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

[P U B
I N M
R A S]
pub.inm.ras.ru

Dmitry V. Savostyanov, Ivan V. Oseledets

Fast adaptive interpolation of multi-dimensional arrays in tensor train format

Preprint 2011-03 30-05-2011

Moscow 2011

Fast adaptive interpolation of multi-dimensional arrays in tensor train format

Dmitry V. Savostyanov, Ivan V. Oseledets

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

30 May, 2011

Abstract. Using recently proposed tensor train format for the representation of multi-dimensional dense arrays (tensors) we develop a fast interpolation method to approximate the given tensor by using only a small number of its elements. The algorithm is based on DMRG scheme, known among the quantum chemistry society. It is modified to make an interpolation on the adaptive set of tensor elements. The latter is selected using the maximum-volume principle which was previously used for the cross approximation schemes for matrices and 3-tensors. The numerical examples includes the interpolation of one– and many–dimensional functions on the uniform grids.
Keywords: High-dimensional problems, tensor train format, adaptive sampling, interpolation algorithms, maximum volume principle
AMS classification: 15A23, 15A69, 65D05

1. Introduction

Muldimensional arrays, or *tensors* $\mathbf{A} = [A(i_1, \ldots, i_d)]$ with d indices (or modes) introduce a number of challenging problems for numerical analysis and scientific computing. The number of tensor elements grows exponentially in d. Even if a *mode size* (i.e. the number of possible values of each index) of a tensor is small, the storage cost for all elements is

[$]This work was supported in part by RFBR grants 09-01-12058, 10-01-00757, RFBR/DFG grant 09-01-91332, Russian Federation Gov. contracts No. П1178, П1112 and П940, 14.740.11.0345 and President Grant 140.2011.1.

prohibitive for large d. To overcome this so-called *curse of dimensionality*, compact low-parametric formats of tensors are crucial.

Two data-sparse tensor formats are well established. The canonical format (CANDECOMP, PARAFAC, CP) [16, 4, 14] is a discrete analogue of *separation of variables*. The tensor is said to be in the CP format if

$$A(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha).$$

It is widely used in different applications in data analysis, see review [18] and references therein. For $d = 2$, this equation turns into dyadic (skeleton) decomposition of a matrix $A(i_1, i_2)$, which can be computed in a stable way via singular value decomposition (SVD). However, for $d \geqslant 3$ everything changes: computation of the canonical rank is an NP-hard problem [15] that can be also ill-posed [8]. There are quite successfull algorithms [14, 3, 4, 7, 23] but they are not always convergent.

The Tucker decomposition [27] has the form

$$A(i_1, \ldots, i_d) = \sum_{\mathbf{a}} G(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d),$$

and is characterized by the *Tucker factors* U_k and the *Tucker core* $\mathbf{G}$. This decomposition is stable, and quasioptimal approximation can be computed via a multilinear SVD (or Higher Order SVD, HOSVD) [5, 6]. However it is inapplicable for large d due to the memory required to store the core $\mathbf{G} = [G(\mathbf{a})] = [G(\alpha_1, \ldots, \alpha_d)]$. Thus, both of decompositions generalize the SVD to high-order tensors, but both of them have disadvantages.

The tensor train decomposition (TT) [21, 19] is in the middle between the canonical decomposition and the Tucker decomposition. It has the following form

$$A(i_1, i_2, i_3, \ldots, i_d) = G_1(i_1) G_2(i_2) G_3(i_3) \ldots G_d(i_d), \tag{1}$$

where $G_k(i_k)$ is a $r_{k-1} \times r_k$ matrix for each index i_k. To make the matrix-by-matrix product a scalar, *boundary conditions* $r_0 = r_d = 1$ have to be introduced. The number of parameters in Eq. (1) does not grow exponentially with d, and the approximation in the TT format is stable and can be computed by SVD-based algorithms.

The SVD-based algorithm (TT-SVD) [24, 25] computes the decomposition / approximation of a given tensor in the TT format (1) with a guaranteed accuracy. However, it requires all elements of a given tensor, which is too large even for small d. Approximation of such tensors appears in particular when dealing with functions of many variables. Evaluation of each tensor element can be done via some subroutine, but computation of full function-related tensor is intractable. Thus, it is interesting to design a method to compute the TT approximation of a tensor using only few entries of it.

Since the TT decomposition is a generalization of the SVD, it is instructive to look at the matrix case. For matrices, the problem is solved via the *skeleton decomposition*, and an adaptive low-rank approximation can be computed by cross approximation techniques [28, 2]. The skeleton decomposition shows that a rank-r matrix can be exactly recovered from r columns and r rows of the matrix that have a nonsingular $r \times r$ submatrix on their intersection. In the approximate low-rank case, different submatrices provide different level of approximation quality. The *maximum volume principle* [11] states, that the submatrix with

maximal volume, i.e. determinant in modulus, provides a quasioptimal approximation. To compute a submatrix of a large volume, heuristic algorithm [10] was proposed (but cross approximation is usually faster).

In [22] the cross method was nontrivially generalized to three-dimensional arrays and Tucker format. The resulted Cross3D method is also based on the maximal volume principle. Since the Tucker format has exponential scaling in d, Cross3D is applicable only for tensors of small orders.

For arbitrary dimension the generalization of the two-dimensional interpolation formulae to the TT-format was proposed in [25]. It is a simple alternating algorithm, based on maximal volume principle. In the case when the tensor has exact TT representation with ranks bounded by r, it is reconstructed using the proposed interpolation formula from $\mathcal{O}(dnr^2)$ tensor elements. The major drawback is that all TT ranks $r_1, \dots r_{d-1}$ have to be properly guessed in advance. If the TT ranks are underestimated, the approximation will be poor, and if they are overestimated, the complexity increases greatly. Thus, our goal is to develop a version of the TT cross method that can adapt the TT ranks of the approximation according to the desired accuracy.

