

TWO-LEVEL QTT-TUCKER FORMAT FOR OPTIMIZED TENSOR CALCULUS*

SERGEY DOLGOV† AND BORIS KHOROMSKIJ†

Abstract. We propose a combined tensor format, which encapsulates the benefits of Tucker, tensor train (TT), and quantized TT (QTT) formats. The structure is composed of subtensors in TT representations, so the approximation problem is proven to be stable. We describe all important algebraic and optimization operations, which are recast to the TT routines. Several examples on explicit function and operator representations are provided. The asymptotic storage complexity is at most cubic in the rank parameter, that is larger than for the global QTT approximation, but the numerical examples manifest that the ranks in the two-level format usually increase more slowly with the approximation accuracy than the QTT ones. In particular, we observe that high rank peaks, which usually occur in the TT/QTT representations, are significantly relaxed. Thus the reduced costs can be achieved.

Key words. tensor formats, QTT format, Tucker format, multilinear algebra, DMRG/ALS, higher dimensions, tensor methods, constructive tensor representations

AMS subject classifications. 65N22, 65F50, 15A69, 33F05, 65F10, 65F30, 65N35

DOI. 10.1137/120882597

1. Introduction. By a very general definition in mathematics, a tensor is an array of values in \mathbb{C} with several indices $A(i_1, \dots, i_d)$, varying in some range $i_k \in \{1, \dots, n_k\}$. If all $n_k \leq n$, one need at most n^d memory cells just to store all the elements. Such exponential growth with d is called the “curse of dimensionality” since [4]. To get rid of this problem, the natural idea is not to keep all the elements explicitly, but rather *approximate* them via some low-parametric structured representation, which itself requires much less data to be stored. If only a few tensor entries are nonzeros, i.e., a tensor is (pointwise) *sparse*, the simplest approach is to store and use in computations only these elements [40, 2].

Unfortunately, this is often not the case in practical problems (in particular, for solutions of multidimensional PDEs), and thus the development of *data-sparse* numerical techniques was motivated. Some approaches are based on the properties of discretization: the most important components, which can be represented with a low number of parameters, are selected and smartly combined to eliminate the errors as far as possible (sparse grids, ANOVA decomposition [6, 59, 42, 22], wavelet, and multiresolution analysis). Another direction is based on the idea of separation of variables from the functional analysis: the initial tensor is approximated with the sum of Kronecker products of one-dimensional vectors (the so-called *tensor formats*).

In this article, we consider only tensor formats. They are described by the dimension splitting in the initial tensor (obviously, we can enumerate the same data arbitrarily), and the summation structure of separable components. Even if the first

*Received by the editors June 27, 2012; accepted for publication (in revised form) by T. G. Kolda February 25, 2013; published electronically May 16, 2013. The first author was partially supported by RFBR grants 12-01-00546-a, 11-01-12137-ofi-m-2011, 11-01-00549-a, 12-01-33013, 12-01-31056, Russian Federation Government contracts II1112, 14.740.11.0345, 16.740.12.0727 at Institute of Numerical Mathematics, Russian Academy of Sciences.

<http://www.siam.org/journals/simax/34-2/88259.html>

†Max-Planck-Institut für Mathematik in den Naturwissenschaften, D-04103 Leipzig, Germany (sergey.v.dolgov@gmail.com, bokh@mis.mpg.de).

issue is more or less fixed by the “physical” model, the second one gives us much more freedom, with a totally different performance of numerical methods, based on a certain choice of a separation format.

In the history of tensor methods, different formats were proposed and analyzed, such as canonical [27, 26, 8], Tucker [58, 11, 12, 38, 37, 20], matrix product states (MPS) [60, 54, 53], and MCTDH [43] in the quantum physics community, as well as the MPS-type HT (hierarchical Tucker) [25] and TT (tensor train) [50, 47, 52] formats, in the numerical linear algebra community. The new generation of tensor formats is based on the QTT (quantized TT) [45, 35] approximation with logarithmic complexity scaling. Detailed discussions and comparisons of these formats can be found in the reviews [40, 36, 23], and [21]. As the combined approach, the Tucker-canonical format was applied to the solution of the Hartree–Fock equation [38, 37, 34]. Various combinations of the canonical, Tucker, and QTT formats have been recently described in [35, 36]. In terms of the asymptotic complexity in the mode size n and dimension parameter d , the best one is the QTT format, with the storage demand $\mathcal{O}(d \log_2(n) r^2)$. However, the important question is, how large is the tensor rank bound r of a given tensor, and even more important, of the tensors, arising in the solution process of a certain problem? In this way, the theoretical answer is available only for some special classes of functions and operators [35, 48, 39, 32, 37].

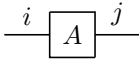
In this paper we show that *in practice* better results can be obtained with a smart combination of Tucker, TT, and QTT formats (further called *QTT-Tucker*), rather than only one of them. Namely, in many cases the Tucker and TT ranks are close. So the QTT approximation of the Tucker factors instead of the TT ones might require smaller ranks for the same accuracy and, moreover, avoid high rank peaks at the middle of a TT. The latter property can be expected from the cyclic tensor networks (say, the tensor chain), but approximation problems on them are usually unstable. The new QTT-Tucker format forms a closed manifold, as does the Tucker, TT, and HT formats; hence it allows stable solution of optimization problems. The corresponding efficient algorithms are described in this work.


The paper is organized as follows. In section 2 we introduce the notations used, and briefly review the existing tensor formats. In section 3, the new format, QTT-Tucker, is introduced, and its basic advantages are discussed. In section 4, we describe the TT-to-QTT-Tucker conversion, and provide the analytical examples, demonstrating the storage reduction possibilities of the new format applied to some discrete functions and operators. In sections 5 and 6, the basic linear algebra operations, rounding (rank compression), and optimization methods are presented and the error analysis and complexity are discussed, with the use of the two-level Tucker-TT structure and TT-structured subtensors. Finally, sections 7 and 8 provide the numerical experiments to check the performance, and the conclusion.

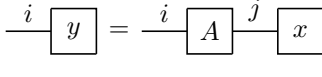
2. Overview of tensor formats.

2.1. Notation. Each element of a represented tensor is computed via sums over the rank indices. More complex tensor networks contain more ranks and sums. Writing them explicitly might be too long and complicated. Alternatively, we will use the graphical notations, proposed in the quantum chemistry community [60, 54, 29], later in [28], and used also in [16]. The idea is very simple: arrays are represented with rectangles (also called *blocks*), their indices are lines, and if a line connects two rectangles, it means that we multiply the elements of these arrays and perform the

summation over the connecting index. For example, the following figures denote

a matrix A with two indices i, j : 

a vector $[x_j]$: 

the matrix-vector product $y = Ax = \left[\sum_j A_{ij} x_j \right]_{i=1}^n$: 

Note that all blocks here are white, which means, that no restrictions are imposed. However, an important property we will need further is the *orthogonality* of blocks.

DEFINITION 2.1. An array $A(i_1, \dots, i_p, j_1, \dots, j_q)$ is said to be left orthogonal if

$$\sum_{i_1, \dots, i_p} \bar{A}(i_1, \dots, i_p, k_1, \dots, k_q) A(i_1, \dots, i_p, j_1, \dots, j_q) = \delta(k_1, j_1) \cdots \delta(k_q, j_q),$$

where \bar{A} means the complex conjugation, and $\delta(k, j) = 1$ if $k = j$, and zero otherwise.

In a matrix case ($p = q = 1$) it means the well-known matrix orthogonality $A^* A = I$.

DEFINITION 2.2. An array $A(i_1, \dots, i_p, j_1, \dots, j_q)$ is said to be right orthogonal, if

$$\sum_{j_1, \dots, j_q} \bar{A}(m_1, \dots, m_p, j_1, \dots, j_q) A(i_1, \dots, i_p, j_1, \dots, j_q) = \delta(m_1, i_1) \cdots \delta(m_p, i_p).$$

In the graphical notation, the left/right orthogonality looks as follows:



The filled part of a rectangle emphasizes the indices, which are being contracted, give the identity tensor.

The orthogonality property is heavily utilized in the approximation and solution tensor routines, since it allows us to control the introduced approximation error, and to make orthogonal projections (principal ingredient in the ALS/DMRG optimization); see [47, 28, 39, 49].

We would like also to define some of the array indices as parameters, to distinguish their purposes, and to be consistent with the previous notations in the MPS and TT communities. Here are, for example, three-dimensional (3D) array elements

$$G(\alpha, i, \beta) = G_{\alpha, i, \beta} = G_{\alpha, \beta}(i), \quad \alpha, \beta = 1, \dots, r, \quad i = 1, \dots, n,$$

which could be written in block form as

$$G(i) = \begin{bmatrix} G_{1,1}(i) & \cdots & G_{1,r}(i) \\ \vdots & G_{\alpha,\beta}(i) & \vdots \\ G_{r,1}(i) & \cdots & G_{r,r}(i) \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} G_{1,1}(1) \\ G_{1,1}(n) \end{pmatrix} & \cdots & \begin{pmatrix} G_{1,r}(1) \\ G_{1,r}(n) \end{pmatrix} \\ \vdots & & \vdots \\ \begin{pmatrix} G_{r,1}(1) \\ G_{r,1}(n) \end{pmatrix} & \cdots & \begin{pmatrix} G_{r,r}(1) \\ G_{r,r}(n) \end{pmatrix} \end{bmatrix}.$$

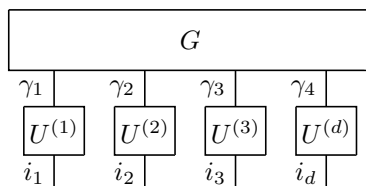
The latter notation is especially convenient for presenting analytical structures in the format, since all the elements of a block are written in the same equation.

We emphasize that the same capital letters denote the same arrays, and the indices can be freely distributed between brackets and subscripts, meaning certain permutation and reshaping. If a set of indices is written in brackets, all the rest of the indices are written in the subscript, $G_{\alpha,\beta}(i)$, or omitted, $G(i)$, $G_{::}(i)$, in the standard sense (substituted with “:” indices mean their full range, explicitly written yield certain elements, and the sense of omitted indices should be clear from the context).

If there are two or fewer subscripted indices, such an object is considered as a matrix, depending on indices in brackets as *parameters*, with row and column indices taken from the subscript. Any linear algebra operations (block concatenations, sums, scalar, Hadamard and contracted products, singular value decompositions, etc.) are then naturally defined.

2.2. Some established formats. One of the most traditional and widely used formats is the Tucker format [58, 11, 12]:

$$X(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} G(\gamma_1, \dots, \gamma_d) U_{\gamma_1}^{(1)}(i_1) \cdots U_{\gamma_d}^{(d)}(i_d), \quad \text{or graphically,}$$



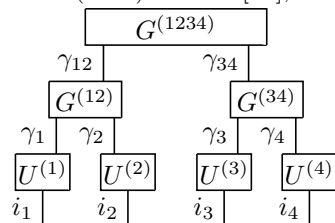
The *mode* indices i_1, \dots, i_d vary in ranges n_1, \dots, n_d , and the *rank* indices $\gamma_1, \dots, \gamma_d$ in ranges r_1, \dots, r_d . We see, that the storage of the *core* G still grows exponentially in d though, if $r \ll n$,¹ it is applicable to low-dimensional problems.

To deal with several tensors in one formula (additions, products, etc.), we may distinguish the parts of different decompositions as follows:

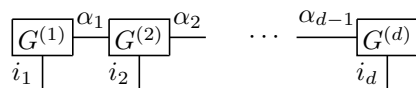
$$X(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} G_{[x]}(\gamma_1, \dots, \gamma_d) U_{[x] \gamma_1}^{(1)}(i_1) \cdots U_{[x] \gamma_d}^{(d)}(i_d),$$

$$Y(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} G_{[y]}(\gamma_1, \dots, \gamma_d) U_{[y] \gamma_1}^{(1)}(i_1) \cdots U_{[y] \gamma_d}^{(d)}(i_d).$$

To avoid intrinsic curse of dimensionality, some new formats were proposed. Among them are the hierarchical (Tree) Tucker [25], or ML-MCTDH [43],



and TT (MPS) [47, 54],



¹For brevity, we use uniform bounds in the complexity estimates: $n_k \leq n$, $r_k \leq r$, $k = 1, \dots, d$.

In terms of parameter-dependent matrices it is written as

$$(2.1) \quad X(i_1, \dots, i_d) = G^{(1)}(i_1) \cdots G^{(d)}(i_d), \quad G^{(k)}(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}.$$

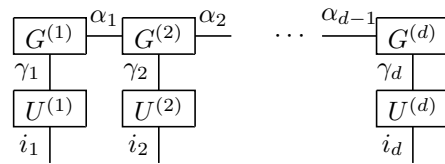
The *canonical* tensor format is just a sum of rank-1 components (or the TT with diagonal blocks, Tucker with a superdiagonal core),

$$X = \left[G_1^{(1)}(i_1) \cdots G_r^{(1)}(i_1) \right] \prod_{k=2}^{d-1} \begin{bmatrix} G_1^{(k)}(i_k) & & \\ & \ddots & \\ & & G_r^{(k)}(i_k) \end{bmatrix} \begin{bmatrix} G_1^{(d)}(i_d) \\ \vdots \\ G_r^{(d)}(i_d) \end{bmatrix}.$$

Despite the simplicity, this format suffers from a serious drawback: it is not a closed manifold, and the approximation problem is not stable.

