank approximation for the canonical factors, we must use carefully selected accuracy bounds, individual for each factor. Low-rank approximation is naturally computed by means of the SVD, as we exactly did in [3]. Here we pursue a new line and replace the SVD with the Cross2D algorithm. In the result, we derive a much faster algorithm, in which the approximation is constructed even without computation of all elements of the canonical factors. The *skeleton/cross approximation* [6, 7] is obtained using a cross of 'very important' columns and rows of a given matrix (i.e. a canonical factor), with a guaranteed approximation accuracy so long as the intersection matrix is picked up close to a *maximum volume* submatrix [8, 5], and can be computed via cross approximation procedures [15, 13].

3.1.1 Full orthogonalization and Tucker compression

A trivial way to make use of a given canonical structure is proposed by Algorithm 1.

```
Algorithm 1 (Canonical → Tucker) global filtering
Input: $\mathbf{F} = (A, B, C)$
Output: Approximation $\tilde{\mathbf{F}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$, with accuracy $\|\mathbf{F} - \tilde{\mathbf{F}}\|_F \leq \varepsilon \|\mathbf{F}\|_F$.
 1: Perform QR decompositions of the canonical factors
        $A =: UR_a, \quad B =: VR_b, \quad C =: WR_c, \qquad R_a, R_b, R_c \in \mathbb{R}^{R \times R}.$
 2: Assemble the core array $\mathbf{G} = (R_a, R_b, R_c) \in \mathbb{R}^{R \times R \times R}$.
 3: Apply standard Tucker approximation procedure (see [14, 10] or Algorithm 5 of this
    paper) to core tensor $\mathbf{G}$.
```

However, the Tucker approximation step has severe memory and time limitations (see Table 1) and, therefore, this algorithm is not acceptable even for very simple molecules, because initial tensor ranks can be quite large, e. g.

$$R(CH_4) = 1334, \quad R(C_2H_6) = 3754, \quad R(C_2H_5OH) = 6970,$$

and so the approximation problem for the $R \times R \times R$ core tensor $\mathbf{G}$ is not much simpler than it is for the initial $n \times n \times n$ tensor with $n \sim 10^4$.

3.1.2 SVD-based filtering

To convey the data reduction problem from tensor-based to matrix-based, we can think of separate truncation for each of the mode basis matrices U, V, W. That requires independent accuracy criteria for approximation of the canonical factors.

As is proved in [3], the accuracy extimate

$$\|\mathbf{F} - \tilde{\mathbf{F}}\|_F \leq \varepsilon \|\mathbf{F}\|_F$$

Table 1: Storage and time requirements for Tucker approximation via 3 SVDs

molecule	CH_4	C_2H_6	C_2H_5OH
tensor rank R	1334	3744	6945
memory for R^3 elements	17GB	400GB	2.4 TB
time for SVD-s*	8 hours	22 days	258 days

* Time is given for hypothetic 10GHz CPU, working at 100% efficiency on SVD algorithm with complexity $32mn\min(m,n)$ flops (where $m \times n$ is matrix size).

is achieved whenever

$$\|A - \tilde{A}\|_F \le \varepsilon_1, \quad \|B - \tilde{B}\|_F \le \varepsilon_2, \quad \|C - \tilde{C}\|_F \le \varepsilon_3$$

with the thresholds

$$\varepsilon_1 = (\varepsilon/3)\|\mathbf{F}\|_F/\gamma_A, \quad \varepsilon_2 = (\varepsilon/3)\|\mathbf{F}\|_F/\gamma_B, \quad \varepsilon_3 = (\varepsilon/3)\|\mathbf{F}\|_F/\gamma_C \tag{2}$$

and the kind of condition numbers

$$\gamma_A^2 = \sum_{t=1}^{R} \|b_t\|_2^2\|c_t\|_2^2, \quad \gamma_B^2 = \sum_{t=1}^{R} \|c_t\|_2^2\|a_t\|_2^2, \quad \gamma_C^2 = \sum_{t=1}^{R} \|a_t\|_2^2\|b_t\|_2^2. \tag{3}$$

The very three matrix approximation problems can be solved by the SVDs for canonical factors followed by truncation of singular vectors corresponding to the singular values below the thresholds $\varepsilon_1, \varepsilon_2$ and ε_3. To complete the prescriptions, note that the norms can be efficiently computed by the formula

$$\|\mathbf{F}\|_F^2 := \sum_{i,j,k} |f_{ijk}|^2 = \sum_{i,j,k} \left(\sum_{t=1}^{R} a_{it} b_{jt} c_{kt} \right)^2 = \sum_{t,\tau=1}^{R} \left[(A^T A) \circ (B^T B) \circ (C^T C) \right]_{t\tau}, \tag{4}$$

where "$\circ$" designates the element-by-element (Hadamard, Schur) multiplication of matrices.

This approach is summarized in Algorithm 2. It requires all elements of canonical factors A, B, C to be computed and the overal complexity is $\mathcal{O}(nR^2)$ if $n \ge R$ or $\mathcal{O}(n^2R + nR^2)$ otherwise.

3.1.3 Cross2D versus SVD

Data reduction computations generally hinge on the following low-rank approximation problem: given a matrix $F \in \mathbb{R}^{n \times R}$, find an approximation $\tilde{F} = UGV^T$ with a prescribed relative accuracy ε.