The idea that helps to solve this problem comes from the Density Matrix Renormalization Group approach [30], which is well-known in solid state physics. Surprisingly, it can be adapted for the TT cross method. This idea is especially helpful for tensors with small mode sizes. Such tensors appear in the context of quantization and the QTT format [20, 17].

Analogous approaches have been proposed for other tensor formats. For the canonical format the heuristic algorithm was proposed in [9]. It is based on minimization of the residue on the adaptively computed set of points. This approach, however, inherits all "bad" properties of the canonical decomposition.

Recently, for the $\mathcal{H}$-Tucker format [13, 12], an adaptive black-box version based on the successive cross approximation of subtensors was presented by Ballani, Grasedyck and Kluge, with good examples of efficiency [1]. The complexity is quasioptimal in the dimension $\mathcal{O}(d \log d)$. The proposed algorithm for the TT format is based on absolutely different approach and is linear in d. It optimizes the TT ranks on each step, and has polynomial complexity in ranks.

The paper is organized as follows. In Section 2 basic notations and definitions are gathered. In Section 3 a basic DMRG scheme for tensor approximation is presented, and in Section 4 it is showed how to modify the basic DMRG step to make it an *interpolation* using maximal volume principle. In Section 5 all algorithmic details are presented (including an intermediate orthogonalization step and the the computation of auxiliary maximal volume submatrices), and a theorem on the worst-case behaviour of the algorithm is presented. Section 6 presents numerical experiments.

2. Basic notations and definitions

Notations are important when working with tensors and their representations. The tensor $\mathbf{A}$ is said to be in the TT-format with cores G_k, if Eq. (1) holds. The matrices $G_k(i_k)$ can be also considered as $r_{k-1} \times n_k \times r_k$ 3-tensors, which are called TT-cores. Further in this paper we often represent high-dimensional arrays as parameter-dependent matrices. For example, $A(i, j)$ denotes some $p \times q$ matrix, depending on two parameters, i, j. Common matrix operations can be performed with such a matrix. On the other hand, it can be

considered as a 4-dimensional tensor of size $p \times m \times n \times q$. The usual indexing for a tensor fits into this notation: $A(i_1, \ldots, i_d)$ is a "parameter-dependent scalar" and scalar is a 1×1 matrix.

When dealing with tensors and matrices, MATLAB notation is often helpful, in particular the `reshape` function. It takes a d-dimensional $n_1 \times \ldots \times n_d$ tensor $\mathbf{A}$ and a size vector $[m_1, \ldots, m_s]$ such that $\prod_{k=1}^{s} m_k = \prod_{i=1}^{d} n_d$, and produces s-dimensional $m_1 \times \ldots \times m_s$ tensor $\mathbf{B}$ comprised from elements of $\mathbf{A}$ as follows

$$A(i_1, \ldots, i_d) = B(j_1, \ldots, j_s),$$
$$\sum_{k=1}^{d} i_k \prod_{k'=1}^{k-1} n_{k'} = \sum_{l=1}^{s} j_l \prod_{l'=1}^{l-1} m_{l'}.$$

Here and after we suppose that tensor indexing starts from zero.

A compact notation is also used for multiindices. Approximation of tensors requires computation of certain subsets of its entries. It is common that these subsets are *subtensors* which appear from restricting certain indices of a tensor to a subset of indices, or, more generally, certain *group of indices* is replaced by a single index. To select subtensors of a given tensor so-called *index sets* will be used, denoted by calligraphic letters. We will illustrate this procedure by an example. Suppose the following index sets are given

$$\mathcal{I}_{k-1} = \{\mathcal{I}_{k-1}^{(\alpha)}\} = \{(i_1^{(\alpha)}, \ldots, i_{k-1}^{(\alpha)}), \quad \alpha = 1, \ldots, r_{k-1}\},$$
$$\mathcal{J}_{k+1} = \{\mathcal{J}_{k+1}^{(\beta)}\} = \{(i_{k+2}^{(\beta)}, \ldots, i_d^{(\beta)}), \quad \beta = 1, \ldots, r_{k+1}\}.$$

Then these index sets define a $r_{k-1} \times n_k \times n_{k+1} \times r_{k+1}$ subtensor $\mathbf{B} = [A(\mathcal{I}_{k-1}, i_k, i_{k+1}, \mathcal{J}_{k+1})]$ with entries

$$B(\alpha, i_k, i_{k+1}, \beta) = A(\mathcal{I}_{k-1}^{(\alpha)}, i_k, i_{k+1}, \mathcal{J}_{k+1}^{(\beta)}),$$
$$\alpha = 1, \ldots, r_{k-1}, \qquad \beta = 1, \ldots, r_{k+1}.$$

In the paper also notions of the tensor-by-matrix multiplication will be used. For a d-dimensional tensor $\mathbf{A}$ and an $n_k \times m_k$ matrix U the tensor-by-matrix product over mode k is defined as an $n_1 \times \ldots \times n_{k-1} \times m_k \times n_{k+1} \ldots \times n_d$ tensor $\mathbf{B} = \mathbf{A} \times_k U$ with elements

$$B(i_1, \ldots, i_{k-1}, j_k, i_{k+1}, \ldots, i_d) = \sum_{i_k=1}^{n_k} A(i_1, \ldots, i_d) U(i_k, j_k).$$