3. Combining Tucker, TT, and QTT into the two-level structure.

3.1. Description of a new tensor network. The storage of the TT is $\mathcal{O}(dnr^2)$, but the storage of the Tucker factors is $\mathcal{O}(dnr)$. To exploit the best properties of these two formats, in [51] it was proposed to leave the Tucker factors “as is,” and compress into the TT only the core. This structure was named “extended TT decomposition.” Thus, the storage of such a representation is $\mathcal{O}(dnr + dr^3)$, i.e., asymptotically the same as in HT, but the structure is much simpler:



that is,

$$(3.1) \quad X(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} \left[G^{(1)}(\gamma_1) \cdots G^{(d)}(\gamma_d) \right] U_{\gamma_1}^{(1)}(i_1) \cdots U_{\gamma_d}^{(d)}(i_d).$$

Remark 1. The extended TT is closely related to the HT format: in both cases the physical modes are separated in the Tucker factors, and the difference is only in the storage of the core. The TT format for the core is a certain realization of the tree summation structure from HT. However, we consider only the TT variant due to its simplicity, and possibility of algorithm reduction to the TT subtensors. The generalization to an arbitrary tree topology can be done straightforwardly with technical changes of the TT algorithms.

Note, that the range of mode indices i_k can formally even be infinite, i.e., we store (say, as analytical formulas) r_k functions in each factor $U^{(k)}$. In the numerical calculus, the modes sizes should be finite, of course. However, we might like to have so many grid points (e.g., several millions), that even linear complexity in n is prohibitive.

To use all the benefits of tensor decompositions, we apply the QTT format [45, 35]. The construction of the *quantized* representation includes two steps. The first step is trivial and it reproduces the commonly used procedure in programming: each mode index i_k is further decomposed via the *binary* coding $i_k = 1 + \sum_{p=1}^L (i_{k,p} - 1) \cdot 2^{p-1}$, where $i_{k,p} \in \{1, 2\}$, so it varies in the minimal possible nontrivial range (“quant”). If the k th mode size is a power of 2, $n = 2^L$, such coding enumerates all and no more values of i_k . The initial tensor is simply reshaped to $2 \times \cdots \times 2$ (so its mode indices

in the quantized space are $i_{k,p}$). Now the main idea behind the QTT method is the TT approximation of the higher-dimensional quantized image. The computational potential of the QTT approach is justified by the surprisingly good approximation properties discovered for the class of function related arrays [35].

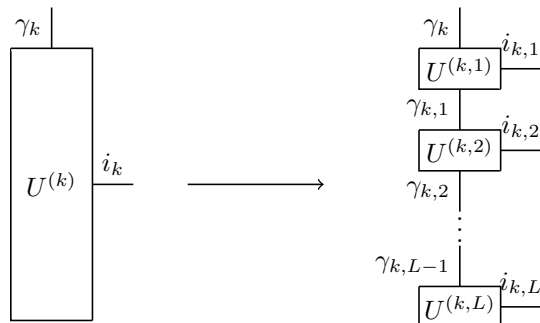
DEFINITION 3.1. *The TT representation (2.1) of the reshaped tensor $X(i_{1,1}, \dots, i_{d,L}) = X(i_1, \dots, i_d)$ with $i_k = 1 + \sum_{p=1}^L (i_{k,p} - 1) \cdot 2^{p-1}$, $k = 1, \dots, d$, is called the linear QTT format. Its TT ranks are called the linear QTT or simply QTT ranks, and are assumed to be bounded by r_Q .*

From the storage complexity $\mathcal{O}(dnr^2)$ of the TT format, we conclude immediately that if the QTT rank r_Q is bounded independently on the grid size, the storage of the QTT blocks is *logarithmic* in n , $\mathcal{O}(L \cdot 2 \cdot r_Q^2) = \mathcal{O}(\log(n)r_Q^2)$. The initial examples analyzed in [35] include various elementary functions on uniform grids, among them are:

- exponential $\exp(\alpha i) = \exp(\alpha \cdot i_1) \cdot \exp(\alpha \cdot 2i_2) \cdots \exp(\alpha \cdot 2^{L-1}i_L)$ with $r_Q = 1$,
- trigonometric $\sin(\alpha i)$, $\cos(\alpha i)$ of $r_Q = 2$,
- p -order polynomial of $r_Q = p + 1$.

Some important matrices, such as the Laplace operator and shift matrices are also of low QTT ranks; see [45, 32, 30] and the next section.

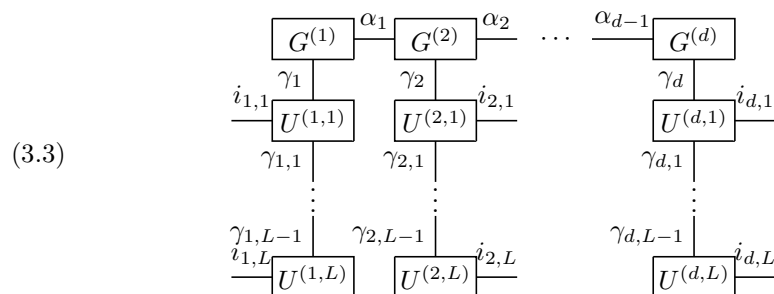
The Tucker factors can be considered as vectors generated by univariate functions, so we apply the QTT for them:



In the formula's notation

$$(3.2) \quad U_{\gamma_k}^{(k)}(i_k) = U_{\gamma_{k,1}}^{(k,1)}(i_{k,1}) \cdots U_{\gamma_{k,L-1}}^{(k,L)}(i_{k,L}).$$

The resulting network then looks as follows:



Notice that here we write the same L for all factors for brevity, but there can be, of course, different L_k 's. Analogously, each mode index $i_{k,p}$ can vary in some different range than $\{1, 2\}$. The formula's notation is more complicated: we have to rewrite

(3.1), plugging in (3.2) instead of $U_{\gamma_k}^{(k)}(i_k)$. That is why it is convenient to work in graphical notations with complex tensor structures.

In the following, to refer to this structure and its properties, we introduce our main definition.

DEFINITION 3.2. (*QTT-Tucker*). The tensor format with the rank summation structure (3.3) is called the QTT-Tucker format. The parameter d is called the physical, or core dimension, L is the quantics, or factor dimension, $G^{(k)}$ is the k th core block, $U^{(k,p)}$ is the k , p th factor block, and the indices vary in the ranges

- $i_{k,p} = 1, \dots, n_{k,p}, \quad n_{k,p} \leq n_Q$ (quantized mode size; mostly used $n_Q = 2$),
- $\alpha_k = 1, \dots, r_{C,k}, \quad r_{C,k} \leq r_C$ (core rank),
- $\gamma_k = 1, \dots, r_{T,k}, \quad r_{T,k} \leq r_T$ (Tucker rank), and
- $\gamma_{k,p} = 1, \dots, r_{F,k,p}, \quad r_{F,k,p} \leq r_{F,k} = r_Q(U^{(k)}) \leq r_F$ (factor rank).

The letters α, γ, i stand for the core rank, Tucker rank, and mode indices, respectively, throughout the paper.

Summing up the sizes of the QTT-Tucker blocks, we formulate the following

LEMMA 3.3. The storage complexity of the QTT-Tucker format is bounded by

$$(3.4) \quad dLn_Q r_F^2 + dr_T r_C^2 \quad \text{or} \quad \mathcal{O}(\log(n)dr^2 + dr^3),$$

assuming that all tensor ranks are bounded by r , and $n = n_Q^L$.

3.2. What can we benefit from?

3.2.1. Complexity issues. The $\mathcal{O}(r^3)$ contribution of the Tucker core to (3.4) may prevail in a high rank case, and if $r \sim r_Q$ for the same tensor, the memory cost of the new format is even larger than the linear QTT. For example, if we apply the HT format directly to the $2 \times \dots \times 2$ -reshaped tensor, it will contain $\mathcal{O}(d \log(n))$ blocks of sizes $r \times r \times r$, resulting in the total complexity of $\mathcal{O}(d \log(n)r + d \log(n)r^3)$ instead of $r \times 2 \times r$ in the linear QTT representation. The resulting serious complexity overhead was illustrated in [41] on the spin system example. Notice the difference with the QTT-Tucker; now we have only d blocks with the cubic storage in rank, whereas $d \log(n)$ blocks possess the quadratic dependence like in the linear QTT.

Performing the QTT approximation of the Tucker factors makes a certain sense in the approximation of tensors generated by smooth functions. We assume that a tensor X comes from the grid discretization of a multivariate function, and the TT and Tucker splitting is done according to the initial (“physical”) variables the function depends on. In this case, each TT block $\{G_{\alpha_{k-1}, \alpha_k}^{(k)}\}$ is in fact a set of $r_{C,k-1}r_{C,k}$ vectors of length n . Moreover, these vectors can be considered as a grid of univariate functions, which are assumed to possess some smoothness properties. As a result, we may expect a good QTT compressibility of each vector,

$$G_{\alpha_{k-1}, \alpha_k}^{(k)}(i_k) = G_{\alpha_{k-1}, :}^{(k,1)}(i_{k,1}) \cdots G_{:, \alpha_k}^{(k,L)}(i_{k,L}), \quad r_Q(G_{\alpha_{k-1}, \alpha_k}^{(k)}) \leq \tilde{r}_Q.$$

Note that this decomposition is performed for only one combination of α_{k-1}, α_k . To assemble the whole TT block $G^{(k)}$ one may need to sum the ranks:

$$(3.5) \quad r_Q(G^{(k)}) \leq r_{C,k-1}r_{C,k}r_Q(G_{\alpha_{k-1}, \alpha_k}^{(k)}) = \mathcal{O}(r_C^2 \tilde{r}_Q).$$

Similar considerations are applicable to the Tucker factors as well. Assume that each vector in a Tucker factor admits the QTT decomposition (3.2) with the rank bound \tilde{r}_Q for each fixed γ_k . Since the factor represents again the behavior of the

function in the k th variable, the rank \tilde{r}_Q is likely to be of the same order as in the TT case. However, the whole factor $U^{(k)}$ stores only $r_{T,k}$ such vectors. Hence,

$$(3.6) \quad r_Q(U^{(k)}) \leq r_{T,k} r_Q(U_{\gamma_k}^{(k)}) = \mathcal{O}(r_T \tilde{r}_Q).$$

The last question is the relation of r_T and r_C . A general estimate is rather pessimistic.

LEMMA 3.4. *If a tensor is represented in TT format with the irreducible rank bound r_C , and in the Tucker format with the rank bound r_T , the following relation holds:*

$$r_T \leq \min(r_C^2, n).$$

The constructive proof is presented in the next section. A similar result was also obtained during the discussion of HT and TT formats in [21]. What we need to note now, is that in the case $r_T \sim r_C^2$ no complexity reduction can be achieved, as follows from (3.5) and (3.6).

Fortunately, many practically relevant function-related tensors have r_T and r_C of the same order of magnitude. Armed with this assumption, we obtain the following memory cost from (3.4):

$$\begin{aligned} \text{mem}(\text{linear QTT}) &= \mathcal{O}(d \log(n) \tilde{r}_Q^2 r_C^4), \\ \text{mem}(\text{QTT-Tucker}) &= \mathcal{O}(d \log(n) \tilde{r}_Q^2 r_T^2 + d r_C^2 r_T). \end{aligned}$$

It could give a significant reduction of complexity, as observed in the numerical experiments below. In the following, unless explicitly specified, we assume $r_T \sim r_C$, e.g., the storage becomes $\mathcal{O}(d \log(n) \tilde{r}_Q^2 r_C^2 + d r_C^3)$. It is usually the case in practical problems with smooth functions.

3.2.2. Theoretical rank estimates. Another interesting feature of the new format, is that it may inherit the rank estimates from the Tucker format, which may be easier to establish than in the TT format. Such arguments have been already used for the rank estimates in [33]. To illustrate this, we formulate the following theorem.

THEOREM 3.5. *Assume that a function $f(x_1, \dots, x_d)$ in $\Omega = [-1, 1]^d$ is given, and a tensor X is its discretization on a tensor product uniform grid. Suppose f admits an analytical extension to the ρ_k -Bernstein ellipse $\mathcal{E}_{\rho_k} = \{z \in \mathbb{C} : |1+z| + |1-z| \leq \rho_k + \frac{1}{\rho_k}\}$ in each variable x_k . Then the Tucker ranks of the ε -approximation are bounded as $r_{T,k} \leq C |\log(\varepsilon)| / \log(\rho_k)$, and the QTT ranks of the Tucker factors are $r_{F,k} = r_{T,k} + 1$.*

Proof. The function f may be considered as a univariate function $f_{x \setminus x_k}^{(k)}(x_k)$ of a variable x_k , depending on the rest coordinates as parameters. So we can apply the following result from the approximation theory: if $f^{(k)}$ admits an analytical extension to the ρ_k -Bernstein ellipse, then there exists the polynomial interpolation P_p of degree p and the accuracy

$$\|f^{(k)}(z) - P_p(z)\|_\infty \leq C \log(n) \frac{M}{1 - \rho_k} \rho_k^{-p}, \quad z \in \mathcal{E}_{\rho_k}, \quad \rho_k > 1, \quad M = \max_{z \in \mathcal{E}_{\rho_k}} |f(z)|,$$

where n is the number of grid points, and C does not depend on p, n, M, ρ_k [5, 57]. However, here the constant M depends on the other variables x_1, \dots, x_d without x_k . To get rid of them, let us assume the uniform bound

$$\mathcal{M} = \max_{z \in \mathcal{E}_{\rho_1} \otimes \dots \otimes \mathcal{E}_{\rho_d}} |f(z)| < \infty.$$

Now we may say that for each $f^{(k)}$ there exists a polynomial $P_{p_k}^{(k)}$ such that

$$\varepsilon = \|f^{(k)}(x_k) - P_{p_k}^{(k)}(x_k)\|_\infty \leq C\rho_k^{-p_k}, \quad \text{so that} \quad p_k = C|\log(\varepsilon)|/\log(\rho_k).$$

Therefore, we may take a tensor product of degree p_k polynomials as the new basis. Obviously, the coefficients in this basis yield a $p_1 \times \cdots \times p_d$ tensor, which may be considered as a Tucker core, and a set of p_k polynomials on a grid is the k th Tucker factor. The estimate for r_T is proven.