It can be solved by the SVD, provided that all the entries are given as a full array. However, there are faster methods based on the *skeleton decomposition* $\tilde{F} = UGV^T$,

```
Algorithm 2 (Canonical → Tucker) Individual factor filtering via SVD
Input: $\mathbf{F} = (A, B, C)$
Output: Approximation $\tilde{\mathbf{F}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$, with accuracy $\|\mathbf{F} - \tilde{\mathbf{F}}\|_F \le \varepsilon \|\mathbf{F}\|_F$.
1: Compute the norm of a given data array $\mathbf{F} = (A, B, C)$ by (4) in $\mathcal{O}(nR^2)$ operations.
2: Perform SVD-s of canonical factors
       $$A =: U\Sigma_1\Phi_1, \quad B =: V\Sigma_2\Phi_2, \quad C =: W\Sigma_3\Phi_3,$$
   with appropriate truncation of Tucker factors $U, V, W$ by thresholds, given by (2) to
   $\rho_1, \rho_2$ and $\rho_3$ basis set, respectively. This step requires $\mathcal{O}(nR\min(n, R))$ operations.
3: Rewrite $\tilde{\mathbf{F}} = (U\Sigma_1\Phi_1, V\Sigma_2\Phi_2, W\Sigma_3\Phi_3)$ as the Tucker decomposition
       $$\tilde{\mathbf{F}} = \mathbf{G} \times_1 U \times_2 V \times_3 W, \quad \mathbf{G} = (\Sigma_1\Phi_1, \Sigma_2\Phi_2, \Sigma_3\Phi_3) \in \mathbb{R}^{\rho_1 \times \rho_2 \times \rho_3}$$
   and assemble $\mathbf{G}$ in $\mathcal{O}(Rr_1r_2r_3)$ operations. This step may be omitted if we decide
   to keep $\mathbf{G}$ in canonical form, as proposed in [9].
```

where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{R \times r}$ consist of columns and rows of F, and $G = B^{-1}$, where B is a $r \times r$-submatrix at the intersection of the columns of U and rows of V. This approach involves only those entries of F that belong to the cross of the columns of U and rows of V.

The skeleton approximation accuracy depends on the choice of cross. In [6, 7, 8] it is proposed to look for the cross with the intersection matrix B of the *maximal volume* (determinant in modulus) among all $r \times r$ submatrices. In this case a good accuracy bound is guaranteed to hold. An efficient algorithm to find a 'sufficiently good' cross was first proposed in [15]. It is expounded in more detail and with important related questions in [5].

The search problem for the maximum-volume submatrix is NP-difficult. Thus, we have to be satisfied with a 'sufficiently good' submatrix and some heuristic algorithms. Since these algorithms are to fetch a cross of some columns and rows, we call them *cross approximation* algorithms.

The simplest, fast and fortunately quite effective method for the cross acquisition turns out to be the Gauss elimination method using some pivoting technique over dynamically selected sets of the entries in the active matrix. A pivoting strategy with kind of minimal information was considered in [1]; despite the simplicity, this method is liable to break-downs (may quit when a good approximation is not obtained).

A cheap practical remedy proposed in [13] is a restarted version of this cross method. For the reader's convenience, a sketch of this method is given by Algorithm 3.

Other robust and still simple pivoting strategies during the cross approximation are suggested in [4].

In Table 2 we compare accuracy of rank-2 approximations of a 5×5 matrix and show

Algorithm 3 Cross2D matrix approximation

```
Input: Matrix $F$.
Output: Rank-$r$ matrix $\tilde{F} = UV^T = \sum_{k=1}^{r} u_k v_k^T$ such that $\|F - \tilde{F}\| \le \varepsilon \|F\|_F$.
1: Initialisation: Set $p = 1$, $\tilde{F} = 0$. Pick random column $j_p$ in matrix $F$.
2: Calculate column $j_p$ of the matrix $F$ and subtract from all elements the corresponding elements of $\tilde{F}$. In the resulting vector find the largest magnitude element. Suppose it is located in the row $i_p$.
3: Calculate the row $i_p$ of the residual $F - \tilde{F}$ and the next pivot which is its largest magnitude element with a restriction that the element from the $j_p$-th column can not be chosen again. Suppose this pivot is located in the $j_{p+1}$-th column.
4: Calculate the new cross with centre at $(i_p, j_p)$. Update $\tilde{F}_{new} = \tilde{F} + u_p v_p^T$ to make the approximation *exact* on the elements, occupied by new cross.
5: Write estimate of residual norm $\|F - \tilde{F}_{new}\|_F$ based on value of elements of $F - \tilde{F}$, computed on positions of cross $(i, j_p)$. If stopping criterion
       $$\|F - \tilde{F}_{new}\|_F \le \varepsilon \|\tilde{F}_{new}\|_F$$
   is not satisfied, set $\tilde{F} = \tilde{F}_{new}$, $p := p + 1$ and repeat from step 2.
6: If current approximation seems to be good, check the accuracy on some random set of matrix elements. Estimate the residual norm again and if it is still good, finish. In other case, restart from step 2 with $\tilde{F} = \tilde{F}_{new}$ and new pivot in position $j_{p+1}$, corresponding to maximum in modulus element of residual in the checked set.
```

that accuracy obtained by Algorithm 3 is close to that of the best cross approximation, maximum-volume cross approximation and as well to the SVD-based truncation.

Although it cannot be so in all cases, in many practical applications we find that the accuracy of Algorithm 3 is reasonably close to that of the SVD truncation [7, 8, 13]. Thus, we may consider Algorithm 3 as a low-rank approximation tool for the canonical factors. This leads immediately to a faster Tucker approximation method.

3.1.4 Cross2D-based filtering

Basically we only need to change Step 2 of Algorithm 2: replace the SVD with the cross approximation procedure (Algorithm 3).

A problem is that the accuracy bounds (2) depend on the tensor norm $\|\mathbf{F}\|_F$, which is not known in advance and cannot be found anymore by (4), because we do not want to increase the compexity up to $\mathcal{O}(nR^2)$. The tensor norm should be estimated on-fly, as well as the norms of approximated canonical mode vectors. Namely, if the canonical factors are somehow approximated by skeleton decompositions of ranks ρ_1, ρ_2, ρ_3

$$A \approx \tilde{A} = UK^T, \quad B \approx \tilde{B} = VL^T, \quad C \approx \tilde{C} = WM^T,$$

Table 2: SVD and Cross rank-2 approximation results for 5×5 matrix

$$\begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix}$$

method		volume	error, $\times 10^{-3}$
SVD			1.57
best cross	$[1,3] \times [1,3]$	0.0208	2.77
maxvol cross	$[1,4] \times [1,4]$	0.0225	3.08
Cross2D	$[1,2] \times [1,3]$	0.0166	4.76
random cross		0.0008	40.0

then we can write

$$\mathbf{F} \approx \tilde{\mathbf{F}} = (\tilde{A}, \tilde{B}, \tilde{C}) = (N^T, M^T, L^T) \times_1 U \times_2 V \times_3 W$$

and estimate $\|\mathbf{F}\|_F$ by the core tensor norm $\|(K^T, L^T, M^T)\|_F$ in $\mathcal{O}(\rho_1\rho_2\rho_3 R)$ operations.