Two norms will be used for tensors: the Frobenius norm,

$$\|A\|_F = \sqrt{\sum_{i_1, \ldots, i_d} |A(i_1, \ldots, i_d)|^2},$$

and the Chebyshev norm,

$$\|A\|_C = \max_{i_1, \ldots, i_d} |A(i_1, \ldots, i_d)|.$$

3. ALS and DMRG schemes

Our main problem is the following: given a tensor $\mathbf{A}$ we want to approximate it by a tensor $\mathbf{B}$ in the TT format with a prescribed accuracy in some norm (usually Frobenius) and TT ranks of $\mathbf{B}$ being as small as possible. Even for tensors of small size it is a good question. For the canonical format robust procedures are not available, maybe the most illustrative example is the tensor of multiplication of 3×3 matrices. For the TT-format the situation is different: there is the TT-SVD algorithm which computes the TT decomposition by taking SVD decompositions of auxiliary unfolding matrices. In quantum information analogous algorithm was proposed by Vidal [29], but the algorithm of [21, 19] has proven and controlled error bound, and the approximation obtained has indeed the prescribed relative accuracy. The disadvantage of the TT-SVD algorithm is that it requires the full tensor and full SVD decompositions. A viable alternative in tensor approximations is the alternating least squares algorithm (ALS). It is not difficult to design such an algorithm for the approximation in the TT format. Given a current approximation of the form

$$\mathbf{A}(i_1, \ldots, i_d) \approx \mathbf{B}(i_1, \ldots, i_d) = B_1(i_1) \ldots B_d(i_d),$$

one fixes all cores except the one with the number k. The least squares problem

$$\|\mathbf{A} - \mathbf{B}\|_F \to \min$$

is then reduced to the linear system with $r_{k-1} n_k r_k$ unknowns. It is natural to organize the ALS iteration in *sweeps*: first the first core, then the second, and so long to the last core ("left-to-right sweep") and then backwards ("right-to-left sweep"). Of course, this is not the only possible ordering, and there can be cases where different sweep strategies are more efficient. But in this way the residue of ALS decays monotonically. Moreover, if the TT ranks of $\mathbf{B}$ (which do not change in the ALS iteration) are properly chosen, then the convergence obtained in the experiments is very fast.

For the canonical format the convegence of ALS is usually poor, but the TT format is based on matrix decomposition. Thus, for matrix low-rank approximation, the ALS method reduces to block power method, which convergence rate is proportional to the $(r+1)$–th singular value. If it is zero (exact low-rank case) then convergence is in two iterations. The same appears to be true in the experiments with the TT format. The mathematical proof of this fact has yet to be found.

The main problem of the ALS in the TT-format is the requirement to specify all the TT ranks. There are $d-1$ of them, so a "try-and-guess" procedure is complicated, and the only possible alternative is to overestimate the rank. That leads to a higher complexity. This problem was first tackled in solid state physics in modelling quantum spin systems. One has to solve eigenvalue problem for a high-dimensional Hamiltonian, and the solution for a k–spin system has S^k degrees of freedom. Usually S is small, $S = 2$ or $S = 4$. Solution is sought in the TT format. In the eigenvalue computations, Rayleigh quotient (Ax, x) has to be minimized, and the idea of ALS can be used. Such approach was called Wilson Renormalization Group and has the same drawbacks as the ALS scheme.

The idea of the Density Matrix Renormalization Group (DMRG), proposed by White [30] was only later realized as an idea of constrained optimization in the matrix product state [26]. It proceeds in the same fashion as ALS, but with one important difference:

the optimization is performed over two cores B_k, B_{k+1} *simultaneously*. This is still non-quadratic optimization, and a new supecore (or superblock) is introduced as follows

$$W_k(i_k, i_{k+1}) = B_k(i_k)B_{k+1}(i_{k+1}). \tag{2}$$

This removes the information about the r_k. The optimization problem over W_k is now linear. When W_k is found, cores B_k, B_{k+1} are recovered from (2) by means of the SVD. The matrix to be decomposed has size $r_{k-1}n_k \times n_{k+1}r_{k+1}$ (thus no exponential dependence on d), and the most important thing is that the rank r_k can be be determined adaptively. This rank-reduction is called the *decimation step* of the DMRG.

There are important details, of course, how to implement the DMRG efficiently. For eigenvalue problems it leads to "small" eigenvalue problems for W. An important problem is the control of the accuracy in the process, since the rank truncation introduces some error. The computations maintaing rigorous error bound can be done elegantly exploiting properties of the TT format. They will be described later on, and after that we will show how to modify the DMRG scheme to avoid full tensors. The key idea is to use the cross approximation instead of the pseudoinverse.

4. DMRG for TT interpolation

The approximation problem

$$\|\mathbf{A} - \mathbf{B}\|_F \to \min \tag{3}$$

is to be solved by the DMRG approach. Note that in fact k-th step of the DMRG is equivalent to the ALS step, applied to the order $(d-1)$ tensor with modes k and $k+1$ merged into a "long mode". Thus all formulae can be derived in terms of an ALS step, and then plugged into the DMRG scheme. Suppose then all cores of the tensor $\mathbf{B}$ except B_k are fixed. Then (3) is a following linear least squares problem for B_k

$$U(i_1, \ldots, i_{k-1})B_k(i_k)V(i_{k+1}, \ldots, i_d) \approx A(i_1, \ldots, i_d).$$