Recalling to [35, 48] that a polynomial of degree p_k on a uniform grid is representable with the QTT ranks $p_k + 1$ (and in fact any lower degree polynomial fits in the same QTT representation, by changing the last block coefficients only), we come to the second claim of the theorem. \square

4. Explicit representations.

4.1. TT to extended TT (TT-Tucker) conversion. During the exploitation of various tensor formats, it is helpful to construct some simple and widely used objects explicitly, by specifying the format elements directly. For the QTT format this work started from [48, 32]. For example, consider the Laplace-like sum

$$A(i_1, \dots, i_d) = x(i_1) \cdot y(i_2) \cdots y(i_d) + \cdots + y(i_1) \cdots y(i_{d-1}) \cdot x(i_d), \quad \text{or} \\ A = x \otimes y \otimes \cdots \otimes y + \cdots + y \otimes \cdots \otimes y \otimes x, \quad x, y \in \mathbb{R}^n.$$

In this work, we follow the notation from [48], where each univariate term $x(i), y(i)$ is a scalar-valued function of an abstract argument i . Now, it is consistent to write tensor product structures using simple matrix products, letting the arguments i_1, \dots, i_d take arbitrary, but fixed, values.

The *exact* rank-2 TT representation of A reads (the indices i_k are omitted for brevity)

$$A = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} y & 0 \\ x & y \end{bmatrix}^{d-2} \begin{bmatrix} y \\ x \end{bmatrix},$$

which is obtained simply by successively splitting linearly independent elements of the leading dimension into the current block, i.e., reconstructing the TT-SVD algorithm [50, 47] analytically. The Tucker decomposition can also be obtained by similar reconstruction of higher-order SVD (HOSVD) [11]. However, here we also present the transformation of the TT to extended TT.

This will be done in parallel over the TT blocks. Consider the k th TT block,

$$\tilde{G}^{(k)}(i_k) = \begin{bmatrix} \tilde{G}_{1,1}^{(k)}(i_k) & \cdots & \tilde{G}_{1,r_k}^{(k)}(i_k) \\ \vdots & \tilde{G}_{\alpha_{k-1},\alpha_k}^{(k)}(i_k) & \vdots \\ \tilde{G}_{r_{k-1},1}^{(k)}(i_k) & \cdots & \tilde{G}_{r_{k-1},r_k}^{(k)}(i_k) \end{bmatrix}.$$

Find the minimal basis set $\{U_{\gamma_k}^{(k)}(i_k)\}$, such that

$$\tilde{G}_{\alpha_{k-1},\alpha_k}^{(k)}(i_k) = \sum_{\gamma_k=1}^{r_{T,k}} G_{\alpha_{k-1},\alpha_k}^{(k)}(\gamma_k) U_{\gamma_k}^{(k)}(i_k).$$

That is, $\tilde{G}^{(k)}$ can be represented as

$$\begin{array}{c}
 \alpha_{k-1} \left[\begin{array}{ccc} G_{1,1}^{(k)}(\gamma_k) & \cdots & G_{1,r_k}^{(k)}(\gamma_k) \\ \vdots & G_{\alpha_{k-1},\alpha_k}^{(k)}(\gamma_k) & \vdots \\ G_{r_{k-1},1}^{(k)}(\gamma_k) & \cdots & G_{r_{k-1},r_k}^{(k)}(\gamma_k) \end{array} \right] \alpha_k \\
 \gamma_k \downarrow \\
 \left[U_1^{(k)}(i_k) \cdots U_{r_{T,k}}^{(k)}(i_k) \right] i_k
 \end{array}$$

where $G_{\alpha_{k-1},\alpha_k}^{(k)}(\gamma_k)$ are the coefficients of $\tilde{G}_{\alpha_{k-1},\alpha_k}^{(k)}(i_k)$ in the basis $U_{\gamma_k}^{(k)}(i_k)$. Obviously, the maximal number of linearly independent elements in the k th block is not greater than $r_{k-1}r_k$, as well as the mode size n_k . This constructive algorithm confirms Lemma 3.4.

Remark 2. The extraction of a Tucker factor from a TT block can also be done approximately via the SVD, by selecting a *truncated* set of *principal* components to ensure a certain accuracy.

The first Laplace-like example in this section contains only 2 linearly independent elements in each block, so its decomposition reads (we omit i_2, \dots, i_{d-1} for brevity)

$$(4.1) \quad \begin{array}{c} \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right] \alpha_1 \left[\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right]^{d-2} \alpha_{d-1} \left[\begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]^T \\ \gamma_1 \downarrow \quad \gamma_2 \dots \gamma_{d-1} \downarrow \quad \gamma_d \downarrow \\ i_1 \left[\begin{array}{|c|} \hline x \quad y \\ \hline \end{array} \right] \quad \left[\begin{array}{|c|} \hline x \quad y \\ \hline \end{array} \right] \quad \left[\begin{array}{|c|} \hline x \quad y \\ \hline \end{array} \right] i_d \end{array}$$

Remarkably, we obtain all the same Tucker factors, and the core contains only zeros and ones to combine the factors correctly.

Remark 3. Other tensor networks can be converted first to the TT format (or a sum of TT tensors, for example, the tensor chain), and then into the QTT-Tucker. It could be helpful in quantum physical problems, if a model provides initial data in a complicated tensor network.

4.2. Operator example. Now, derive the quantics representation for $[x, y] = [\Delta_L, I_L]$, where Δ_L is the finite difference Laplacian with Dirichlet boundary conditions on a grid with 2^L nodes. In [32] its explicit QTT representation was proven.² What is interesting, the full identity matrix I_L can be represented using the *same* blocks as Δ_L except the first:

$$\begin{array}{c}
 \begin{bmatrix} \Delta_L \\ I_L \end{bmatrix} = \gamma \left[\begin{array}{cc} (\Delta_1) & (-J) & (-J') \\ (I_1) & & \end{array} \right] \gamma_1 \left[\begin{array}{cc} (I_1) & (J) \\ (J') & (J') \end{array} \right]^{L-2} \gamma_{L-1} \left[\begin{array}{c} (I_1) \\ (J') \\ (J) \end{array} \right] \\
 i_1 \downarrow \quad j_1 \downarrow \quad i_2 \dots i_{L-1} \downarrow \quad j_2 \dots j_{L-1} \downarrow \quad i_L \downarrow \quad j_L \downarrow
 \end{array}$$

where Δ_1 is a 2×2 Laplacian matrix, I_1 is the identity of size 2, and $J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ is a Jordan block. Note that in the blocks $2, \dots, L$, the only nonzero element in the first

²Here we write the blocks, corresponding to lower bits, on the left, i.e., with the MATLAB and Fortran *little-endian* indexing, whereas in [32] they are written on the right.

row is I_1 at the first place, which admits the simultaneous QTT representation of Δ_L and I_L .

Plugging this decomposition into (4.1), we obtain the QTT-Tucker representation of the Laplace operator.

LEMMA 4.1. *The finite difference Laplace operator on a uniform $2^L \times \dots \times 2^L$ grid with Dirichlet boundary conditions is exactly representable in the QTT-Tucker format with the ranks*

$$r_F = 3, \quad r_T = 2, \quad r_C = 2.$$

The linear QTT representation of the same operator possesses the rank bound 4 [32].

Notice that the crucial point here is to put the Tucker index γ into the *lowest* bit block. If we connect γ to the senior bit, the rank has to be 4. In the linear QTT, the TT indices appear at both ends of inner blocks. That is why the linear QTT decomposition of the multidimensional Laplacian possesses the rank 4 for all TT blocks except the last. This simplest example confirms the considerations in the previous section: instead of two TT rank indices in the linear QTT structure, the QTT-Tucker format requires only one Tucker rank index to connect the QTT representation of one-dimensional objects, thus admitting smaller QTT ranks.

4.3. Bilinear form example. In many applications we are to compute the following bilinear form:

$$(4.2) \quad V(X', X) = \sum_{p,q=1}^d B_{pq} X'_p \otimes X_q,$$

where X'_p, X_q are the following rank-1 tensor product objects:

$$X'_p = e_1 \otimes e_2 \otimes \dots \otimes x'_p \otimes \dots \otimes e_d, \quad X_q = e_1 \otimes e_2 \otimes \dots \otimes x_q \otimes \dots \otimes e_d, \\ e_k, x_k, x'_k \in V \sim \mathbb{R}^n, \text{ a certain Euclidean vector space,}$$

$\otimes : V \times V \rightarrow V$ is a multiplicative bilinear operation, distributive with “ \otimes ” (e.g., Hadamard or matrix product), e_k are the unities with respect to \otimes , and B is a matrix of scalars. Note that the mode indices are omitted in this section for brevity.

The quadratic *Lyapunov* function³ is such a form, with e_k being vectors of all ones, $x_k = x'_k$ are the vectors of grid points, and \otimes is the Hadamard product. For a general polynomial, it was investigated in [39], where the rank bound $d+2$ was proven for the polynomial degree 2. Here, we give a refined result. An explicit construction of a low-rank representation of V , provided that the basis elements $e_k, x_k, x'_k, (x'_k \otimes x_k) \equiv x_k^2$ are defined, gives the following theorem.

THEOREM 4.2. *Introduce the off-diagonal lower $C^k = B_{k+1:d, 1:k}$ and upper $\hat{C}^k = B_{1:k, k+1:d}$ submatrices. The bilinear form (4.2) possesses an exact TT decomposition $V = V_1 \dots V_d$ with the ranks:*

$$(4.3) \quad r_{TT,k} = \text{rank}(C^k) + \text{rank}(\hat{C}^k) + 2 \leq d + 2.$$

Moreover, if $x_k = x'_k$ (symmetric case), the ranks reduce to

$$r_{TT,k} = \text{rank}\left((C^{k\top} + \hat{C}^k)/2\right) + 2 \leq d/2 + 2.$$

³For example, it arises in the calculation of the typical d -dimensional probability density $\psi = \exp(-V)$.

In the QTT-Tucker format, the estimates above correspond to the TT ranks of the core, $r_{C,k} = r_{TT,k}$. The Tucker ranks $r_{T,k}$ equal 4, in the general case, and 3 in the symmetric case. The QTT ranks of the Tucker factors depend on e_k, x_k, x'_k, x_k^2 :

$$r_{F,k} \leq r_Q(e_k) + r_Q(x_k) + r_Q(x'_k) + r_Q(x_k^2).$$

The linear QTT rank bound is $r_Q(e_k) + \text{rank}(C^k)r_Q(x_k) + \text{rank}(\hat{C}^k)r_Q(x'_k) + r_Q(x_k^2)$.

The proof is rather technical and is written in the appendix.

In the case of a degree 2 polynomial on a uniform grid, the QTT ranks of the Tucker factors are equal to 3 [48, 35].

Remark 4. The ranks of the off-diagonal blocks are the so-called *quasiseparable* ranks. Theorem 4.2 establishes a connection between the matrix quasiseparability and the TT structure. This rank estimate is sharp: if the matrix is diagonal, i.e., $C^{k\top} = \hat{C}^k = 0$, then the TT ranks equal 2 (Laplace operator, harmonic oscillator potential).

Remark 5. The estimate (4.3) is applicable to the d -dimensional anisotropic elliptic operator $\sum_{p,q} \nabla_p^\top B_{p,q} \nabla_q$ with a constant matrix B , by setting $e_k = I_k$, $x_k = \nabla_k$, $x'_k = \nabla_k^\top$, $x_k^2 = \Delta_k$, and \circledast being the operator composition. A discussion on such structures was started in [32, 15].

Note that similar results on the explicit TT/QTT representation of the discrete multidimensional diffusion operator with variable coefficients, and the sharp rank estimates in terms of ranks of the accompanying quasiseparable coefficient matrix have been independently published in [31]. This preprint was discovered during the revision process of our manuscript, and contains a more thorough study of the issue. However, our preprint [14] was published earlier,⁴ and moreover, Theorem 4.2 is applied in the nontrivial numerical example for the Fokker–Planck equation; see section 7.2.

5. Fast QTT-Tucker arithmetics. Very basic operations worth having with a format, are the linear algebra operations: linear combinations, matrix products, etc. For all tensor formats, they are easily derived, using the following rules:

- separable components are added, or multiplied independently in all variables;
- all rank sums (and hence, the ranks) are added in linear operations, and multiplied in bilinear ones.

During the iterative methods, the rank may increase rapidly due to multiplications. Hence, another important operation, which should be provided with the format, is the rank truncation.

The QTT-Tucker format consists of several connected TTs. So, it will be convenient in the following to split the description of a certain procedure to the underlying TTs, for which it is known.

5.1. Linear combination, matrix, and Hadamard products. We recall that the QTT-Tucker contains “physical” modes (i.e., the modes, on which the initial operation is defined) only in factors. Therefore, we can recast the operations to the factors, describing only the proper combination of cores.

- *Linear combination* $Z = aX + bY$: TT linear combination of factors [47]:

$$U_{[Z]}^{(1,1)} = \begin{bmatrix} aU_{[X]}^{(1,1)}(i_{1,1}) & \\ & bU_{[Y]}^{(1,1)}(i_{1,1}) \end{bmatrix}, \quad U_{[Z]}^{(k,p)} = \begin{bmatrix} U_{[X]}^{(k,p)}(i_{k,p}) & \\ & U_{[Y]}^{(k,p)}(i_{k,p}) \end{bmatrix},$$

⁴See <http://www.mis.mpg.de/publications/preprints/2012/prepr2012-19.html>

$k, p \neq 1$; block-diagonal concatenation of core blocks

$$G_{[Z]}^{(k)} = \begin{bmatrix} G_{[X]}^{(k)}(\gamma_k^x) & \\ & G_{[Y]}^{(k)}(\gamma_k^y) \end{bmatrix}.$$

All ranks are summed: $r^z = r^x + r^y$. Remember that we distinguish the format blocks of different tensors by $G_{[x]}^{(k)}$, $U_{[y]}^{(k)}$, and so on.