A tricky thing is still how to check the accuracy of this estimate and to recompute it efficiently after each rank-one update. All implementation details are gathered in Algorithm 4.

In the description of this algorithm we hide internal Cross2D variables and pivot indices and use Cross2D as a routine that computes the cross in a given matrix and returns a rank-one update for the approximation under construction.

3.1.5 Numerical results

Given the electron density in the canonical form (1), we apply the SVD-based Algorithm 2 and the Cross2D-based Algorithm 4, and compare timings and mode ranks ρ_1, ρ_2, ρ_3 delivered by both methods.

Then, we apply the Tucker approximation algorithm [14] (cf. Algorithm 5 below) to the obtained core tensor $\mathbf{G}$ and re-approximate the core by $\tilde{\mathbf{G}}$ as follows:

$$\|\mathbf{G} - \tilde{\mathbf{G}}\|_F \leq \varepsilon \|\mathbf{G}\|_F, \qquad \mathbf{G} = \mathbf{H} \times_1 U' \times_2 V' \times_3 W',$$
$$\tilde{\mathbf{F}} = \mathbf{H} \times_1 UU' \times_2 VV' \times_3 WW', \qquad \mathbf{H} \in \mathbb{R}^{r_1 \times r_2 \times r_3}.$$

As a rule, this leads to a reduction in the mode ranks, because the threshold values $\varepsilon_1, \varepsilon_2$ and ε_3 chosen in accordance with (2) may be found in the end excessively small. This additional compression requires $\mathcal{O}(\rho^4)$ flops with $\rho = \max(\rho_1, \rho_2, \rho_3)$ and give us

Algorithm 4 (Canonical $\to$ Tucker) Individual factor filtering via Cross2D

```
Input: $\mathbf{F} = (A, B, C)$
Output: Approximation $\tilde{\mathbf{F}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$, with accuracy $\|\mathbf{F} - \tilde{\mathbf{F}}\|_F \le \varepsilon \|\mathbf{F}\|_F$.
 1: Initialisation: $U = V = W = 0,\ \mathbf{G} = 0,\ \mathit{upd}A = \mathit{upd}B = \mathit{upd}C = \mathit{upd}\mathbf{G} = \mathbf{true}$.
 2: for all modes $\mu = 1, 2, 3$, i.e. factors $A, B, C$ do
 3:   {part for $\mu = 2$, factor $B \approx \tilde{B} = VL^T,\ V \in \mathbb{R}^{n \times \rho_2}, L \in \mathbb{R}^{R \times \rho_2}, V^T V = I.$}
 4:   if $\mathit{upd}B$ then
 5:     Do one step of Algorithm 3 to find rank-one update $\tilde{B} := \tilde{B} + vl^T$
 6:     Project $v$ on span$(V)$ to find $\hat{v}$ such that $v = [V\hat{v}]x,\ [V\hat{v}]^T[V\hat{v}] = I$.
        Then $\tilde{B} = VL^T + [V\hat{v}]xl^T = [V\hat{v}]([L0] + lx^T)^T$
 7:     if $\mathit{upd}\mathbf{G}$ then
 8:       Update $\mathbf{G} := (K^T, [L0]^T, M^T) + (K^T, lx^T, M^T)$. In element-wise notation

          $$g_{ijk} = \sum_{\alpha=1}^{R} k_{\alpha i}[L0]_{\alpha j} m_{\alpha k} + \sum_{\alpha=1}^{R} k_{\alpha i} l_\alpha x_j m_{\alpha k},$$

          and it is easy to see that
          first term expands $\mathbf{G}$ from size $\rho_1 \times \rho_2 \times \rho_3$ to $\rho_1 \times (\rho_2 + 1) \times \rho_3$ by zeroes,
          second term is rank-one update for unfolding $G^{(\mu)}_{j\,ki} = G^{(\mu)}_{j\,ki} + x_j (M^T \operatorname{diag}(l) K)_{ki}$,
          where $\operatorname{diag}(l)$ denotes diagonal matrix defined by vector $l$.
 9:       Compute $\|\mathbf{F}\|_F := \|\mathbf{G}\|_F$. {Tucker factors $U, V, W$ are orthonormal}
10:       if $\|(K^T, lx^T, M^T)\|_F \le \varepsilon_g \|\mathbf{G}\|_F$ for the last $n_g$ updates then
11:         {Stop updating norm if it remains stable. We use $\varepsilon_g \sim 10^{-3}, n_g \sim 10.$}
12:         $\mathit{upd}\mathbf{G}$ = false.
13:       end if
14:     end if
15:     Set $V := [V\hat{v}],\ L := [L0] + lx^T$.
16:     Compute $\|b_t\|_2 = \|l_t\|_2$ for $t = 1, \ldots, R$, evaluate condition numbers by (3)
        and update $\varepsilon_1, \varepsilon_2, \varepsilon_3$ by (2).
17:     if $\|lx^T\|_F \le \varepsilon_2$ for last $n_b$ updates then
18:       $\mathit{upd}B$ := false {Stop updating $B$, it converges. We use $n_a = n_b = n_c = 3.$}
19:     else if $\|lx^T\|_F \ge \varepsilon_g \|B\|_F$ then
20:       $\mathit{upd}\mathbf{G}$ := true {If $B$ changes significantly, we have to recompute $\|\mathbf{G}\|_F$.}
21:     end if
22:   end if
23:   {Enough for current mode, repeat for others}
24: end for
25: If $\mathit{upd}A$ or $\mathit{upd}B$ or $\mathit{upd}C$ is required, repeat from step 2.
```