It is natural to consider $U(i_1, \ldots, i_{k-1})$ as an $N \times \alpha_{k-1}$ matrix U, where $N = n_1 \cdot \ldots \cdot n_{k-1}$, i.e. the rows of U are indexed by a multiindex $I = (i_1, \ldots, i_d)$. The same is for $V(i_{k+1}, \ldots, i_d)$ which is natural to consider as an $\alpha_{k+1} \times M$ matrix, where $M = n_{k+1} \cdot \ldots \cdot n_d$, with columns indexed by a multiindex $J = (i_{k+1}, \ldots, i_d)$. The problem for $B_k(i_k)$ is then cast as

$$UB_k(i_k)V \approx \tilde{A}_k(i_k), \tag{4}$$

where an $N \times M$ matrix $\tilde{A}_k(i_k)$ in position (I, J) has element $A(I, i_k, J)$. The local problem (4) in least squares sense is solved as

$$B_k(i_k) = U^{\dagger}A_k(i_k)V^{\dagger}. \tag{5}$$

The computation of the pseudoinverses $U^{\dagger}$ and $V^{\dagger}$ is expensive, but it can be done very cheap in during the sweep: U and V can be kept orthogonal! The orthogonalization algorithm in the TT format is very cheap and robust. It is based on the left/right–orthogonality of the TT cores.

Definition 1. The $r_{k-1} \times n_k \times r_k$ core $Q_k(i_k)$ is called *left–orthogonal* if

$$\sum_{i_k=1}^{n_k} Q_k^{\top}(i_k) Q_k(i_k) = I_{r_k},$$

i.e. the $r_{k-1} n_k \times r_k$ matrix obtained as a reshape of Q_k has orthogonal columns.

Definition 2. The $r_{k-1} \times n_k \times r_k$ core $Q_k(i_k)$ is called *right–orthogonal* if

$$\sum_{i_k} Q_k(i_k) Q_k^{\top}(i_k) = I_{r_{k-1}},$$

i.e. the $r_{k-1} \times n_k r_k$ matrix obtained as a reshape of Q_k has orthogonal rows.

Lemma 1. If an $n_1 n_2 \dots n_k \times r_k$ matrix $Q = [Q(i_1, i_2, \dots, i_k, \alpha_k)]$ is represented in the form

$$Q(i_1, \dots, i_k, \alpha_k) = Q_1(i_1) Q_2(i_2) \dots Q_k(i_k)$$

with left-orthogonal cores $Q_k(i_k)$, then Q has orthonormal columns, $QQ^{\top} = I_{r_k}$, i.e.

$$\sum_{i_1, \dots, i_k} Q(i_1, \dots, i_k, \alpha_k) Q(i_1, \dots, i_k, \beta_k) = \delta(\alpha_k, \beta_k).$$

The analogous structure with right-orthogonalal cores leads the matrix with orthogonal rows. Note, the cores can not be both left and right orthogonal. In the ALS and DMRG iterations it is natural to keep the cores $B_1, \dots, B_{k-1}$ which constitute the matrix U left-orthogonal, and cores $B_{k+1}, \dots, B_d$, which constitute the matrix V right-orthogonal. If it holds, then U has orthonormal columns, and V has orthonormal rows, and (5) is reduced to

$$B_k(i_k) = U_k^{\top} A_k(i_k) V^{\top}.$$

Moreover, due to the orthogonality of U and V, an approximation error introduced to $A_k(i_k)$ by decimation step in the DMRG will be the same for the whole tensor $B_k(i_k)$. This allows full error control. The only thing that is left to describe is how to make core orthogonal.

The algorithm that make the cores orthogonal is a part of the TT-rounding procedure from [21, 19] , is very simple and has a low complexity. It proceeds incrementally. It is easy to left-orthogonalize the first core by the QR decomposition of an a $n_1 \times r_1$ matrix:

$$U_1(i_1) = Q_1(i_1) R_1,$$

and the new TT-representation becomes

$$U = Q_1(i_1) U_2'(i_2) \dots U_k(i_k),$$

where $U_2'(i_2) = R_1 U_2(i_2)$. Suppose now U has orthogonalized cores

$$U = Q_1(i_1) \dots Q_s(i_s) U_s'(i_s) \dots U_k(i_k)$$

for some $s < k$. Then $U_s'(i_s)$ is represented as

$$U_s'(i_s) = Q_s(i_s) R_s,$$

by the QR-decomposition applied to an $r_{s-1} n_s \times r_s$ matrix which is a reshaped core $U_s'(i_s)$ (see definition of the left-orthogonality). The matrix R_s is then transfered to the next core and the process continues.

Another advantage is that during the ALS (or DMRG) sweep only one core has to orthogonalized. For example, after the k-th core has been recomputed, it has to be left-orthogonalized by the QR and that is all.

5. Maximal volume principle and TT-RC method

5.1. Replacing least squares by interpolation

The standard DMRG/ALS algorithm requires all elements of tensor to be computed: just look and the basic step (5) which requires full matrix-by-matrix products. For high-dimensions it is not possible. Thus, some alternative has to be presented. In the matrix case one can think of cross approximation techniques, which interpolate a given low-rank matrix from a small number of so-called *crosses*. However in the DMRG setting only a slight modification should be made and it is based on another way of approximate solution of the problem (4), which is a set of matrix problems of form

$$A \approx U\Phi V, \tag{6}$$

where a (small) matrix Φ of size $p \times q$ is sought, and matrices U and V have orthonormal columns and orthonormal rows, respectively. Recall our main assumption: A can be approximated by a matrix of form $U\Phi V$ with a very small error, i.e. we seek for a good approximation. The cases where sufficiently accurate approximations does not exist, are not considered. Then, the problem (6) can be solved by the *interpolation*. Indeed, if an adequate index set I with p elements, and J with q elements are found, then in the exact case Φ can be recovered by formula