- *Matrix-by-matrix(vector)* product $y = Ax$: TT multiplication of factors:

$$U_{[y]}^{(k,p)}(i_{k,p}) = \sum_{j_{k,p}} U_{[A]}^{(k,p)}(i_{k,p}, j_{k,p}) \otimes U_{[x]}^{(k,p)}(j_{k,p});$$

3-way Kronecker product of core blocks

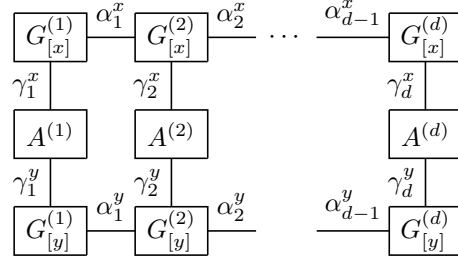
$$G_{[y]}^{(k)} = G_{[A]}^{(k)} \otimes G_{[x]}^{(k)}.$$

All ranks are multiplied: $r^y = r^A \cdot r^x$.

- *Hadamard* product is reducible to the matrix product: $z = x \odot y = \text{diag}(x)y$, and $\text{diag}(x)$ is constructed by taking diag of each factor without changing the ranks.
- *Dot* product $a = (x, y)$ requires more detailed description. First, we perform the TT dot products over factors:

$$\begin{aligned} A^{(k)} &= \left(U_{[x]}^{(k)}, U_{[y]}^{(k)} \right) \\ &= \left(\sum_{i_{k,1}} U_{[x]}^{(k,1)}(i_{k,1}) \otimes U_{[y]}^{(k,1)}(i_{k,1}) \right) \cdots \left(\sum_{i_{k,L}} U_{[x]}^{(k,L)}(i_{k,L}) \otimes U_{[y]}^{(k,L)}(i_{k,L}) \right). \end{aligned}$$

It is the matrix of size $r_T^x \times r_T^y$, since it contains all combinations of one-dimensional vectors in the Tucker factors. Now, compute the following sum:



Contracting the matrices $A^{(k)}$ with one of the cores, and taking their TT dot product, we obtain the result.

5.2. QTT-Tucker rounding. The algorithm of the rounding procedure should be written carefully to make the method stable. In [47, 11] it was shown that the only way to control the error introduced to some part of the format, is to impose the proper orthogonality on the other connected parts. We can reduce the QTT-Tucker rounding procedure to the structured variants of HOSVD and TT rounding. For the definitions and descriptions of the TT orthogonalization (ort_{TT}) and TT rounding (round_{TT}) we refer to [47].

To consider the factor and core blocks in a uniform way, we proceed as follows. Denote formally the core rank indices α_{k-1}, α_k as the new mode index of $G^{(k)}$, and γ_k as the rank index, so that $G_{\alpha_{k-1}, \alpha_k}^{(k)}(\gamma_k) \rightarrow G_{\gamma_k}^{(k)}(\alpha_{k-1}, \alpha_k)$ with the same elements,

but now with the rank index γ_k consistent with the QTT representation of the factor. Now, assemble an $(L+1)$ -dimensional TT tensor, by connecting the factor with such a reshaped core block.

DEFINITION 5.1. *Given a tensor X in the QTT-Tucker format. The TT tensor*

$$E_{[x]}^{(k)}(i_k, \alpha_{k-1}, \alpha_k) = U_{[x]}^{(k,L)}(i_{k,L}) \cdots U_{[x]}^{(k,1)}(i_{k,1}) G_{[x]}^{(k)}(\alpha_{k-1}, \alpha_k), \quad k = 1, \dots, d,$$

with the last block obtained from the core block by the permutation of indices $\alpha_{k-1}, \gamma_k, \alpha_k \rightarrow \gamma_k, (\alpha_{k-1}, \alpha_k)$, is called the k th extended factor of size $n_k \times r_{C,k-1} \times r_{C,k}$.

Now, the Tucker ranks become additional factor ranks, and we can update them all in a single call to the TT procedure on the extended factor. However, certain orthogonalizations have to be written explicitly due to the TT structure of the core. The whole method is summarized in Algorithm 1.

ALGORITHM 1. QTT-TUCKER ROUND.

Require: QTT-Tucker tensor X , accuracy bounds $\varepsilon_{k,p}$, or rank bounds $R_{k,p}$.

Ensure: QTT-Tucker tensor Y with possibly smaller ranks, and $\|Y - X\|^2 \leq \sum \varepsilon_{k,p}^2$,
or $r_{k,p} \leq R_{k,p}$.

1: **for** $k = 1, \dots, d$ **do**

2: Left-orthogonalize the extended factor:

$$E_{[y]}^{(k)}(i_k, \alpha_{k-1}, \alpha_k) = \text{ort}_{\text{TT}}(E_{[x]}^{(k)}(i_k, \alpha_{k-1}, \alpha_k)).$$

3: **end for**

4: Round the core $G_{[y]} = \text{round}_{\text{TT}}(G_{[y]}, \varepsilon_k, R_k)$, such that $G_{[y]}$ is left-orthogonal.

5: **for** $k = d, \dots, 2$ **do**

6: Round the extended factor:

$$E_{[y]}^{(k)}(i_k, \alpha_{k-1}, \alpha_k) = \text{round}_{\text{TT}}(E_{[y]}^{(k)}(i_k, \alpha_{k-1}, \alpha_k), \varepsilon_{k,p}, R_{k,p})$$

such that it is left-orthogonal.

7: Right-orthogonalize the core block $G_{[y]}^{(k)}$:

$$G_{[y]}^{(k)}(\gamma_k) = R Q_{[y]}^{(k)}(\gamma_k), \quad G_{[y]}^{(k)} = Q_{[y]}^{(k)}, \quad G_{[y]}^{(k-1)}(\gamma_{k-1}) = G_{[y]}^{(k-1)}(\gamma_{k-1}) R.$$

8: **end for**

9: Round the first extended factor:

$$E_{[y]}^{(1)}(i_1, \alpha_1) = \text{round}_{\text{TT}}(E_{[y]}^{(1)}(i_1, \alpha_1), \varepsilon_{1,p}, R_{1,p}).$$

The complexity of Algorithm 1 comes directly from the complexities of the TT routines. Namely, the orthogonalization/truncation requires $\mathcal{O}(dLn_Q r_F^3)$ operations for factors and $\mathcal{O}(dr_C^4)$ for the core, resulting in total cost $\mathcal{O}(dLn_Q r_F^3 + dr_C^4)$.

It is also interesting that the orthogonalization/rounding of the extended factors (lines 2, 6, and 9 of the algorithm) in fact mimics the Tucker HOSVD: the difference is in using only one *block of the core* instead of the whole core as the full array. Lines 4 and 7 take into account the TT structure of the core. The states of the QTT-Tucker tensor after certain lines of Algorithm 1 are shown in Figure 5.1 (the sizes of the blocks are proportional to their ranks).

To finish with rounding, we provide the error analysis.

THEOREM 5.2. *Suppose that each truncation in QTT-Tucker factors is done via the SVD with the Frobenius error $\varepsilon_{k,p}$, $p = 2, \dots, L$, the Tucker ranks (ranges of γ_k in extended factors) are determined with $\varepsilon_{k,1}$, and each core block is truncated with $\varepsilon_{k,0}$. Then, the Frobenius error in the whole tensor estimates as*

$$(5.1) \quad \|Y - X\|^2 \leq \sum_{(k,p)=(1,2)}^{d,L} \varepsilon_{k,p}^2 + \sum_{k=1}^d \varepsilon_{k,1}^2 + \sum_{k=1}^{d-1} \varepsilon_{k,0}^2.$$

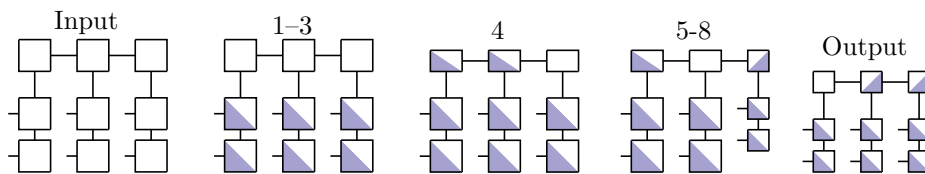


FIG. 5.1. QTT-Tucker rounding steps in graphical notation.

Proof. We will utilize the fact that each SVD truncation performs the orthogonal projection onto the subspace spanned by the senior singular vectors. In the two-dimensional case it reads

$$X = \sum_{\alpha=1}^R U_{\alpha} \sigma_{\alpha} \otimes V_{\alpha}, \quad Y = \sum_{\alpha=1}^r U_{\alpha} \sigma_{\alpha} \otimes V_{\alpha} = (\tilde{U} \otimes I) (\tilde{U} \otimes I)^{\top} X, \quad \tilde{U} = \{U_{\alpha}\}_{\alpha=1}^r;$$

that is, $Y = PX$, $P = (\tilde{U} \otimes I)(\tilde{U} \otimes I)^{\top}$, or $P = (I \otimes \tilde{V})(I \otimes \tilde{V})^{\top}$. The error writes $\|Y - X\| = \|(I - P)X\| \leq \varepsilon = \sqrt{\sigma_{r+1}^2 + \dots + \sigma_R^2}$.

In the QTT-Tucker format, we compute the SVD for each block in factors and core. So, for each $k = 1, \dots, d$, $p = 1, \dots, L$ introduce the factor block projectors

$$P^{(k,p)} = \left(\bigotimes_{m=1}^{k-1} I_n \right) \otimes \left(\bigotimes_{l=1}^{p-1} I_{n_Q} \otimes (\mathcal{V}^{(k,p)} \mathcal{V}^{(k,p)\top}) \right) \otimes \left(\bigotimes_{m=k+1}^d I_n \right),$$

where $\mathcal{V}^{(k,p)}(i_{k,p}, \dots, i_{k,L}) = \tilde{V}^{(k,p)}(i_{k,p}) \dots \tilde{V}^{(k,L)}(i_{k,L})$, $\tilde{V}^{(k,p)}$ are the senior singular vectors of the (k, p) th factor block, and I_s is the identity matrix of size s , and core block projectors

$$Q^{(k)} = \left(\bigotimes_{m=1}^d \mathcal{V}^{(m,1)} \right) \left((\mathcal{U}^{(k)} \mathcal{U}^{(k)\top}) \otimes \bigotimes_{m=k+1}^d I_{r_T} \right),$$

where $\mathcal{U}^{(k)}(\gamma_1, \dots, \gamma_k) = \tilde{U}^{(1)}(\gamma_1) \dots \tilde{U}^{(k)}(\gamma_k)$, and $\tilde{U}^{(m)}$ are the senior singular vectors of the k th core block. In other words, $\mathcal{V}^{(k,p)}$ are the right singular vectors of the p th unfolding of the k th factor, and similarly $\mathcal{U}^{(k)}$ are the left singular vectors of the k th unfolding of the core.

In this framework, the approximation reads

$$Y = \prod_{k,p} P^{(k,p)} \prod_k Q^{(k)} X.$$

For the error, we obtain the following expansion:

$$\begin{aligned} \|X - Y\|^2 &= \|X - P^{(1,1)}X + P^{(1,1)}X - Y\|^2 \\ &= \left\| (I - P^{(1,1)})X + P^{(1,1)} \left(X - \prod_{(k,p) \neq (1,1)} P^{(k,p)} \prod_k Q^{(k)} X \right) \right\|^2 \\ &= \|(I - P^{(1,1)})X\|^2 + \left\| P^{(1,1)} \left(X - \prod_{(k,p) \neq (1,1)} P^{(k,p)} \prod_k Q^{(k)} X \right) \right\|^2. \end{aligned}$$

The last equality is due to the orthogonality of $I - P^{(1,1)}$ and $P^{(1,1)}$. Now, the first term

stands for the first truncation, i.e., $\|(I - P^{(1,1)})X\|^2 \leq \varepsilon_{1,1}^2$. For the second it holds

$$\left\| P^{(1,1)} \left(X - \prod_{(k,p) \neq (1,1)} P^{(k,p)} \prod_k Q^{(k)} X \right) \right\|^2 \leq \left\| X - \prod_{(k,p) \neq (1,1)} P^{(k,p)} \prod_k Q^{(k)} X \right\|^2,$$

and we can proceed by a recursion, splitting $\varepsilon_{1,2}^2$ and so on. Note that the order of $P^{(k,p)}$ and $Q^{(k)}$ does not matter, since the only nonidentities in different projectors stay in different places. For example, we separate first all $P^{(k,p)}$, and obtain $\|X - \prod_k Q^{(k)} X\|^2$, which gives the third term in (5.1), and finishes the proof. \square

COROLLARY 5.3. *If for all blocks the truncation error is fixed to the same value ε , the total accumulated error estimates as*

$$\|Y - X\| \leq \sqrt{dL + d - 1} \varepsilon.$$

In practical implementation, setting the local accuracy $\varepsilon_{k,p} = \varepsilon / \sqrt{dL + d - 1}$ provides the controlled accuracy in the whole tensor ε .

Remark 6. The Tucker, TT or extended TT are certain cases of the QTT-Tucker format. The error estimate above reduces to the corresponding estimates of simpler formats straightforwardly, by setting certain terms (being exactly represented) to zero.

6. Optimization problems in the QTT-Tucker format.

6.1. Problem setting. In numerical modeling, we would not be satisfied with the data compression techniques only. The problem of equal importance, is how to solve a certain equation keeping all the data in the format.

We are to optimize a functional $J(x)$ with x in the format $J(x) \rightarrow \min$, $x \in \mathbb{M}$. We are interested in the following functionals:

$$\begin{aligned} (a) \quad \text{error} & \quad J(x) = \|x - y\|^2 & \Leftrightarrow & \quad x = y, \\ (b) \quad \text{energy} & \quad J(x) = (Ax, x) - 2(y, x) & \Leftrightarrow & \quad Ax = y, \quad A^T = A, \\ (6.1) \quad (c) \quad \text{residual} & \quad J(x) = \|Ax - y\|^2 & \Leftrightarrow & \quad A^T Ax = A^T y, \text{ or} \\ (d) \quad \text{Rayleigh quotient} & \quad J(x) = \frac{(Ax, x)}{(x, x)} & \Leftrightarrow & \quad Ax = \lambda x, \quad A^T = A. \end{aligned}$$

They correspond to the approximation, linear system solution, and eigenvalue problems, and what is important, are reducible to the format parameters without changing the form, which allows application of efficient matrix methods.