Table 3: Ranks and timings for 'filtering' algorithms, $n = 10241$

ε	by SVD		by Cross2D		r_1, r_2, r_3
			$\mathbf{F}(CH_4)$	-0:06[*]	
10^{-5}	$61 \times 61 \times 61$	3:20	$67 \times 72 \times 69$	0:05	$28 \times 28 \times 28$
10^{-7}	$79 \times 79 \times 79$	3:20	$84 \times 88 \times 83$	0:07	$41 \times 41 \times 41$
10^{-9}	$96 \times 96 \times 96$	3:20	$101 \times 103 \times 102$	0:10	$55 \times 55 \times 55$
10^{-11}	$113 \times 113 \times 113$	3:20	$118 \times 121 \times 121$	0:15	$66 \times 66 \times 66$
			$\mathbf{F}(C_2H_6)$	-0:17	
10^{-5}	$66 \times 73 \times 98$	42:30	$66 \times 81 \times 108$	0:10	$20 \times 36 \times 30$
10^{-7}	$85 \times 94 \times 125$	42:30	$89 \times 103 \times 135$	0:16	$30 \times 50 \times 44$
10^{-9}	$103 \times 113 \times 149$	42:30	$114 \times 125 \times 161$	0:24	$45 \times 69 \times 59$
10^{-11}	$120 \times 130 \times 175$	42:30	$127 \times 142 \times 190$	0:31	$59 \times 84 \times 75$
			$\mathbf{F}(C_2H_5OH)$	-0:27	
10^{-5}	$101 \times 99 \times 131$	244:00	$106 \times 113 \times 145$	0:22	$48 \times 48 \times 47$
10^{-7}	$129 \times 127 \times 166$	244:00	$146 \times 146 \times 180$	0:39	$69 \times 69 \times 72$
10^{-9}	$155 \times 153 \times 200$	244:00	$167 \times 167 \times 219$	0:57	$96 \times 94 \times 98$
10^{-11}	$181 \times 177 \times 232$	244:00	$201 \times 194 \times 248$	1:33	$116 \times 115 \times 122$

[*] Time is given as mm:ss. It is measured on 3.0GHz Pentium4 CPU, with code compiled by GNU Fortran 4.3.2 compiler, optimised by `-O2 options and linked with GotoBLAS-1.25 library.`

'true' values of the Tucker ranks r_1, r_2, r_3 for the considered tensors. From the results collected in Table 3, we can see that the mode ranks produced by the SVD and Cross2D method are very close, but the Cross2D method is up to 500 times faster than the SVD approach.

We also present the initialization time, i.e. time for the computation of entries of matrices A, B, C (with the negative sign for the Cross2D data) to lay an extra stress on the fact that the Cross2D method can be even faster than the computation of all matrix elements. In this sense we can say that the Cross2D-based filtering is made in 'negative time'.

Once the input data are in the canonical format, a fast accuracy check is implemented via the formula

$$\|(A,B,C) - \mathbf{G} \times_1 U \times_2 V \times_3 W\|_F^2 = \|(A,B,C)\|_F^2 - 2\,\langle (A,B,C),\ \mathbf{G} \times_1 U \times_2 V \times_3 W \rangle + \|\mathbf{G}\|^2, \tag{5}$$

where the first term is evaluated by (4), the third is computed trivially, and the second is transformed as follows:

$$\langle (A,B,C),\ \mathbf{G} \times_1 U \times_2 V \times_3 W \rangle := \sum_{ijk} \left(\sum_{\alpha=1}^{R} a_{i\alpha} b_{j\alpha} c_{k\alpha} \right) \left(\sum_{i'j'k'} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'} \right) =$$

$$= \sum_{\alpha=1}^{R} \sum_{i'j'k'} g_{i'j'k'} (U^T A)_{i'\alpha} (V^T B)_{j'\alpha} (V^T C)_{k'\alpha} = \sum_{\alpha=1}^{R} \mathbf{G} \times_1 (U^T A)_{:\alpha} \times_2 (V^T B)_{:\alpha} \times_3 (V^T C)_{:\alpha}.$$

Since the right-hand side of (5) gives the squared error, the estimated error should not be lower than the square root of the machine precision. Thus, this formula suits only for approximations with accuracies less than the square root of machine precision.

In all considered experiments the application of (5) ensures that our heuristic methods compute the approximations with a required accuracy.

3.2 General tensors

3.2.1 Tucker approximation via SVD-s of unfoldings

The standard Tucker approximation method for general unstructured tensors is based on the SVD decompositions of the so-called *unfolding* matrices, see Algorithm 5.

Algorithm 5 Tucker approximation of general three-dimensional tensor

```
Input: $\mathbf{F} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.
Output: Approximation $\tilde{\mathbf{F}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$, with accuracy $\|\mathbf{F} - \tilde{\mathbf{F}}\|_F \le \varepsilon \|\mathbf{F}\|_F$.
 1: Reshape tensor to unfoldings
        $F^{(1)} = [f^{(1)}_{ijk}] = [f_{i(jk)}],\quad F^{(2)} = [f^{(2)}_{ijk}] = [f_{j(ki)}],\quad F^{(3)} = [f^{(3)}_{ijk}] = [f_{k(ij)}],$
    that are of size $n_1 \times n_2 n_3$, $n_2 \times n_1 n_3$ and $n_3 \times n_1 n_2$ and consist of columns, rows
    and fibres $\mathbf{F}$, respectively.
 2: Perform three SVD-s
        $F^{(1)} = U \Sigma_1 \Phi_1^T,\qquad F^{(2)} = V \Sigma_2 \Phi_2^T,\qquad F^{(3)} = W \Sigma_3 \Phi_3^T.$
    Left ('short') singular vectors of these matrices after the truncation of vectors, corre-
    sponding to the singular values below the desired threshold, give the factors $U, V, W$
    of the Tucker decomposition.
 3: Tucker core is computed as contraction of the data array with the Tucker factors
        $g_{i'j'k'} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} f_{ijk} u_{ii'} v_{jj'} w_{kk'},\qquad \text{or}\qquad \mathbf{G} = \mathbf{F} \times_1 U^T \times_2 V^T \times_3 W^T.$
```