$$\Phi = \hat{U}^{-1}\hat{A}\hat{V}^{-1}, \tag{7}$$

where $\hat{U} = U(:, I), \hat{V} = V(J, :), \hat{A} = A(I, J)$. Theoretically, all submatrices $\hat{U}, \hat{V}$ are equally good in the exact case, if they are only nonsingular. However, when (6) is valid only with some accuracy ε, the error can be amplified greatly if $\hat{U}$ or $\hat{V}$ are badly conditioned. The submatrix should be of a "good quality". The *maximal volume submatrix* is a good candidate for such a good submatrix, and it can be computed robustly by `maxvol` algorithm [10]. Moreover, if

$$A = U\Phi V + E,$$

where $\|E\| = \varepsilon$, then $\tilde{\Phi}$ computed from the maximal-volume submatrices in U and V by interpolation (7), satisfies

$$\|\tilde{\Phi} - \Phi\| \leqslant \varepsilon c(n, r),$$

where the constant $c(n, r)$ depends mildly on n and r. The only problem is how to compute these submatrices. The number of elements depends exponentially on d. Thus, as a replacement to maximal volume submatrices, an algorithm that computes a good submatrix in a TT-structured matrix is proposed. In may not produce the true maximal volume submatrix, but it appears to be sufficiently good in many cases. Rigorous mathematical estimates for the approximation quality that fit numerical experiments are to be developed, but the TT-maxvol algorithm is simple and provides good results.

5.2. Maximum volume in TT-tensors

The basic algorithm is the computation of a "good" submatrix in a tall $n \times r$ matrix that is close to the maximal-volume submatrix. Such algorithm (**maxvol**) was provided in [10], is fast and robust. The idea is to sequentially update the submatrix by row permutations. Indeed, if $\hat{A}$ is the current approximation to the maximal volume submatrix, then it can

be assumed without the loss of generality that it is located in the first r rows of A, and one has to find the maximal volume submatrix in a matrix

$$A\hat{A}^{-1} = \begin{pmatrix} I \\ B \end{pmatrix},$$

If a maximal in modulus element in B is located in the position (i, j) and is equal to $1 + \delta$ in modulus (with $\delta > 0$), then it is not difficult to show that after the permutation of the row i and the row j, the leading submatrix will have the determinant $(1 + \delta)$, thus the determinant can be increased until all elements in B are not greater than 1 in modulus. This is not strictly a maximal volume submatrix. Such submatrix can be called *dominant*. The maximal volume submatrix is a dominant submatrix. Fortunately, cross approximation estimates hold for all dominant submatrices. There are several details that significantly improve the speed of the basic algorithm, see [10].

In our case, the matrix has few columns and too much rows, so the original maxvol algorithm can not be applied. Fortunately, the matrix in question possesses additional TT-structure:

$$U = U_1(i_1) \dots U_k(i_k). \tag{8}$$

The analogous situation is with SVD and QR-decompositions. But QR-decomposition of U can be computed fast *exactly*. The situation with the maxvol submatrix is different, however the approach can be generalized in the same fashion. To avoid the computation of full columns of U, we restrict ourselves to the certain subsets of these columns. Consider the case $k = 2$:

$$U = U(i_1, i_2) = U_1(i_1)U_2(i_2).$$

The idea is first to compute good indices $\mathcal{J}_1$ in U_1 and then consider rows $(i_1^{(k)}, i_2^{(k)})$ of U only with $i_1^{(k)} \in \mathcal{J}_1$. It is easy to generalize this method to an arbitrary k. The index sets $\mathcal{J}_1, \dots, \mathcal{J}_k$ are called *left-nested*, if

$$\forall (i_1, \dots, i_k) \in \mathcal{J}_k \quad \text{it holds} \quad (i_1, \dots, i_{k-1}) \in \mathcal{J}_{k-1}.$$

Suppose that the index sets $\mathcal{J}_1, \dots, \mathcal{J}_{k-1}$ are computed and left-nested. Then the interpolating rows are selected from a $(r_{k-1}n_k) \times r_k$ submatrix $U(\mathcal{J}_{k-1}, i_k)$ of the matrix U.

The matrix U can be made orthogonal using the TT-QR algorithm. By the algorithm described above it is guaranteed that the computed submatrix will be nonsingular. The quality of this submatrix in the worst case is estimated by the following Theorem.

Theorem 1. Let $B(i)$ be a $1 \times p$ matrix for all $i = 1, \dots, n$ and $C(j)$ — $p \times r$ matrix for all $j = 1, \dots m$. Consider

$$A(i, j) = B(i)C(j),$$

as an $nm \times r$ matrix with (i, j) indexing its rows.

Consider a submatrix of A computed in the following way: First compute the maxvol submatrix in an $n \times p$ matrix B with indices $\mathcal{J}$, and then the maxvol submatrix $A_\square$ in the rectangular submatrix of $\mathbf{A}$ of size $pm \times r$ comprised from the elements $A(\mathcal{J}, j)$. Then $A_\square$ is quasi-dominant:

$$\|AA_\square^{-1}\|_C \leqslant p.$$

Proof. First, let us find the dominant submatrix in B, i.e. the set of indices

$$\mathcal{I} = \operatorname{maxvol} B, \qquad B_{\square} := B[\mathcal{I}], \qquad \#\mathcal{I} = p.$$

The domination property of the submatrix $B_{\square}$ ensures

$$\|BB_{\square}^{-1}\|_C \leqslant 1.$$

This means that every row of B is a linear combination of rows from $B(\mathcal{I})$ with coefficients bounded by 1 in modulo.