Provided with the fast algebra and rounding procedure, one can implement all the standard matrix-vector methods; see, for example, the Krylov methods in TT and HT [3, 13], preconditioned iterations, or greedy algorithms. However, these methods are required to keep some *intermediate* vectors of the same size as the initial tensor, thus they also must be approximated. Unfortunately, in many cases, the better the solution is, the worse structure (and larger ranks) these vectors have (e.g., residual, Krylov vectors, etc.), and though advanced techniques relax the complexity to some extent [13], a much better way is to deduce a solver, exploiting the format structure.

Since the invention of the canonical format, one of such techniques was the alternating least squares (ALS) method for approximation and solution purposes. It uses the separation of variables naturally; all the factors except one are fixed, the minimization/solution problem is reduced to its elements, then some other factor is

updated (usually, the next), and so on. A similar approach is applicable to more complex formats, such as Tucker, TT, HT. However, it has several serious drawbacks:

- the ranks must be prescribed, and are not changed during the iterations, i.e., the method is not adaptive;
- the convergence might be very slow;
- in some formats (e.g., canonical, tensor chain), the problem is in general ill-conditioned and unstable.

Fortunately, for the Tucker, TT, and HT formats, there exist modifications which can treat these drawbacks. There is a lack of convergence analysis (sort of given in [55], but almost inapplicable in practice); however most numerical experiments with practical problems manifest good performance.

6.2. From the Tucker ALS to the QTT-Tucker DMRG. For the Tucker format, the alternating scheme called TALS (Tucker ALS) (see [12] for details on the approximation problem $J = \|x - y\|^2$) is conducted as described in Algorithm 2. We see that the reduced optimization problem affects two connected blocks of the format, and thus the rank can be updated in each step of the algorithm, which is similar to the DMRG algorithm [60, 53, 39, 28, 49] for the MPS (TT) format.

ALGORITHM 2. TUCKER ALS ITERATION.

Require: The functional $J(x)$, initial guess x (with orthogonal $U^{(k)}$), accuracy ε or rank R tolerances.

Ensure: Improved solution x in the Tucker format.

- 1: **for** $k = 1, \dots, d$ **do**
 - 2: Fix all the factors except k th, optimize over this factor connected with the core:

$$W^{(k)} = \arg \min_{W^{(k)}} J(x) \in \mathbb{R}^{n \times r^{d-1}},$$

$$x(i_1, \dots, i_d) = \left(W^{(k)}(i_k) \right)^\top \left(U^{(1)}(i_1) \otimes \dots \otimes U^{(k-1)}(i_{k-1}) \right. \\ \left. \otimes U^{(k+1)}(i_{k+1}) \otimes \dots \otimes U^{(d)}(i_d) \right).$$
 - 3: Split the updated factor and core:
 $[U^{(k)}, S, G] = \text{svd}(W^{(k)}, \varepsilon, R), \quad G(\gamma_k, :) = S_{\gamma_k} G(\gamma_k, :).$
 - 4: **end for**
-

We briefly review the ALS_{TT} and DMRG_{TT} algorithms here. For more details, see [39, 28, 49].

The major difference between the Tucker ALS and the ALS_{TT} methods is that the ALS_{TT} is not adaptive, and all ranks are fixed to some prescribed values. That is, if they are underestimated for the prescribed threshold cannot be reached, the latter cannot be reached (however, the method usually converges to the optimal solution with given ranks). The ALS_{TT} method is written in Algorithm 3.

The DMRG_{TT} ([60], MALS in [28], TT solve in [49], DMRG MatVec in [46]) allows one to determine the ranks adaptively *at runtime*. Instead of minimizing over each block, it merges *two* neighboring blocks into one larger block with two mode indices, performs the ALS optimization step on such a $(d-1)$ -dimensional tensor, and then splits the large block back to two TT blocks via the SVD. On the latter step, the rank is updated, since the data were improved during optimization. See Algorithm 4.

What is nice is that the Tucker ALS scheme generalizes to the QTT-Tucker case straightforwardly by means of the following substitutions:

- instead of the factor-core connected block $W^{(k)} = U^{(k)}G$, we can consider the extended factor (see Definition 5.1) of sizes $n \times r^2$, like in the QTT-Tucker rounding Algorithm 1;

ALGORITHM 3. ALS_{TT} ITERATION.

Require: The functional $J(x)$, initial guess x (with orthogonal $G^{(2)}, \dots, G^{(d)}$).**Ensure:** Improved solution x in the TT format.

```

1: for  $k = 1, \dots, d, d-1, \dots, 1$  do
2:   Fix all the blocks except  $k$ th, optimize  $G^{(k)}(i_k) = \arg \min_{G^{(k)}} J(x)$ .
3:   if  $k$  is increasing {Left orthogonality} then
4:      $G^{(k)}(i_k) = Q^{(k)}(i_k)R$ ,  $G^{(k)} = Q^{(k)}$ ,  $G^{(k+1)}(i_{k+1}) = RG^{(k+1)}(i_{k+1})$ .
5:   else
6:      $G^{(k)}(i_k) = RQ^{(k)}(i_k)$ ,  $G^{(k)} = Q^{(k)}$ ,  $G^{(k-1)}(i_{k-1}) = G^{(k-1)}(i_{k-1})R$ .
7:   end if
8: end for

```

ALGORITHM 4. DMRG_{TT} ITERATION.

Require: The functional $J(x)$, initial guess x (with orthogonal $G^{(2)}, \dots, G^{(d)}$), accuracy ε or rank R tolerances.**Ensure:** Improved solution x in the TT format.

```

1: for  $k = 1, \dots, d-1, d-2, \dots, 1$  do
2:   Fix all the blocks except  $k, k+1$ , merge  $W^{(k)}(i_k, i_{k+1}) = G^{(k)}(i_k)G^{(k+1)}(i_{k+1})$ .
3:   optimize  $W^{(k)}(i_k, i_{k+1}) = \arg \min_{W^{(k)}} J(x)$ .
4:   Split:  $[U^{(k)}, S, V^{(k+1)}] = \text{svd} \left( \{W^{(k)}(i_k, i_{k+1})\}_{i_k, i_{k+1}=1}^{n_k, n_{k+1}}, \varepsilon, R \right)$ .
5:   if  $k$  is increasing then
6:      $G^{(k)} = U^{(k)}$ ,  $G^{(k+1)}(i_{k+1}) = SV^{(k+1)}(i_{k+1})$ .
7:   else
8:      $G^{(k+1)} = V^{(k+1)}$ ,  $G^{(k)}(i_k) = U^{(k)}(i_k)S$ .
9:   end if
10: end for

```

- if $L > 1$, instead of the full-format optimization of the extended factor we employ the DMRG_{TT} Algorithm 4 to update the QTT representation for $E^{(k)}$;
- since each factor optimization does not touch the core ranks, we have to add the core optimization step (also via the DMRG_{TT} method).

Now, we can write the QTT-Tucker optimization algorithm, Algorithm 5.

In the same way as the QTT-Tucker rounding mimics the HOSVD, the QTT-Tucker DMRG algorithm implicitly performs the TALS. Note the similarity with the rounding Algorithm 1; indeed, it is the particular case of the optimization algorithm, with the initial guess $x = y$.

6.3. Some technical details on the QTT-Tucker DMRG method. An important operation, which is absent in Algorithm 5, is the reduction of the functional $J(x)$ to a certain part of the format (extended factor, or core). For the TT case, it was shown [28, 49], that it is equivalent to the *projection* of the functional gradient via the fixed TT blocks, and moreover, this projection is *orthogonal*, if the proper orthogonality constraints are imposed on fixed blocks. For the QTT-Tucker case the situation is the same, but we need to describe technical differences.

We consider in this section only the energy functional minimization. Its optimal condition is written as $Ax = y$, similarly to the error/residual functionals.

The first operation is the core optimization step (line 4). We need to prepare the core matrix in the TT format, i.e., the projection of the full QTT-Tucker MatVec Ax

ALGORITHM 5. QTT-TUCKER DMRG (OR ALS) ITERATION.

Require: The functional $J(x)$, initial guess x , accuracy ε or rank R tolerances.**Ensure:** Improved solution x in the QTT-Tucker format.

- 1: **for** $k = 1, \dots, d$ **do**
- 2: Left-orthogonalize the extended factor:

$$E^{(k)}(i_k, \alpha_{k-1}, \alpha_k) = \text{ort}_{\text{TT}}(E^{(k)}(i_k, \alpha_{k-1}, \alpha_k)).$$
- 3: **end for**
- 4: Optimize the core $G = \text{DMRG}_{\text{TT}}(J, G, \varepsilon, R)$, using the DMRG_{TT} Algorithm 4 or ALS Algorithm 3, such that G is left-orthogonal.
- 5: **for** $k = d, \dots, 2$ **do**
- 6: Optimize the extended factor:

$$E^{(k)}(i_k, \alpha_{k-1}, \alpha_k) = \text{DMRG}_{\text{TT}}(J, E^{(k)}(i_k, \alpha_{k-1}, \alpha_k), \varepsilon, R)$$
such that it is left-orthogonal.
- 7: Right-orthogonalize the core block $G^{(k)}$:

$$G^{(k)}(\gamma_k) = RQ^{(k)}(\gamma_k), \quad G^{(k)} = Q^{(k)}, \quad G^{(k-1)}(\gamma_{k-1}) = G^{(k-1)}(\gamma_{k-1})R.$$
- 8: **end for**
- 9: Optimize the first extended factor

$$E^{(1)}(i_1, \alpha_1) = \text{DMRG}_{\text{TT}}(J, E^{(1)}(i_1, \alpha_1), \varepsilon, R).$$

and right-hand side y on factors. The linear system on the core reads

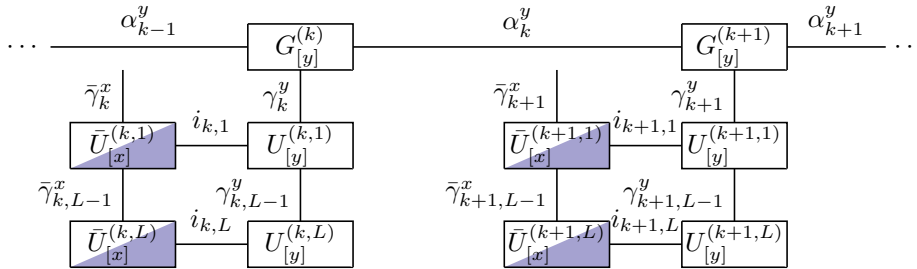
$$\sum_{\gamma_1^x, \dots, \gamma_d^x} P_{[A]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x, \gamma_1^x, \dots, \gamma_d^x) G_{[x]}(\gamma_1^x, \dots, \gamma_d^x) = P_{[y]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x),$$

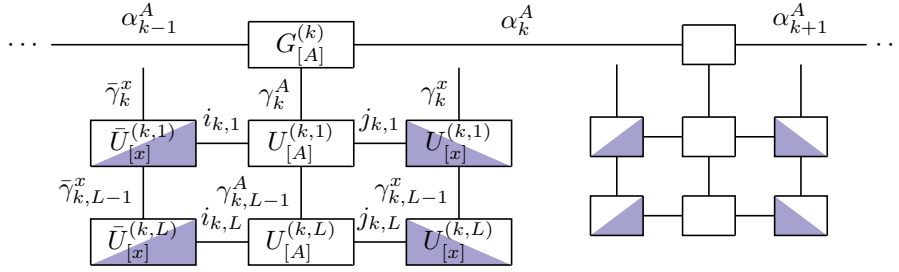
$$P_{[A]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x, \gamma_1^x, \dots, \gamma_d^x) = P_{[A]}^{(1)}(\bar{\gamma}_1^x, \gamma_1^x) \cdots P_{[A]}^{(d)}(\bar{\gamma}_d^x, \gamma_d^x),$$

$$P_{[y]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x) = P_{[y]}^{(1)}(\bar{\gamma}_1^x) \cdots P_{[y]}^{(d)}(\bar{\gamma}_d^x),$$

where the tensor networks for the projected right-hand side $P_{[y]}$ and matrix $P_{[A]}$ are shown in Figures 6.1 and 6.2, respectively (for brevity, only the part corresponding to the k and $k+1$ core blocks are presented). We see that the projection is performed by contracting the orthogonal factor blocks $U_{[x]}^{(k)}$ over the indices $i_{k,p}$ with the parts of the linear system Ax and y , and the Tucker ranks play the role of the mode sizes for the core optimization. Note that if $A = I$ (the simple approximation problem is posed), the matrix projection is also identity due to the orthogonality, and we can skip the equation in Figure 6.2.

As soon as $P_{[A]}$ and $P_{[y]}$ are computed, they can be easily used to reduce the problem to extended factors. Indeed the step 6 of Algorithm 5 means that all the

FIG. 6.1. $P_{[y]}^{(k)}(\bar{\gamma}_k^x)P_{[y]}^{(k+1)}(\bar{\gamma}_{k+1}^x)$.

FIG. 6.2. $P_{[A]}^{(k)}(\bar{\gamma}_k^x, \gamma_k^x)P_{[A]}^{(k+1)}(\bar{\gamma}_{k+1}^x, \gamma_{k+1}^x)$.

factors, and all the core blocks except the k th, are fixed. Thus, all the blocks of $P_{[A]}$ and $P_{[y]}$ except the k th contain proper projections with factors. Now, we only need to perform their projection on all $G_{[x]}$ blocks except the k th, in the same way, as it was done in the ALS_{TT} and DMRG routines [28, 49]. After that, we will have the matrix and right-hand side blocks for computing the last block of the extended factor $G^{(k)}(\alpha_{k-1}, \alpha_k)$.