Although this algorithm is simple and easily allows one to control the accuracy of approximation, it is not feasible for large-scale tensors because of memory and time limitations, see Table 4. We can (and do) apply this method only to sufficiently small

Table 4: Storage and time requirements for Tucker approximation via 3 SVDs

grid size n	1281	2561	5121
memory for n^3 elements	16GB	125GB	1 TB
time for SVD-s*	7 hours	5 days	76 days

* Time is given for hypothetic 10GHz CPU, working at 100% efficiency on SVD algorithm with complexity $32mn\min(m,n)$ flops (where $m \times n$ is matrix size).

Tucker core tensors in pursuit of data re-approximation with lower values of the mode ranks.

3.2.2 Cross3D algorithms

The main advantage of this method is in combination of stability granted by Algorithm 5 and linear complexity of Algorithm 3. Basically we need to find low-rank approximations for unfolding matrices, just like in Algorithm 5, but the Cross2D is used instead of the SVD. In order to approximate an unfolding matrix, we compute a set of 'short', size-n columns directly representing some fibres of the initial tensor and 'long' rows consisting of n^2 elements and representing vectorized *slices* of the initial tensor. The short fibres to be chosen are called *VIP fibres*, for they span the dominating subspace for all fibres of the given tensor, as the columns of Tucker factors do in Algorithm 5. To compute the required slices with a linear in n complexity we apply Cross2D again for elements of the slice matrix. This results in linear complexity in the grid size, with $\mathcal{O}(nr^2)$ evaluations of tensor elements and $\mathcal{O}(nr^4)$ additional operations. In a practical implementation the complexity can be further reduced to $\mathcal{O}(nr^\alpha)$ evaluations (with $1 \le \alpha \le 2$) and $\mathcal{O}(nr^3)$ additional operations by several 'matrix tricks' [13, 11, 5].

3.2.3 Cross3D multilevel algorithms

To benefit from smoothness of original function, in [3] we proposed a multilevel version of the Cross3D algorithm. Therein, we consider the cross algorithm as a method to find indices of VIP fibres of a given tensor. To reduce the complexity for large-scale problems, we can determine the bulk of these VIP fibres on coarser grids, and then refine the approximation with taking up just few extra mode vectors on finer grids. For smooth data, the multilevel Cross3D method is scaled down to $\mathcal{O}(nr)$ evaluations of tensor elements and $\mathcal{O}(nr^2)$ additional operations. If the tensor entries are defined by a function which is the sum of R terms with separated variables or by the cubic root of this function, we spare order of R operations for evaluation of any individual entry and come up with the total complexity $\mathcal{O}(nRr + nr^2)$. Since $r \le R$, the complexity is linear in all principal parameters.

The common idea of extracting some essential information from coarser grids appears also in [9], where positions of most important fibres (MIFs) are determined via the

Table 5: Ranks and timings for Cross3D algorithms, $n = 10241$

ε	Cross3D		Cross3D-ml		r_1, r_2, r_3
			$\mathbf{F}(CH_4)$		
10^{-5}	$25 \times 28 \times 32$	3:00	$62 \times 36 \times 53$	1:35*	$28 \times 28 \times 28$
10^{-7}	$45 \times 47 \times 47$	6:30	$63 \times 73 \times 69$	2:55	$41 \times 41 \times 41$
10^{-9}	$56 \times 59 \times 59$	8:25	$86 \times 80 \times 82$	4:45	$55 \times 55 \times 55$
10^{-11}	$72 \times 73 \times 70$	15:20	$101 \times 111 \times 110$	7:15	$66 \times 66 \times 66$
			$\mathbf{F}(C_2H_6)$		
10^{-5}	$22 \times 37 \times 32$	12:50	$33 \times 52 \times 60$	4:10	$20 \times 36 \times 30$
10^{-7}	$34 \times 57 \times 47$	19:30	$50 \times 73 \times 74$	5:40	$30 \times 50 \times 44$
10^{-9}	$50 \times 76 \times 61$	28:00	$66 \times 96 \times 90$	8:00	$45 \times 69 \times 59$
10^{-11}	$64 \times 88 \times 79$	40:00	$91 \times 133 \times 110$	22:30	$59 \times 84 \times 75$
			$\mathbf{F}(C_2H_5OH)$		
10^{-5}	$49 \times 51 \times 55$	39:00	$73 \times 66 \times 72$	9:50	$48 \times 48 \times 47$
10^{-7}	$79 \times 79 \times 79$	63:20	$108 \times 93 \times 123$	15:25	$69 \times 69 \times 72$
10^{-9}	$104 \times 101 \times 107$	93:50	$133 \times 138 \times 156$	26:00	$96 \times 94 \times 98$
10^{-11}	$123 \times 123 \times 134$	110:20	$195 \times 180 \times 213$	58:20	$116 \times 115 \times 122$
			$\mathbf{F}^{1/3}(CH_4)$		
10^{-5}	$38 \times 35 \times 39$	4:30	$57 \times 54 \times 65$	2:15	$32 \times 32 \times 32$
10^{-7}	$52 \times 52 \times 54$	8:10	$88 \times 94 \times 95$	5:20	$51 \times 51 \times 51$
10^{-9}	$81 \times 80 \times 79$	19:10	$132 \times 122 \times 130$	11:45	$74 \times 74 \times 74$
10^{-11}	$108 \times 109 \times 108$	47:50	$166 \times 166 \times 162$	23:00	$101 \times 101 \times 101$
			$\mathbf{F}^{1/3}(C_2H_6)$		
10^{-5}	$27 \times 40 \times 31$	15:10	$37 \times 71 \times 67$	4:55	$22 \times 40 \times 33$
10^{-7}	$41 \times 73 \times 61$	21:30	$66 \times 106 \times 96$	7:30	$37 \times 64 \times 56$
10^{-9}	$65 \times 99 \times 88$	46:30	$90 \times 141 \times 120$	14:10	$59 \times 92 \times 79$
10^{-11}	$92 \times 133 \times 119$	88:00	$134 \times 202 \times 180$	37:10	$87 \times 125 \times 109$
			$\mathbf{F}^{1/3}(C_2H_5OH)$		
10^{-5}	$62 \times 63 \times 58$	50:40	$84 \times 81 \times 86$	12:15	$53 \times 53 \times 53$
10^{-7}	$93 \times 94 \times 98$	94:00	$137 \times 134 \times 139$	26:10	$87 \times 86 \times 88$
10^{-9}	$142 \times 134 \times 143$	197:00	$177 \times 180 \times 201$	57:10	$125 \times 122 \times 129$
10^{-11}	$176 \times 180 \times 198$	471:00	$262 \times 256 \times 270$	149:00	$170 \times 168 \times 175$