Then consider a rectangular submatrix

$$A_{\diamond} := A(\mathcal{I}j) := B(\mathcal{I})C(j)$$

and find the dominant square submatrix in it

$$\mathcal{K} = \operatorname{maxvol} A(\mathcal{I}j), \qquad A_{\square} = A(\mathcal{K}), \qquad \#\mathcal{K} = r.$$

The domination property of the submatrix $A_{\square}$ ensures $|A_{\diamond}A_{\square}^{-1}| \leqslant 1$. For the whole matrix A we have

$$A(i,j) = B(i)C(j) = \gamma(\mathcal{I})B(\mathcal{I})C(j), \qquad |\gamma| \leqslant 1.$$

Finally,

$$\begin{aligned} \|A(i,j)A_{\square}^{-1}\|_C &\leqslant p\|B(\mathcal{I})C(j)A_{\square}^{-1}\|_C \\ &= p\|A(\mathcal{I},j)A_{\square}^{-1}\|_C = p\|A_{\diamond}A_{\square}^{-1}\|_C \\ &\leqslant p. \end{aligned} \tag{9}$$

□

So, the submatrix computed at one step of the algorithm is at most p times worse than the dominant one, and in the worst case it would lead to error detirioration by a factor of p. This procedure has to be done d times, thus the worst-case bound for this algorithm (we leave the name **maxvol** or **TT-maxvol** to it) is εr^d, where ε is the best possible accuracy. This looks not very optimistic, however our numerical experiments confirm that usually the estimate is much better. Conditions under which the exponential estimate can be improved should be investigated in more details, but the algorithm is constructive and seems to work very well, thus it will be used without any modifications. Formal description of the algorithm is presented in the Algorithm 1. Note the role of auxiliary matrices P_k. One has to keep track of submatrices of tall matrices U and V in (7), and since $U(i_1,\dots,i_k) = U(i_1,\dots,i_{k-1})U_k(i_k)$ the submatrix at the next step can be cheaply computed from the previous ones by the matrix-by-matrix multiplication. These submatrices are stored as auxiliary submatrices P_k in Algorithm 1. The current interpolation indices, $\mathcal{I}_k$, are treated analogously due to their nestedness. When maximal volume indices (denote them by I_k) in an auxiliary $r_{k-1}n_k \times r_k$ matrix, are computed, then $\mathcal{I}_k$ is computed as follows: each row index corresponds to a pair (α_{k-1}, i_k), where $1 \leqslant \alpha_{k-1} \leqslant r_{k-1}$ and $1 \leqslant i_k \leqslant n_k$. The index α_{k-1} indicates, which multiindex from $\mathcal{I}_{k-1}$ is chosen, therefore the required multiindex in $\mathcal{I}_k$ is $\mathcal{I}_{k-1}(\alpha_{k-1}, i_k)$. This is what is meant by "compute $\mathcal{J}_k$ from $\mathcal{J}_{k-1}$ and J_k" in Algorithm 1.

Algorithm 1: TT-RC (Tensor Train Renormalization Cross)

Input: Subroutine to compute a prescribed element $A(i_1, \ldots, i_d)$, accuracy parameter ε, initial crude approximation $\mathbf{B}_0$ to $\mathbf{A}$

Output: TT-approximation $\mathbf{B}$ to the tensor

$\mathbf{B} := \mathbf{B}_0$, $R := 1$, $P_0 := 1$, $P_d := 1$
{Warmup: QR + right-to-left maxvol}
for $k = d$ to 1 step -1 **do**
  $B_k := B_k \times_3 R$
  $C := \mathtt{reshape}(B_k, [r_{k-1}, n_k r_k])$
  $[Q, R] := \mathtt{QR}_{\mathtt{rows}}(C)$
  $B_k := \mathtt{reshape}(Q, [r_{k-1}, n_k, r_k])$.
  {Compute reduced submatrix and new maxvol}
  $Q := \mathtt{reshape}(Q, [r_{k-1} n_k, r_k])$
  $Q := QP_k$, $Q = \mathtt{reshape}(Q, [r_{k-1}, n_k r_k])$
  $J_k := \mathtt{maxvol}_{\mathtt{rows}}(Q)$
  Compute $\mathcal{J}_k$ from $\mathcal{J}_{k+1}$ and J_k
  $P_{k-1} := Q(:, I_k)$
end for
`not_converged=.true.`
while `not_converged` **do**
  `not_converged=.false.`
  {Left-to-right sweep}
  for $k = 1$ to d **do**
    {Compute current subtensor and current supercore}
    $C := A(\mathcal{I}_{k-1}, i_k, i_{k+1}, \mathcal{J}_{k+1})$
    $C := \mathtt{reshape}(C, [r_{k-1}, n_k, n_{k+1}, r_{k+1}])$.
    $C := P_{k-1}^{-1} \times_1 C \times_4 P_{k+1}^{-1}$
    $C := \mathtt{reshape}(C, [r_{k-1} n_k, n_{k+1} r_{k+1}])$
    {Check internal convergence}
    if $\|C - B_k(i_k) B_{k+1}(i_{k+1})\|_F > \frac{\varepsilon}{\sqrt{d-1}} \|C\|_F$ **then**
      `not_converged=.true.`
    end if
    {Decimation step by ε-truncated SVD of C}
    $C =: USV$, $r_k := \mathtt{rank}_\varepsilon(C)$
    $B_k = \mathtt{reshape}(U, [r_{k-1}, n_k, r_k])$
    $B_{k+1} = \mathtt{reshape}(SV, [r_k, n_{k+1}, r_{k+1}])$
    {Recompute left maxvol and compute P_k}
    $U := P_{k-1} U$, $I_k = \mathtt{maxvol}(U)$, $P_k := U(I_k, :)$
    Recompute $\mathcal{I}_k$ from $\mathcal{I}_{k-1}$ and I_k.
  end for
  {Perform right-to-left sweep}
end while