Notice that in the eigenvalue problem (6.1)(d), the local system on the core reads

$$\sum_{\gamma_1^x, \dots, \gamma_d^x} P_{[A]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x, \gamma_1^x, \dots, \gamma_d^x) G_{[x]}(\gamma_1^x, \dots, \gamma_d^x) = \lambda G_{[x]}(\bar{\gamma}_1^x, \dots, \bar{\gamma}_d^x).$$

6.4. Discussion on convergence and complexity. Despite the lack of theoretical convergence analysis of the DMRG-type algorithms, in most numerical experiments they perform quite well. Sometimes, several technical tricks have to be added [49, 46], which are reasonable in certain cases. Initially, the DMRG algorithm was applied to the MPS format in quantum chemistry and recently in the numerical analysis community, including the HT variant [41]. Here, the QTT-Tucker version is introduced. All these approaches give satisfactory results for all considered problems.

The complexity issues remain open even in practice. One of them is the difference between the ALS and DMRG approaches. In the ALS algorithm, we have to specify (usually, overestimated) ranks a priori, so the first iterations are as difficult as the last (or even more, if the linear systems are ill-conditioned). Moreover, as was pointed out, the fixed-rank ALS may converge slowly. The DMRG algorithm allows one to start with a low-rank initial guess (first iterations are cheap), and, when the ranks become larger, we already have some reasonable approximation. Thus, the DMRG approach works better on first glance.

However, this consideration excludes the complexity of each step. One block in the DMRG method consists of two blocks of the initial tensor (see line 2 in Algorithm 4), thus its storage is $\mathcal{O}(r^2 n^2)$ instead of $\mathcal{O}(r^2 n)$ in the ALS. The truncation step (line 4, Algorithm 4) involves SVD with the complexity $\mathcal{O}(r^3 n^3)$. In the linear QTT format this is not a serious issue, since $n = n_Q = 2$. However, for the optimization of the Tucker core, $n = r_T$ must be plugged into the DMRG complexities, resulting in $\mathcal{O}(dr_T^3 r_C^3) = \mathcal{O}(dr^6)$ cost. In this case, the overhead is more significant. Thus, the development of improved adaptive one-block techniques is planned for future research.

7. Numerical experiments. In this section we compare the performances of the linear QTT and QTT-Tucker formats (quantized mode size $n_Q = 2$ in all cases). The asymptotic storage complexity of the linear QTT format is smaller than that of the QTT-Tucker, but the ranks of the latter are usually much milder. Usually the

ranks should grow with the accuracy, and the resulting computational complexity will be tracked.

All algorithms were developed in MATLAB in the framework of the TT-Toolbox 2.2 by Oseledets et al.,⁵ and run on a Linux x86_64 machine with Intel Xeon E5504 processor at 2.00 GHz.

7.1. Poisson equation. As the first example, consider the five-dimensional Poisson equation

$$\begin{aligned} -\Delta u &= f = 1 \quad \text{in } \Omega = [0, 1]^5, \\ u_{\partial\Omega} &= 0, \end{aligned}$$

discretized using the finite difference scheme on a grid with $n = 256$ points. To solve the tensor linear system, we use the DMRG_{TT} solver [49], Algorithm 4 for the linear QTT, and also the DMRG-like Algorithm 5 for the QTT-Tucker.

We compare both the storage complexity (maximal rank and the total memory cells used to keep the representation) and the solution timings and residuals versus the truncation accuracy ε in Frobenius norm; see Figures 7.1–7.4.

The maximal rank is smaller in the QTT-Tucker format, but the memory plots have an intersection at $\varepsilon^* \sim 10^{-3}$. The very impressive thing is the “reversed” behavior of the plots: due to higher asymptotic complexity of the QTT-Tucker, we might expect it to be less efficient for smaller tolerances and hence larger ranks. On the contrary, the ranks of the linear QTT grow so fast, that the QTT-Tucker outperforms the linear QTT for high accuracies. If the ranks are almost the same (lower accuracies), the linear QTT is preferable.

One may note, that for both formats the memory requirements are still mild: we need only $10^4 - 10^5$ cells to approximate the tensor with $256^5 \sim 10^{12}$ elements with single precision.

Now, let us consider the solution times and the residual (Figures 7.3, 7.4).

Like in the storage figure, the CPU times have a “reversed” intersection, but at the larger accuracy $\varepsilon^* \sim 5 \cdot 10^{-5}$. We recall here that the solution complexity is higher than storage: the DMRG method for the linear QTT requires $\mathcal{O}(d \log_2(n) r^3)$ operations, and for the QTT-Tucker $\mathcal{O}(d \log_2(n) r_F^3 + d r_C^6)$. However, since generally it is

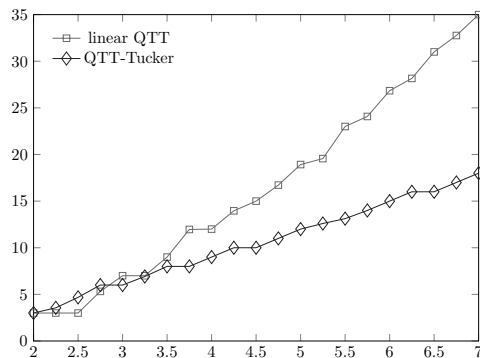


FIG. 7.1. Maximal rank versus $-\log_{10}(\varepsilon)$. Poisson equation.

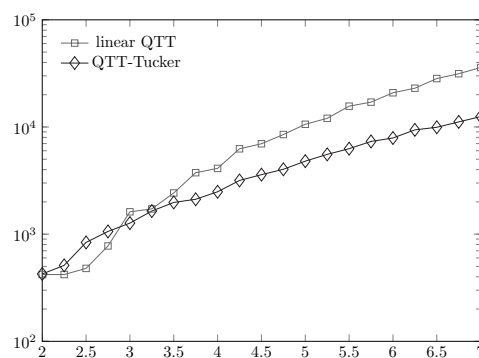


FIG. 7.2. Total memory cells versus $-\log_{10}(\varepsilon)$. Poisson equation.

⁵Free download from <http://github.com/oseledets/TT-Toolbox>

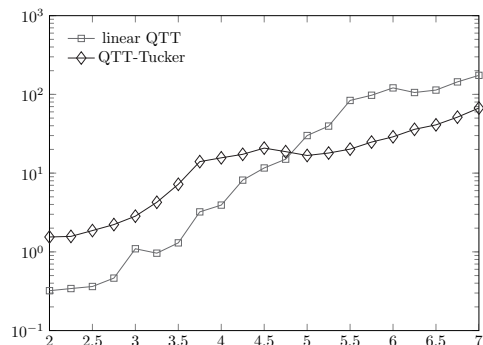


FIG. 7.3. CPU time (sec) versus $-\log_{10}(\varepsilon)$. Poisson equation.

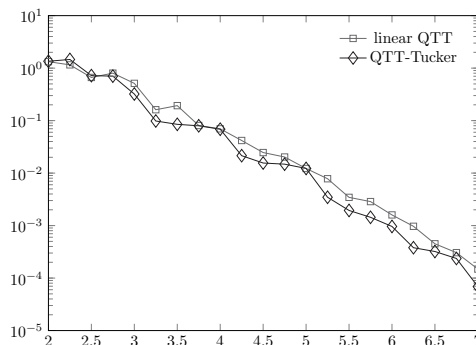


FIG. 7.4. Residual $\|-\Delta u - f\|/\|f\|$ versus $-\log_{10}(\varepsilon)$. Poisson equation.

possible to keep $r_C < r$, $r_F < r$ for the same accuracy, the practically observed complexity appears to be more optimistic than the theoretical one.

The nonmonotonous behavior of the CPU times is mostly due to the two-way solution of local problems in DMRG (line 3 in Algorithm 4): when the local problem is small, the system is solved directly via LU factorization, but with increasing ranks, it switches to the GMRES method (see [49] for details). However, the GMRES method may deliver a certain error in the solution, which is also a source of nonregularity in the CPU times.

It is worth checking the residuals, since they are the only available measure of the solution correctness. From Figure 7.4 we see that the methods manifest almost equal convergence rates, and the QTT-Tucker is even slightly better.

Remark 7. One might see that the relative residual is much larger than the relative rounding accuracy, due to significantly large constants of equivalence (c_1 , c_2),

$$\frac{1}{c_1} \frac{\|A(x-y)\|}{\|Ax\|} \leq \frac{\|x-y\|}{\|x\|} \leq c_2 \frac{\|A(x-y)\|}{\|Ax\|},$$

depending on the condition number of the matrix.

7.2. Generalized Gaussian function—solution to a high dimensional Fokker–Planck equation. One interesting application of tensor methods is the Fokker–Planck equation, which is usually high dimensional. It models the joint probability density distribution of noisy dynamical system configurations (e.g., positions of particles). The (stochastic) system ODE reads

$$\frac{dx}{dt} = -A(x) + G\eta \in \mathbb{R}^d,$$

where $\langle \eta \rangle = 0$, $\langle \eta_i \eta_j \rangle = \delta_{ij}$. The probability of finding the configurations in some volume $x^* + dx$ is written as

$$P(x \in \mathbb{B}_{|dx|}(x^*)) = \psi(x^*)dx,$$

and the *deterministic* PDE on the probability density is the Fokker–Planck equation

$$\frac{\partial \psi}{\partial t} = \frac{\partial}{\partial x} \cdot (A(x)\psi) + \frac{1}{2} \frac{\partial}{\partial x} \cdot \left(D \frac{\partial \psi}{\partial x} \right), \quad D = GG^T.$$

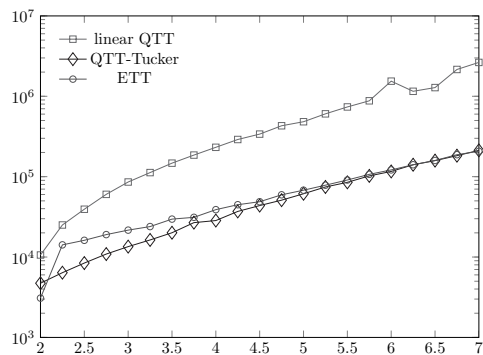


FIG. 7.5. *Memory cells versus $-\log_{10}(\epsilon)$. Generalized Gaussian 12-dimensional example.*

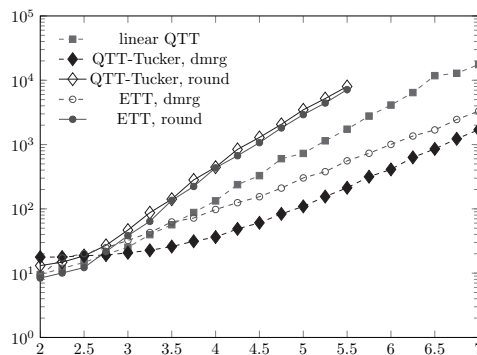


FIG. 7.6. *CPU time versus $-\log_{10}(\epsilon)$. Generalized Gaussian 12-dimensional example.*

It was proven, for example, in [7], that for a linear system $A(x) = Ax$, the (unnormalized) steady probability density is given by the generalized Gaussian

$$(7.1) \quad \psi|_{t \rightarrow \infty} = \exp(-x^T B x),$$

where B is the solution of the following Lyapunov matrix equation:

$$AB^{-1} + B^{-1}A^T = -2D = -2GG^T.$$

So, we would like at least to compute and store the generalized Gaussians in a tensor format.

7.2.1. 12-dimensional example. In this test, we consider the three-dimensional Hookean model for a polymer chain [9, 1, 17] with four springs in the shear flow regime, which yields us the 12-dimensional problem, with the matrix

$$A = I_{4 \times 4} \otimes \begin{bmatrix} 0 & \beta & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{2}D, \quad D = \frac{1}{2} \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \otimes I_{3 \times 3}, \quad \beta = 0.2.$$

This problem was also considered in [15], with an attempt to solve the corresponding Fokker–Planck equation in the linear QTT format. It was shown that the ranks grow linearly with the shear flow rate β , and to get a satisfactory approximation of the stationary solution requires several hours of CPU time. The exponential argument $V = x^T B x$ admits an exact low-rank TT decomposition; see Theorem 4.2. To construct the stationary solution (7.1), we use the scaling and squaring method [44] to compute the pointwise exponential in the QTT or QTT-Tucker formats.

In Figures 7.5, 7.6 we show, respectively, the memory cells required to store the Fokker–Planck equation solution, and the CPU time of the scaling and squaring method versus the Frobenius rounding accuracy in different formats and methods. We compare the linear QTT, QTT-Tucker (with mode size 2), and also the Extended TT (ETT) formats ($n = 256$, $L = 1$). The most time consuming operation is the squaring of the approximant, and there are two approaches: exact Hadamard product + rounding (round), and the DMRG-based approximation of the result (dmrg). We failed to proceed with the former approach for high accuracies due to the memory

limitations: the ranks of the product are squared, thus we need $\mathcal{O}(dLr_F^4 + dr_C^6)$ storage, and the rounding complexity is $\mathcal{O}(dLr_F^6 + dr_C^8)$. By the same reasons, we do not consider the TT format with $n = 256$. The DMRG possesses $\mathcal{O}(dLr_F^4 + dr_C^6)$ computational, and $\mathcal{O}(dLr_F^3 + dr_C^4)$ storage complexities (note the difference with the DMRG cost in the previous example: in the solver, the tensor ranks of the matrix are fixed and excluded from the estimates, while in the Hadamard product both inputs have the same ranks). It is also interesting, that the QTT-Tucker and ETT formats occupy asymptotically the same amount of memory (which indicates that the largest rank is in the Tucker core, r_C), but since the most computations are performed with the Tucker factors, the DMRG procedure is faster in the QTT-Tucker format.

In this example, the complexity with the QTT-Tucker format is more than 10 times smaller than with the linear QTT for 6–7 digits of accuracy. It shows the advantages of the new format in solutions of nontrivial high-dimensional problems.

7.3. Superfast Fourier transform. To see more significant speedup of the QTT-Tucker versus linear QTT, let us consider the multidimensional discrete Fourier transform (DFT).