* Time is given as mm:ss. It is measured on 3.0GHz Pentium4 CPU, with code compiled by GNU Fortran 4.3.2 compiler, optimised by `-O2 options and linked with GotoBLAS-1.25 library.`

maximum energy principle, found on a coarser grid and then corrected on finer grids in another variant of multilevel approximation algorithm. However, constructions of that paper explicitly use canonical factors of the input tensor. Since these factors are not known for the cubic root of this function, the multilevel method of [9] cannot be applied in this case. Note that our multilevel algorithm [3] is free from this kind of limitation, as it needs only a procedure for computation of a tensor entry with given indices.

3.2.4 Numerical results

In the experiments of this section we do not use advantages of canonical input for $\mathbf{F}$, neither for $\mathbf{F}^{1/3}$, where this format is lost. Multi-level versions of the cross algorithm are applied to 'black box' tensors of which we know only by a method of evaluation of a tensor element on required positions. The approximated tensors never appear as full arrays: only a small number of principle fibres is computed.

We can not compare our methods with Algorithm 5 on the considered grids, for, as is clearly shown in Table 4, the full-size SVDs would need memory and time beyond affordable. In Table 5 we present the ranks and timings for the Cross3D method and its multilevel version. We can see that the number of VIP fibres picked up by the multilevel Cross3D is larger compared to the standard Cross3D, but the overall computation time falls down by a factor of 2 or 3, because most of the computations are performed on coarser levels.

To check the accuracy, we compare initial tensor entries with those of the Tucker approximation at a large random set of randomly picked positions. As soon as the dispersion of the relative error appears to be small, we compare the obtained estimate for the relative error with the required accuracy bound. In the end we are led to conclusion that the approximations computed by our algorithm are as accurate as required.

4 Fast mimic approximation for the cubic root

Although the (multilevel) cross algorithm is able to tackle the case of cubic root of a well-represented function, it is a way less efficient than the above proposed pre-filtering technique for the canonical factors, in case they are available. In this section we are going to suggest a new algorithm that has almost the same efficiency, yet with a certain restriction on accuracy possible in the current implementation.

The idea behind this new algorithm is that VIP positions for $\mathbf{F}$ may serve well also as VIP positions for $\mathfrak{f}\mathbf{F} := \mathbf{F}^{1/3}$ and probably other smooth functions of $\mathbf{F}$. We mean that fibres of $\mathfrak{f}\mathbf{F}$ on those positions span the principal subspace of all fibres of $\mathfrak{f}\mathbf{F}$ with a sufficiently good accuracy. In this case the VIP structure for $\mathfrak{f}\mathbf{F}$ *mimics* the one for $\mathbf{F}$.

The workhorse tool of this method is the maximal volume principle implemented by a routine named `maxvol`. It locates a so-called *dominant* submatrix, a certain approximation to the one of maximal volume [8, 5]. The complexity of `maxvol` for $n \times r$ matrices with $n > r$ is $nr^2 + cnr$, where the number of iterative steps c is usually from

5 to 50. In practice, time requirements of this method are negligible even in comparison with a pretty fast Algorithm 4.