6. Numerical experiments

6.1. Interpolation of one-dimensional functions

To demonstrate the behaviour of the QTT-cross algorithm we apply it for the approximation of the function-related vectors on the one-dimensional uniform grid, that are transfromed to tensors by the quantization of mode index. We consider a vector $y(i) = y(x_i)$ as a trace of a uniform function $y(x)$ on the grid $x_i = h(i + \frac{1}{2})$, $i = 0, \ldots, n-1$, $n = 2^d$ and $h = L/n$. Then we transform it to a d-tensor by a mapping

$$i \leftrightarrow (i_1, \ldots i_d), \quad i = \sum_{p=1}^{d} i_p 2^{p-1}, \quad i_p = 0, 1, \tag{10}$$

and apply the the QTT-cross to the resulted tensor

$$\mathbf{Y} = [y(i_1, \ldots, i_d)], \quad y(i_1, \ldots, i_d) = y(i) = y(x_i).$$

This idea appears in [20] in the context of matrix approximation. Then it was called the *QTT* format [17] and applied for some univariate functions with small QTT ranks. For the sine and cosine functions, the analytical QTT decomposition with ranks $r_0 = r_1 = \ldots = r_{d-1} = 2$ and $r_d = 1$ exists. Therefore, a tensor $\mathbf{Y}$ related to the sum of the two sine functions, $y(x) = \frac{1}{4}\sin x + \frac{3}{4}\sin 7x$, has QTT-ranks not larger than 4. For the sinc function $y(x) = \frac{\sin x}{x}$, the approximation with accuracy $\varepsilon = 10^{-2}$ has values of QTT ranks less than 3. In both of the experiments we start from the very crude initial guess $\mathbf{Y}_0 = \mathbf{1}$, with all entries being ones. However, the QTT-cross manages to recover the function in less than one full sweep, see Fig. 1. With accuracy $\varepsilon = 10^{-2}$, the QTT-cross uses 86 entries to interpolate the 'sine' tensor and 98 entries to interpolate 'sinc' tensor. The total number of elements in each tensor is 2^{10}.

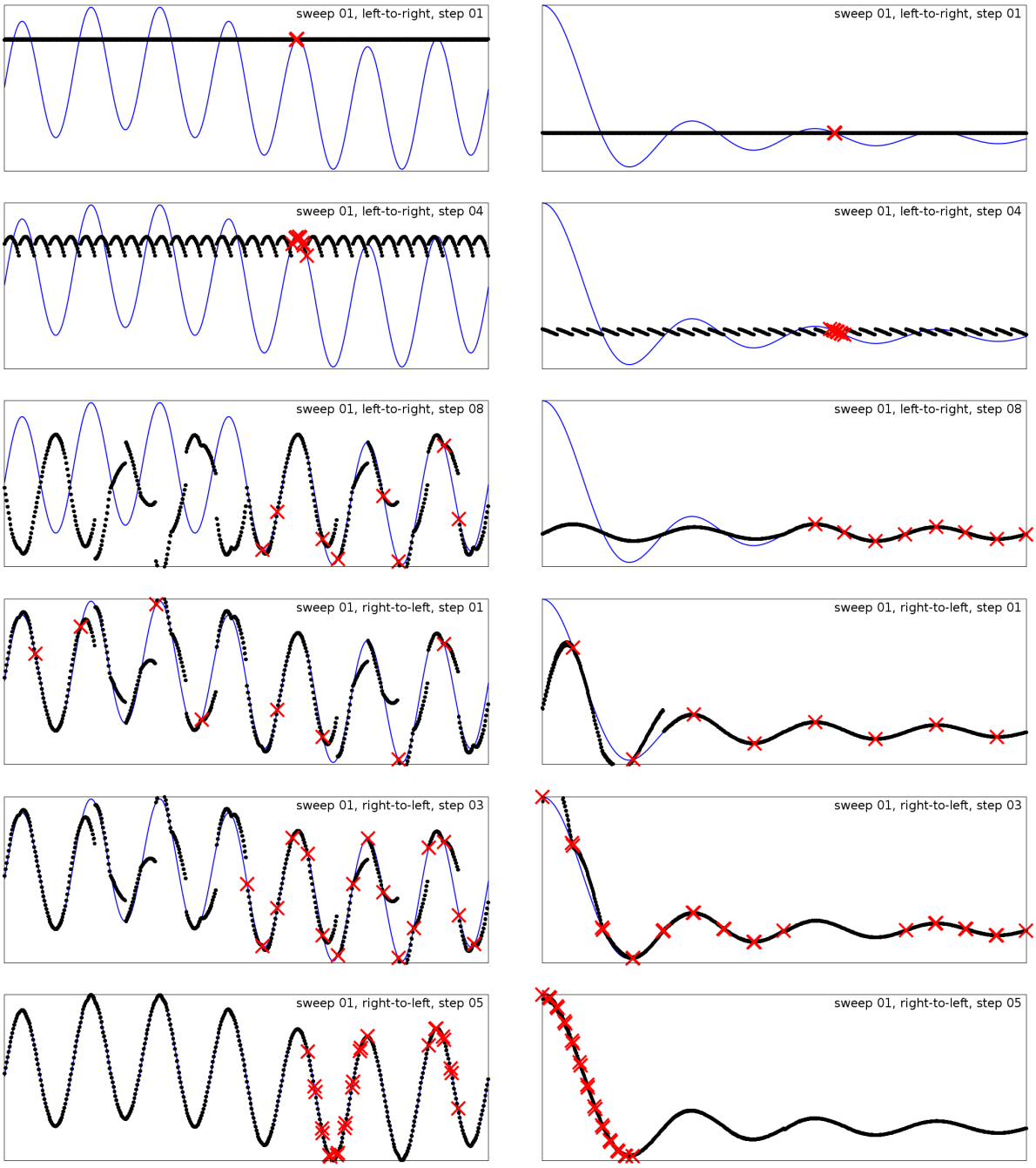
6.2. Approximation in many dimensions

We consider a m-dimensional Slater function

$$y(\mathbf{x}) = \frac{e^{-|\mathbf{x}|}}{|\mathbf{x}|}, \; |\mathbf{x}| = \sqrt{x_1^2 + \ldots + x_m^2}, \quad x \in [0 : L]^m$$