Given a tensor $X(i_1, \dots, i_d)$, $i_k = 0, \dots, n_k - 1$, $k = 1, \dots, d$, the DFT is defined as follows:

$$(7.2) \quad Y(j_1, \dots, j_d) = \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} \frac{1}{\sqrt{n_1}} \exp\left(-2\pi i \frac{i_1 j_1}{n_1}\right) \cdots \frac{1}{\sqrt{n_d}} \exp\left(-2\pi i \frac{i_d j_d}{n_d}\right) X(i_1, \dots, i_d).$$

First, notice that the multidimensional DFT is in fact a sequence of independent one-dimensional transforms in each variable. Second, the special structure of the exponential weights allows one to develop fast algorithms for $n = 2^L$, for example, the Cooley–Tukey radix-2 recursion formula [10, 18], which in the matrix description reads

$$(7.3) \quad P_L F_L = \begin{bmatrix} F_{L-1} & \\ & F_{L-1} \end{bmatrix} \begin{bmatrix} I_{L-1} & \\ & \Omega_{L-1} \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} I_{L-1} & I_{L-1} \\ I_{L-1} & -I_{L-1} \end{bmatrix},$$

where $F_p = [\frac{1}{\sqrt{2^p}} \exp(-2\pi i \frac{ij}{2^p})]_{i,j=0}^{2^p-1}$ is the Fourier matrix, I_p is the identity of size 2^p , $\Omega_p = \text{diag}(F_{p+1}(0 : 2^p - 1, 1))$, and P_p is the bit-shift permutation matrix, which agglomerates even and odd elements of a vector. Thus, only the Fourier transform of size 2 is actually computed according to (7.2); all the rest of the operations involve sparse matrices, leading to $\mathcal{O}(n \log(n))$ complexity.

What is even more important in our context, the binary splitting in (7.3) allows its efficient realization in the QTT format [16]. For example, the multiplication with the last matrix in (7.3) touches the last QTT block only, without changing the rank. The product with $\text{diag}(I_p, \Omega_p)$ increases the ranks twice, and the rounding is needed. After all steps of the recursion, the even-odd permutation matrices aggregate to the bit-reverse permutation, which is performed in the QTT format with no rank change by simply reversing the QTT blocks, $Y^{(p)} := (Y^{(L-p+1)})^\top$. As a result, we achieve the $\mathcal{O}(L^2 r^3) = \mathcal{O}(\log(n)^2 r^3)$ complexity, thus naming this algorithm “superfast.” For more information we refer to [16].

Now we need to consider carefully the multidimensional ($d > 1$) DFT. As was noted, it reduces to the one-dimensional DFT in each index i_k . In the linear QTT format it corresponds to performing the superfast QTT DFT on each *subtrain*

$$G_{\alpha_{k-1},:}^{(k,1)}(i_{k,1}) G^{(k,2)}(i_{k,2}) \cdots G_{:, \alpha_k}^{(k,L)}(i_{k,L}) \quad \text{for each } \alpha_{k-1}, \alpha_k.$$

After the bit-reverse permutation the QTT subtrain $Y^{(k)}$ reads:

$$G_{[Y] \alpha_k, :}^{(k,1)}(j_{k,1}) G_{[Y]}^{(k,2)}(j_{k,2}) \cdots G_{[Y] :, \alpha_{k-1}}^{(k,L)}(j_{k,L}).$$

The TT rank indices α_{k-1}, α_k are swapped and do not match the neighboring blocks anymore. We need to “transplant” α_{k-1} into the first block, and α_k into the last one. One way to do this is presented in Algorithm 6. In this way, we have to perform r_C^2 roundings of the complexity $\mathcal{O}(r_Q^3)$ each, leading to $\mathcal{O}(r^5)$ in total (assuming $r_T \sim r_C \sim r_F$).

ALGORITHM 6. BOUNDARY (Q)TT RANKS REVERSE.

Require: The QTT tensor $Y_{\alpha_k, \alpha_{k-1}}(i_k)$, accuracy ε .

Ensure: The tensor with reversed boundary ranks $Z_{\alpha_{k-1}, \alpha_k}(i_k) \approx Y_{\alpha_k, \alpha_{k-1}}(i_k)$.

```

1: Set  $Z = []$ .
2: for  $\alpha_{k-1} = 1, \dots, r_{C,k-1}$  do
3:   Set  $W = []$ .
4:   for  $\alpha_k = 1, \dots, r_{C,k}$  do
5:     Extract one component  $Y_{\alpha_k, \alpha_{k-1}}$ .
6:     TT-concatenate  $W_{1:\alpha_k} = [W_{1:\alpha_{k-1}} \quad Y_{\alpha_k, \alpha_{k-1}}]$ .  $\{r^W := r^W + r^Y\}$ 
7:     Round  $W = \text{round}(W, \varepsilon)$ .
8:   end for
9:   TT-concatenate  $Z_{1:\alpha_{k-1}, :} = \begin{bmatrix} Z_{1:\alpha_{k-1}-1, :} \\ W \end{bmatrix}$ .  $\{r^Z := r^Z + r^W\}$ 
10:  Round  $Z = \text{round}(Z, \varepsilon)$ .
11: end for
```

In the QTT-Tucker format, the one-dimensional DFTs are computed for the Tucker factors only. After finishing each DFT, the corresponding factor is the following QTT tensor:

$$U_{[y] 1, :}^{(k,1)}(j_{k,1}) U_{[y]}^{(k,2)}(j_{k,2}) \cdots U_{[y] :, \gamma_k}^{(k,L)}(j_{k,L}),$$

but we wish γ_k to enter $U^{(k,1)}$ according to (3.3). Algorithm 6 may be used as well, but now the inner loop degenerates simply to $W = U_{\gamma_k}^{(k)}$. The resulting complexity is $\mathcal{O}(r_T \cdot r_F^3)$, so we get the cost reduction from r^5 in the linear QTT to r^4 in the QTT-Tucker. Moreover, as we have seen previously, the ranks of the Tucker factors might be smaller than those of the linear QTT for the same accuracy, i.e., $r_F \leq r_Q$.

To illustrate the effect of the QTT-Tucker format, let us consider the Fourier transform of the function

$$f(x) = \frac{1}{||x||^{d-2}} = \frac{1}{||x||^2}, \quad x \in [-10, 10]^4.$$

This function is known to be the fundamental solution of the Poisson equation in $d = 4$: given $-\Delta u = g$ in \mathbb{R}^d , the solution can be computed as a convolution $u(x) = Cf(x) * g(x)$. On the discrete level the convolution reduces to the product of a Toeplitz matrix by a vector, which can be done efficiently via the triple Fourier transform [19]. Also, the convolution can be computed in the QTT directly [30].

We discretize $f(x)$ on the tensor product uniform grid: $x_k, i_k = -10 + \frac{h}{2} + i_k \cdot h$, $h = \frac{20}{2^L}$, which is separated from zero symmetrically by $\pm h/2$ and contains 2^L points in each direction. This corresponds to the Nyström collocation technique. Its approximation error $\mathcal{O}(h^2 \log(h))$ was addressed in [34, 16].

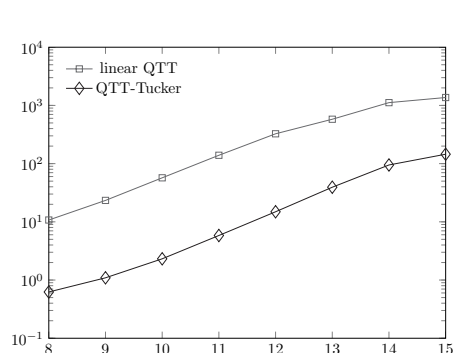


FIG. 7.7. CPU time (sec) versus L , $\varepsilon = 10^{-5}$. A four-dimensional FFT example.

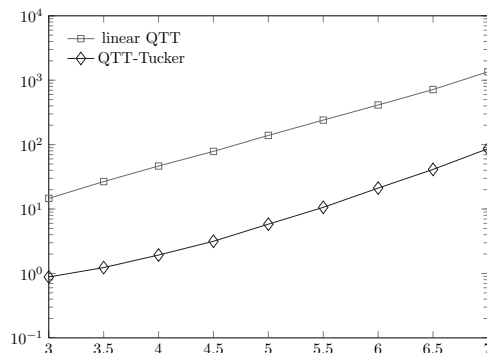


FIG. 7.8. CPU time (sec) versus $-\log_{10}(\varepsilon)$, $L = 11$. A four-dimensional FFT example.

There are several ways to prepare the tensor $F(i_1, \dots, i_d) = f(x_{1,i_1}, \dots, x_{d,i_d})$. First, one generates $h(x) = \|x\|^2 = x_1^2 \otimes \dots \otimes \mathbf{1} + \dots + \mathbf{1} \otimes \dots \otimes x_d^2$ with $r_C = r_T = 2$ like the Laplacian operator in section 4.1. Second, one calculates $f(x) = 1/h(x)$ either by Newton iterations $f = 2f - fhf$ for the pointwise inversion (the method of choice in this paper), or by the quadrature applied to $h^{-1} = \int \exp(-th)dt$ [24], or by the TT cross interpolation technique [52, 56].

Now, we show the CPU times of the DFT(F) computation in the linear QTT and QTT-Tucker format versus the grid level L (Figure 7.7) and the rounding accuracy ε (Figure 7.8). Both linear QTT ranks, and the factor ranks of the QTT-Tucker are about 60, but we observe that the QTT-Tucker FFT outperforms the linear QTT FFT by a factor greater than 10 in all cases, and even > 20 at $L = 11$, $\varepsilon = 10^{-5}$. In addition, even an algorithm description may become simpler, as we have seen on the QTT-Fourier transform example, which makes the new format a very promising tool.

8. Conclusion. A combined tensor representation was introduced. It encapsulates the benefits of the Tucker, TT, and QTT formats. As its predecessors, the new format yields a closed tensor manifold. All basic algebraic operations, as well as rounding and alternating optimization methods are available and stable. Despite the worse asymptotic storage estimate, it usually performs better (giving an acceleration up to several tenths) than the “standard” linear-structured QTT, due to significantly lower ranks required to fix the same accuracy. Moreover, they are distributed more uniformly (similarly to the Tucker ranks), on the contrary to the linear QTT, which normally keeps a “peak” rank at the middle of a TT.

From that point of view, the QTT-Tucker format can be considered as a “rank equalizing” tool. In terms of TT-like formats, an attempt to implement such an idea resulted in the so-called tensor chain [35, 29] format, which is the TT up to the summation over the connected first and last rank indices. Such a representation was expected to manifest a uniform ranks distribution, since none of them are “boundary” or “middle.” Unfortunately, the tensor chain manifold is not closed, and optimization problems are ill-posed. The QTT-Tucker format reduces the peak rank values by making the effective “length” of the involved tensor trains smaller (d or L versus dL), but without losing the stability. Moreover, any tensor chain representation can be easily converted to the QTT-Tucker format.

However, the rank overhead of the DMRG optimization algorithms can be significantly worse in the Tucker-based formats rather than the TT, since the effective

mode sizes are now r but not 2, and DMRG techniques require merging of two blocks. To achieve the same rank complexity as in the rounding procedure, robust one-block alternating methods have to be developed. This is the aim for the future work.

Appendix A.

Proof of Theorem 4.2. First of all, compute the off-diagonal skeleton decompositions

$$\begin{aligned} B_{k+1:d,1:k} &= C^k = W^k U^k, & W^k &\in \mathbb{R}^{d-k \times r_k}, & U^k &\in \mathbb{R}^{r_k \times k}, \\ B_{1:k,k+1:d} &= \hat{C}^k = \hat{U}^{k\top} \hat{W}^{k\top}, & \hat{W}^k &\in \mathbb{R}^{d-k \times \hat{r}_k}, & \hat{U}^k &\in \mathbb{R}^{\hat{r}_k \times k}, \end{aligned}$$

with orthonormal U^k, \hat{U}^k . To get rid of the mode indices in V, X, X' , we agree as in section 4.1 that i_1, \dots, i_d are fixed, but the equations hold for all possible values:

$$\begin{aligned} X'_p(i_1, \dots, i_d) &= e_1(i_1) e_2(i_2) \cdots x'_p(i_p) \cdots e_d(i_d), \\ X_q(i_1, \dots, i_d) &= e_1(i_1) e_2(i_2) \cdots x_q(i_q) \cdots e_d(i_d), \end{aligned}$$

are scalars, as well as $V(i_1, \dots, i_d)$. Now, we start to separate the TT blocks recursively, omitting i_1, \dots, i_d .

In the first step we have $V = B_{11} X'_1 \otimes X_1 + \sum_{p=2}^d B_{p1} X'_p \otimes X_1 + \sum_{q=2}^d B_{1q} X'_1 \otimes X_q + V^{[2]}$, where $V^{[k]} = \sum_{p,q=k}^d B_{pq} X'_p \otimes X_q$. So,

$$(A.1) \quad V = \begin{bmatrix} x_1^2 B_{11} & x'_1 & x_1 & e_1 \end{bmatrix} \begin{bmatrix} E^{[2]} & \sum_{q=2}^d X_q^{[2]} B_{1,q} & \sum_{p=2}^d X_p'^{[2]} B_{p,1} & V^{[2]} \end{bmatrix}^\top,$$

where $E^{[k]} = e_k \cdots e_d$, $X_q^{[k]} = e_k \cdots e_{q-1} \cdot x_q \cdot e_{q+1} \cdots e_d$, $X_p'^{[k]} = e_k \cdots e_{p-1} \cdot x'_p \cdot e_{p+1} \cdots e_d$, $p, q \geq k$. The first term in (A.1) is the first TT block V_1 . On the other hand, we can represent it using the skeleton factors for the first row and column:

$$V = \begin{bmatrix} x_1^2 B_{11} & x'_1 \hat{U}^1 & x_1 U^1 & e_1 \end{bmatrix} \begin{bmatrix} E^{[2]} & \sum_{q=2}^d X_q^{[2]} \hat{W}_{q-1,1}^1 & \sum_{p=2}^d X_p'^{[2]} W_{p-1,1}^1 & V^{[2]} \end{bmatrix}^\top.$$

Now, we need to derive the recursive representation for the second term.