Algorithm 6 Fast mimic approximation

```
Input: $\mathbf{F} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ with mode ranks $r_1, r_2, r_3$.
Output: Approximation for $\mathfrak{f}\mathbf{F} \approx \mathbf{T} = \mathbf{H} \times_1 U \times_2 V \times_3 W$, that may be good.
1: {Find VIP sets of fibres in $\mathbf{F}$ by the following steps}
2: Call maxvol to find dominating submatrices in $U, V$ and $W$. Each maximum-volume
   submatrix is defined by set of indices $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ respectively.
3: Compute subtensor $\mathbf{A} = \mathbf{F}_{\square} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ at positions defined by $\mathcal{J}_1 \times \mathcal{J}_2 \times \mathcal{J}_3$.
4: Write unfoldings of subtensor $A^{(1)} \in \mathbb{R}^{r_1 \times r_2 r_3}$, $A^{(2)} \in \mathbb{R}^{r_2 \times r_3 r_1}$, $A^{(3)} \in \mathbb{R}^{r_3 \times r_1 r_2}$.
   Call maxvol to find dominating submatrices in unfoldings $\mathbf{A}^{(\mu)}$, that is defined by
   a set of $r_\mu$ multi-indices (positions) $\mathcal{I}_\mu$, $\mu = 1, 2, 3$.
5: Compute VIP sets of rows $\mathcal{U} = \mathbf{F}_{\mathcal{I}_1}$, columns $\mathcal{V} = \mathbf{F}_{\mathcal{I}_2}$ and fibres $\mathcal{W} = \mathbf{F}_{\mathcal{I}_3}$, that
   consists of elements of given tensor on VIP sets of positions $\mathcal{I}_1, \mathcal{I}_2$ and $\mathcal{I}_3$.
6: {We suppose that Tucker-like form $\tilde{\mathbf{F}} = \hat{\mathbf{G}} \times_1 \mathcal{U} \times_2 \mathcal{V} \times_3 \mathcal{W}$ (as fact Tucker form, but
   without the restriction of orthogonality of factors) can provide a good approximation
   to $\mathbf{F}$ after appropriate choice of $\hat{\mathbf{G}}$. But now we are interested in VIP factors, not
   approximation for $\mathbf{F}$.}
7: Compute VIP sets in $\mathfrak{f}\mathbf{F}$ at the *same positions* $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$. Denote them $\mathfrak{f}\mathcal{U}, \mathfrak{f}\mathcal{V}, \mathfrak{f}\mathcal{W}$.

8: {We suppose that Tucker-like form $\mathbf{T} = \mathbf{H} \times_1 \mathfrak{f}\mathcal{U} \times_2 \mathfrak{f}\mathcal{V} \times_3 \mathfrak{f}\mathcal{W}$ can provide a good
   approximation to $\mathfrak{f}\mathbf{F}$ after appropriate choice of $\mathbf{H}$. The following is to find good
   core tensor $\mathbf{H}$ in linear by $n$ complexity.}
9: Call maxvol to find dominating submatrices in $\mathfrak{f}\mathcal{U}, \mathfrak{f}\mathcal{V}$ and $\mathfrak{f}\mathcal{W}$. Each maximum-
   volume submatrix $(\mathfrak{f}\mathcal{U})_\square, (\mathfrak{f}\mathcal{V})_\square$ and $(\mathfrak{f}\mathcal{W})_\square$ is defined by set of indices $\mathcal{K}_1, \mathcal{K}_2$ and
   $\mathcal{K}_3$ respectively.
10: Compute subtensor $\mathbf{B} = \mathfrak{f}\mathbf{F}_\square \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ at positions defined by $\mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{K}_3$.
11: Find $\mathbf{H}$ to satisfy equation $\mathfrak{f}\mathbf{F} = \mathbf{T}$ on positions of $\mathbf{B}$ *exactly*.
    $$\mathbf{B} = \mathbf{H} \times_1 (\mathfrak{f}\mathcal{U})_\square \times_2 (\mathfrak{f}\mathcal{V})_\square \times_3 (\mathfrak{f}\mathcal{W})_\square, \quad \mathbf{H} = \mathbf{B} \times_1 (\mathfrak{f}\mathcal{U})_\square^{-1} \times_2 (\mathfrak{f}\mathcal{V})_\square^{-1} \times_3 (\mathfrak{f}\mathcal{W})_\square^{-1}. \quad (6)$$

12: Return $\mathbf{T} = \mathbf{H} \times_1 \mathfrak{f}\mathcal{U} \times_2 \mathfrak{f}\mathcal{V} \times_3 \mathfrak{f}\mathcal{W}$ as Tucker-like approximation for $\mathfrak{f}\mathbf{F}$. If orthogonality
    of Tucker factors is required, perform QR decompositions
    $$\mathfrak{f}\mathcal{U} = \hat{U} R_u, \qquad \mathfrak{f}\mathcal{V} = \hat{V} R_v, \qquad \mathfrak{f}\mathcal{W} = \hat{W} R_w,$$
    compute new core $\hat{\mathbf{H}} = \mathbf{H} \times_1 R_u \times_2 R_v \times_3 R_w$ and return
    $$\mathbf{T} = \hat{\mathbf{H}} \times_1 \hat{U} \times_2 \hat{V} \times_3 \hat{W}.$$
```

Steps 1-7 of this Algorithm are required only to locate positions of VIP sets of rows, columns and fibres in the given tensor. They can be omitted if these positions are

Table 6: Accuracy of fast mimic approximation algorithm, $n = 5121$

ε	r_1, r_2, r_3	ε_1	ε_2	ε_3
		$\mathbf{F}^{1/3}(CH_4)$		
10−4	21 × 21 × 21	0.4.10−4	1.3.10−4	4.8.10−3
10−5	29 × 29 × 29	0.4.10−5	1.0.10−5	2.0.10−3
10−6	35 × 35 × 35	0.4.10−6	1.3.10−6	5.0.10−4
10−7	41 × 41 × 41	0.5.10−7	3.0.10−7	1.2.10−4
10−8	48 × 48 × 48	0.5.10−8	3.4.10−8	2.0.10−5
10−9	55 × 55 × 55	0.5.10−9	1.5.10−9	2.0.10−5
10−10	60 × 60 × 60	0.4.10−10	3.5.10−10	3.8.10−6
10−11	67 × 67 × 67	0.4.10−11	5.6.10−11	3.2.10−7
10−12	74 × 74 × 74	0.4.10−12	1.9.10−12	1.1.10−7
		$\mathbf{F}^{1/3}(C_2H_6)$		
10−4	15 × 25 × 23	0.4.10−4	2.6.10−4	3.9.10−3
10−5	20 × 36 × 30	0.5.10−5	3.4.10−5	8.0.10−4
10−6	24 × 45 × 37	0.5.10−6	3.3.10−6	3.8.10−4
10−7	30 × 50 × 44	0.5.10−7	2.8.10−7	1.2.10−4
10−8	38 × 60 × 52	0.4.10−8	2.4.10−8	1.5.10−5
10−9	45 × 69 × 59	0.5.10−9	2.3.10−9	5.7.10−6
10−10	52 × 78 × 67	0.5.10−10	2.0.10−10	2.7.10−6
10−11	58 × 84 × 75	0.5.10−11	3.0.10−11	8.3.10−7
10−12	67 × 94 × 83	0.5.10−12	7.2.10−12	2.9.10−7
		$\mathbf{F}^{1/3}(C_2H_5OH)$		
10−4	35 × 35 × 35	0.5.10−4	2.0.10−4	7.2.10−3
10−5	47 × 47 × 47	0.5.10−5	3.6.10−5	2.4.10−3
10−6	59 × 60 × 61	0.5.10−6	3.3.10−6	4.1.10−4
10−7	70 × 70 × 71	0.5.10−7	3.0.10−7	1.2.10−4
10−8	83 × 82 × 85	0.4.10−8	5.0.10−8	6.0.10−5
10−9	96 × 94 × 98	0.5.10−9	3.9.10−9	1.0.10−5
10−10	105 × 104 × 110	0.5.10−10	4.4.10−10	3.0.10−6
10−11	116 × 115 × 122	0.5.10−11	4.1.10−11	1.5.10−6
10−12	128 × 127 × 134	0.5.10−12	4.7.10−12	7.7.10−7

already known from the output of some other method, e. g. Cross3D.