and discretize it on a uniform grid with $n = 2^d$ steps in each mode dinension. Using the quantization idea described in the previous paragraph, we come to tensor with $D = dm$ dimensions, each of size 2. Then we apply the QTT-cross algorithm for the approximation, starting from the tensor of all ones. The results are reported in Table 1. We see that QTT ranks of the Slater function grow fast with the physical dimension m but almost do not depend on the 'virtual' dimension d. Note also that for a fixed $m = 3$ (and for almost fixed QTT ranks), the total number of interpolation points on all steps of the QTT-cross grows linearly in d, but the number of different points among them (i.e. the number of tensor entries being evaluated) grows faster than linearly (for $\varepsilon = 10^{-5}$ almost quadratically). Since the time to compute one tensor element also grows linearly in d, the overall computation time grows faster than quadratically with d (for $\varepsilon = 10^{-5}$ almost cubically) for this example.

Figure 1. TT-RC applied to function $y(x) = \frac{1}{4}\sin x + \frac{3}{4}\sin 7x$, $x \in [0 : 2\pi]$ (left) and one-dimensional sinc function $\frac{\sin x}{x}$, $x \in [0 : 25]$ (right). Uniform grid with $n = 2^{10}$ points. Thin line — approximated function, thick line — approximation, X— interpolation points



Accuracy $\varepsilon = 10^{-3}$

m	d	N	sweeps	points	unique	rank	time, sec.
2	10	2^{20}	10	38684	1662	7	1.0
3	10	2^{30}	10	255512	34600	16	91
4	10	2^{40}	10	1197376	238582	34	9927

Accuracy $\varepsilon = 10^{-3}$

m	d	N	sweeps	points	unique	rank	time, sec.
3	10	2^{30}	5	93712	17637	16	63
3	12	2^{36}	5	122592	24302	16	97
3	14	2^{42}	5	148316	33626	16	141
3	16	2^{48}	5	172168	39530	16	222
3	18	2^{54}	5	196332	49512	16	290
3	20	2^{60}	5	201856	51104	16	356

Accuracy $\varepsilon = 10^{-5}$

m	d	N	sweeps	points	unique	rank	time, sec.
3	10	2^{30}	5	346936	74155	31	1126
3	12	2^{36}	5	450168	108908	31	1674
3	14	2^{42}	5	544348	133400	32	3770
3	16	2^{48}	5	630528	165589	32	5114
3	18	2^{54}	5	713112	194934	33	6026
3	20	2^{60}	5	789236	211809	33	8010

Table 1. TT-RC approximation for Slater function
Box size $L = 10$, grid size $n = 2^d$, total number of points $N = 2^{md}$. In tables, **points** is total number of interpolation points an all steps (with repetitions), **unique** is total number of different tensor entries being evaluated, **rank** is maximum QTT rank of the result, **time** is time for the whole computation.

References

[1] J. Ballani, L. Grasedyck, and M. Kluge, *Black box approximation of tensors in Hierarchical Tucker format*, Preprint 57, MPI MIS, Leipzig, 2010.

[2] M. Bebendorf, *Approximation of boundary element matrices*, Numerische Mathematik, 86 (2000), pp. 565–589.

[3] R. Bro, *PARAFAC: Tutorial and applications*, Chemometrics and Intelligent Lab. Syst., 38 (1997), pp. 149–171.

[4] J. D. Caroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

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

[6] ——, *On best rank-1 and rank-($R_1, R_2, ..., R_N$) approximation of high-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[7] ——, *Computing of canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 295–327.

[8] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.

[9] M. Espig, L. Grasedyck, and W. Hackbusch, *Black box low tensor rank approximation using fibre-crosses*, Constr. Appr., 30 (2009), pp. 557–597.

[10] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms, Applications, V. Olshevsky and E. Tyrtyshnikov, eds., World Scientific, Hackensack, NY, 2010, pp. 247–256.

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

[12] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.

[13] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[14] R. A. Harshman, *Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[15] J. Hastad, *Tensor rank is NP–complete*, Journal of Algorithms, 11 (1990), pp. 644–654.

[16] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys, 6 (1927), pp. 164–189.

[17] B. N. Khoromskij, $\mathcal{O}(d \log n)$–*Quantics approximation of* N–d *tensors in high-dimensional numerical modeling*, Constr. Appr., 34 (2011), pp. 257–280.

[18] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[19] I. V. Oseledets, *Compact matrix form of the* d*-dimensional tensor decomposition*, Preprint 2009-01, INM RAS, Moscow, 2009.

[20] ——, *Approximation of* $2^d \times 2^d$ *matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.

[21] ——, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[22] 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 (2008), pp. 939–956.

[23] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Fast simultaneous orthogonal reduction to triangular matrices*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 316–330.

[24] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.

[25] ——, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.

[26] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett., 75 (1995), pp. 3537–3540.

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

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

[29] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations*, Physical Review Letters, 91 (2003), p. 147902.

[30] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.