Notation $\mathfrak{f}$ can define either a point-wise operation applied to elements of the tensor or VIP sets of its fibres, or a discretization of some transformation of the original function. In any case, the underlying hypothesis behind this method is that the VIP structure for $\mathfrak{f}\mathbf{F}$ mimics one for the initial tensor $\mathbf{F}$, i. e. the indices of VIP sets for $\mathbf{F}$ make out a 'sufficiently representative' set to capture the behavior of $\mathfrak{f}\mathbf{F}$. This hypothesis definitely requires a more precise formulation in future research.

To demonstrate advantages of this mimic approximation method, we apply it to $\mathbf{F}^{1/3}$ with $\mathbf{F}$ being the electron density for a few simple molecules. The initial approximation for $\mathbf{F} = (A, B, C)$ is provided by Algorithm 4. The results are given in Table 6. Here

- ε is the required accuracy for Tucker approximation for $\mathbf{F}$;
- r_1, r_2, r_3 are mode ranks of Tucker approximation for $\mathbf{F}$ given by Cross2D filtering;
- ε_1 is the relative error checked for the initial approximation for $\mathbf{F}$ by Alg. 4;
- ε_2 is the relative error checked for approximation of $\mathbf{F}$ by $\tilde{\mathbf{F}} = \hat{\mathbf{G}} \times_1 \mathcal{U} \times_2 \mathcal{V} \times_3 \mathcal{W}$, based on non-orthogonal VIP basises $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ (it appears on step 6 of Algorithm 6). The core tensor $\hat{\mathbf{G}}$ is found in a simple way, just like in (6).
- ε_3 is accuracy check for approximation for $\mathbf{F}^{1/3}$, given by Algorithm 6.

The persuasive accuracy check of our algorithms includes *full* computation of the involved residuals for $n = 5121$. This exhaustive computational work was performed on parallel distributed memory platforms. The tests for CH_4 and C_2H_5OH molecules were performed on `MVS100K` cluster in the JSCC RAS, the verification for C_2H_5OH molecule was done on `SKIF` cluster in the SRCC MSU.

We do not provide timings for Algorithm 6, because they are negligible in comparison with timings for algorithm 4 applied to $\mathbf{F}$ (see Table 3). Practically, mimic approximation for $\mathbf{F}^{1/3}$ is of the same cost as the approximation of $\mathbf{F}$ by Algorithm 4.

Acknowledgements

This work was partially supported by the RFBR grant 08-01-00115 and joint RFBR/DFG grant 09-01-91332. The authors acknowledge the use of computational resources of the Joint SuperComputer Center of Russian Academy of Sciences (`www.jscc.ru`) and the Scientific Research Computer Centre of Moscow State University. Part of this work was done during the stay of the second author 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 and to Boris Khoromskij and Venera Khoromskaja for useful discussions.

References

[1] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.

[2] G. Beylkin and M. J. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. USA*, 99(16):10246–10251, 2002.

[3] H.-J. Flad, B.N. Khoromskij, D.V. Savostyanov, and E.E. Tyrtyshnikov. Verification of the cross 3D algorithm on quantum chemistry data. *Rus. J. Numer. Anal. Math. Model.*, 23(4):210–220, 2008.

[4] J.M. Ford and E.E. Tyrtyshnikov. Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related systems. *SIAM J. Sci. Comp.*, 25(3):961–981, 2003.

[5] S.A. Goreinov, I.V. Oseledets, D.V. Savostyanov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. How to find a good submatrix. Research Report 08-10, ICM HKBU, Kowloon Tong, Hong Kong, 2008. Available from World Wide Web: www.math.hkbu.edu.hk/ICM/pdf/08-10.pdf.

[6] S.A. Goreinov, E.E. Tyrtyshikov, and N.L. Zamarashkin. Pseudo–skeleton approximations of matrices. *Reports of Russian Academy of Sciences*, 342(2):151–152, 1995.

[7] S.A. Goreinov, E.E. Tyrtyshikov, and N.L. Zamarashkin. A theory of pseudo–skeleton approximations. *Linear Algebra Appl.*, 261:1–21, 1997.

[8] S.A. Goreinov and E.E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 208:47–51, 2001.

[9] B.N. Khoromskij and V. Knoromskaia. Multigrid Accelerated Two-level Tensor Approximation. Preprint 40, MPI MIS, Leipzig, 2008.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multlinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.

[11] I.V. Oseledets, D.V. Savostianov, and E.E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J. Matrix Anal. Appl.*, 30(3):939–956, 2008.

[12] I.V. Oseledets and E.E. Tyrtyshnikov. Approximate inversion of matrices in the process of solving a hypersingular integral equation. *Comp. Math. and Math. Phys.*, 45(2):302–313, 2005.

[13] D.V. Savostianov. *Polilinear approximation of matrices and integral equations*. PhD thesis, INM RAS, 2006. (in Russian).

[14] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[15] E. E. Tyrtyshnikov. Incomplete cross approximation in the mosaic–skeleton method. *Computing*, 64(4):367–380, 2000.

[16] E.E. Tyrtyshnikov. Tensor approximations of matrices generated by asymptotically smooth functions. *Sbornik: Mathematics*, 194(5-6):941–954, 2003.

[17] E.E. Tyrtyshnikov. Kronecker-product approximations for some function-related matrices. *Linear Algebra Appl.*, 379:423–437, 2004.