Suppose we have

$$(A.2) \quad \tilde{V} = \begin{bmatrix} E^{[k]} & \sum_{q=k}^d X_q^{[k]} \hat{W}_{q-k+1,:}^{k-1} & \sum_{p=k}^d X_p'^{[k]} W_{p-k+1,:}^{k-1} & V^{[k]} \end{bmatrix}^\top.$$

Applying the first step to $V^{[k]}$, we obtain (I_s is the identity matrix of size s)

$$\tilde{V} = \begin{bmatrix} e_k & & & & & & & \\ x_k \hat{W}_{1,:}^{k-1\top} & e_k I_{\hat{r}_{k-1}} & & & & & & \\ x'_k \hat{W}_{1,:}^{k-1\top} & & e_k I_{r_{k-1}} & & & & & \\ x_k^2 B_{kk} & & & x'_k & x_k & e_k & & \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ \sum_{q=k+1}^d X_q^{[k+1]} \hat{W}_{q-k+1,:}^{k-1\top} \\ \sum_{p=k+1}^d X_p'^{[k+1]} W_{p-k+1,:}^{k-1\top} \\ \sum_{q=k+1}^d X_q^{[k+1]} B_{k,q} \\ \sum_{p=k+1}^d X_p'^{[k+1]} B_{p,k} \\ V^{[k+1]} \end{bmatrix}.$$

Note that in the terms with indices, e.g., $W_{1,:}^{k-1\top}$, we *first* take the slice, and then its transposition. Separating the scalar coefficients from X -related data in the last column, we obtain

$$(A.3) \quad \begin{bmatrix} E^{[k+1]} \\ \sum_{q=k+1}^d X_q^{[k+1]} \hat{W}_{q-k+1,:}^{k-1\top} \\ \sum_{p=k+1}^d X_p'^{[k+1]} W_{p-k+1,:}^{k-1\top} \\ \sum_{q=k+1}^d X_q^{[k+1]} B_{k,q} \\ \sum_{p=k+1}^d X_p'^{[k+1]} B_{p,k} \\ V^{[k+1]} \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & \hat{W}_{2:d-k+1,:}^{k-1\top} & & & \\ & & W_{2:d-k+1,:}^{k-1\top} & & \\ & B_{k,k+1:d} & & B_{k+1:d,k}^\top & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ X_{k+1:d}^{[k+1]} \\ X_{k+1:d}'^{[k+1]} \\ V^{[k+1]} \end{bmatrix}.$$

Now, recall that $W_{2:d-k+1,:}^{k-1\top} = U^{k-1} B_{k+1:d,1:k-1}^\top = U^{k-1} U_{:,1:k-1}^{k\top} W_{1:d-k,:}^{k\top}$. A similar equation holds for \hat{W}^{k-1} . That is, (A.3) splits into the scalar part, and a counterpart of (A.2) with k substituted by $k+1$; therefore,

$$(A.4) \quad \tilde{V} = \begin{bmatrix} e_k & & & & \\ x_k \hat{W}_{1,:}^{k-1\top} & e_k \hat{U}^{k-1} \hat{U}_{:,1:k-1}^{k\top} & & & \\ x_k' W_{1,:}^{k-1\top} & & e_k U^{k-1} U_{:,1:k-1}^{k\top} & & \\ x_k^2 B_{kk} & x_k' \hat{U}_{:,k}^{k\top} & x_k U_{:,k}^{k\top} & e_k & \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ \sum_{q=k+1}^d X_q^{[k+1]} \hat{W}_{q-k,:}^{k\top} \\ \sum_{p=k+1}^d X_p'^{[k+1]} W_{p-k,:}^{k\top} \\ V^{[k+1]} \end{bmatrix}.$$

The first term is nothing else than the k th TT block V_k , since it contains only x_k , e_k . It has the size $(1 + \hat{r}_{k-1} + r_{k-1} + 1) \times (1 + \hat{r}_k + r_k + 1)$, which confirms the first statement of the theorem. Assuming the QTT separability of basis elements e_k , x_k , and so on, obtain the linear QTT rank bound. With the second term we can proceed recursively, and the last TT block reads (since $V^{[d]}$ is just a one-dimensional $x_d^2 B_{dd}$)

$$(A.5) \quad V_d = \begin{bmatrix} e_d & x_d \hat{W}_{1,:}^{d-1} & x_d' W_{1,:}^{d-1} & x_d^2 B_{dd} \end{bmatrix}^\top.$$

If the first-order term is presented with a unique object $x_k = x_k'$, the ranks are reduced as follows. First,

$$V_1 = \begin{bmatrix} x_1^2 B_{11} & 2x_1 & e_1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 0.5 & 0.5 \\ & & 1 \end{bmatrix}.$$

Reassigning $\bar{V}_1 = \begin{bmatrix} x_1^2 B_{11} & 2x_1 & e_1 \end{bmatrix}$, we contract the rest of the matrix with the middle blocks, obtaining

$$\bar{V}_k = \begin{bmatrix} e_k & & & & \\ x_k \hat{W}_{1,:}^{k-1\top} & \frac{1}{2} e_k U^{k-1} U_{:,1:k-1}^{k\top} & \frac{1}{2} e_k U^{k-1} U_{:,1:k-1}^{k\top} & & \\ x_k^2 B_{kk} & x_k U_{:,k}^{k\top} & x_k U_{:,k}^{k\top} & e_k & \end{bmatrix}.$$

We note here, that the constraint $x_k = x_k'$ yields $V \equiv 0$ if $B = -B^\top$, so that only the symmetric part $B := 0.5(B + B^\top)$ is relevant. Thus, $\hat{W}^k = W^k$, $\hat{U}^k = U^k$. The rest

of the TT blocks are simplified as follows:

$$\begin{bmatrix} E^{[k+1]} \\ \sum_{q=k+1}^d X_q^{[k+1]} \hat{W}_{q-k,:}^{k\top} \\ \sum_{p=k+1}^d X_p'^{[k+1]} W_{p-k,:}^{k\top} \\ V^{[k+1]} \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ \sum_{q=k+1}^d X_q^{[k+1]} W_{q-k,:}^{k\top} \\ V^{[k+1]} \end{bmatrix}$$

Multiplying \bar{V}_k with the first (scalar) factor in (A.6), we obtain the reduced block

$$(A.6) \quad \bar{V}_k = \begin{bmatrix} e_k & & & \\ x_k W_{1,:}^{k-1\top} & e_k U^{k-1} U_{:,1:k-1}^{k\top} & & \\ x_k^2 B_{kk} & 2x_k U_{:,k}^{k\top} & e_k & \end{bmatrix} \quad \text{of size } (r_{k-1} + 2) \times (r_k + 2).$$

Applying the TT-to-Tucker conversion technique from section 4.1, we obtain immediately, that the TT ranks of the Tucker core are equal to the ranks obtained above, and the Tucker ranks equal 4 (3 in the symmetric case), since there are 4 (respectively, 3) linearly independent elements in each block, e_k , x_k , x'_k , and x_k^2 .

Remark 8. The proof gives a constructive routine for fast assembly of a bilinear form in the TT format. Indeed, performing the SVDs of submatrices of B (their sizes are of the order of tens) and building the blocks from (A.1), (A.4) (or (A.6) in the symmetric case), (A.5) we get the analytical TT representation.

REFERENCES

- [1] A. AMMAR, B. MOKDAD, F. CHINESTA, AND R. KEUNINGS, *A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids*, J. Non-Newtonian Fluid Mech., 139 (2006), pp. 153–176.
- [2] B. W. BADER AND T. G. KOLDA, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30 (2008), pp. 205–231.
- [3] J. BALLANI AND L. GRASEDYCK, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43.
- [4] R. E. BELLMAN, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.
- [5] S. BERNSTEIN, *Lecons sur les Propriétés Extrémales et la Meilleure Approximation des Fonctions Analytiques d'une Variable Réelle*, Gauthier-Villars, Paris, 1926.
- [6] H.-J. BUNGATZ AND M. GRIEBEL, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.
- [7] B. R. BUTCHART, *An explicit solution to the Fokker-Planck equation for an ordinary differential equation*, Internat. J. Control, 1 (1965), pp. 201–208.
- [8] 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.
- [9] C. CHAUVIÉRE AND A. LOZINSKI, *Simulation of dilute polymer solutions using a Fokker-Planck equation*, Comput. & Fluids, 33 (2004), pp. 687–696.
- [10] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.
- [11] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
- [12] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *On best rank-1 and rank- (R_1, R_2, \dots, R_N) approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.
- [13] S. V. DOLGOV, *TT-GMRES: On solution to a linear system in the structured tensor format*, Russian J. Numer. Anal. Math. Modeling, 28 (2013), pp. 149–172.
- [14] S. V. DOLGOV AND B. N. KHOROMSKIJ, *Two-level Tucker-TT-QTT Format for Optimized Tensor Calculus*, preprint 19, MPI MIS, 2012.

- [15] S. V. DOLGOV, B. N. KHOROMSKIJ, AND I. V. OSELEDETS, *Fast solution of parabolic problems in the tensor train/quantized tensor train-format with initial application to the Fokker-Planck equation*, SIAM J. Sci. Comput., 34 (2012), pp. A3016–A3038.
- [16] S. V. DOLGOV, B. N. KHOROMSKIJ, AND D. V. SAVOSTYANOV, *Superfast Fourier transform using QTT approximation*, J. Fourier Anal. Appl., 18 (2012), pp. 915–953.
- [17] L. E. FIGUEROA AND E. SÜLI, *Greedy approximation of high-dimensional Ornstein-Uhlenbeck operators*, Found. Comput. Math., 12 (2012), pp. 573–623.
- [18] C. F. GAUSS, *Nachlass: Theoria interpolationis methodo nova tractata*, in Werke, Vol. 3, Königliche Gesellschaft der Wissenschaften, Göttingen, 1866, pp. 265–330.
- [19] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1996.
- [20] S. A. GOREINOV, I. V. OSELEDETS, AND D. V. SAVOSTYANOV, *Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case*, SIAM J. Sci. Comput., 34 (2012), pp. A1–A27.
- [21] L. GRASEDYCK AND W. HACKBUSCH, *An introduction to hierarchical (H-) and TT-rank of tensors with examples*, Comput. Methods Appl. Math., 3 (2011), pp. 291–304.
- [22] M. GRIEBEL, F. Y. KUO, AND I. H. SLOAN, *The smoothing effect of the ANOVA decomposition*, J. Complexity, 26 (2010), pp. 523–551.
- [23] W. HACKBUSCH, *Tensor spaces and Numerical Tensor Calculus*, Springer-Verlag, Berlin, 2012.
- [24] W. HACKBUSCH AND B. N. KHOROMSKIJ, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions*, Computing, 76 (2006), pp. 177–202.
- [25] W. HACKBUSCH AND S. KÜHN, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.
- [26] R. A. HARSHMAN, *Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis*, UCLA Working Pap. Phonetics, 16 (1970), pp. 1–84.
- [27] F. L. HITCHCOCK, *Multiple invariants and generalized rank of a p-way matrix or tensor*, J. Math. Phys., 7 (1927), pp. 39–79.
- [28] S. HOLTZ, T. ROHWEDDER, AND R. SCHNEIDER, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.
- [29] T. HUCKLE, K. WALDHERR, AND T. SCHULTE-HERBRÜGGEN, *Computations in quantum tensor networks*, Linear Algebra Appl., 438 (2013), pp. 750–781.
- [30] V. KAZEYEV, B. N. KHOROMSKIJ, AND E. E. TYRTYSHNIKOV, *Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity*, SISC, to appear.
- [31] V. KAZEYEV, O. REICHMANN, AND C. SCHWAB, *Low-Rank Tensor Structure of Linear Diffusion Operators in the TT and QTT Formats*, preprint 2012-13, ETH SAM, Zürich, 2012.
- [32] V. A. KAZEYEV AND B. N. KHOROMSKIJ, *Low-rank explicit QTT representation of the Laplace operator and its inverse*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 742–758.
- [33] B. KHOROMSKIJ AND C. SCHWAB, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput., 33 (2011), pp. 364–385.
- [34] B. N. KHOROMSKIJ, *Fast and accurate tensor approximation of multivariate convolution with linear scaling in dimension*, J. Comput. Appl. Math., 234 (2010), pp. 3122–3139.
- [35] B. N. KHOROMSKIJ, *$\mathcal{O}(d \log n)$ -Quantics approximation of N -d tensors in high-dimensional numerical modeling*, Constr. Approx., 34 (2011), pp. 257–280.
- [36] B. N. KHOROMSKIJ, *Tensor-structured numerical methods in scientific computing: Survey on recent advances*, Chemometr. Intell. Lab. Systems, 110 (2012), pp. 1–19.
- [37] B. N. KHOROMSKIJ AND V. KHOROMSKAIA, *Low rank Tucker-type tensor approximation to classical potentials*, Cent. Eur. J. Math., 5 (2007), pp. 523–550.
- [38] B. N. KHOROMSKIJ AND V. KHOROMSKAIA, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comput., 31 (2009), pp. 3002–3026.
- [39] B. N. KHOROMSKIJ AND I. V. OSELEDETS, *DMRG+QTT Approach to Computation of the Ground State for the Molecular Schrödinger Operator*, preprint 69, MPI MIS, Leipzig, 2010.
- [40] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
- [41] D. KRESSNER AND C. TOBLER, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Comput. Methods Appl. Math., 11 (2011), pp. 363–381.
- [42] R. LIU AND A. OWEN, *Estimating mean dimensionality of analysis of variance decompositions*, J. Amer. Statist. Assoc., 101 (2006), pp. 712–721.

- [43] H.-D. MEYER, F. GATTI, AND G. A. WORTH, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, Wiley-VCH, Weinheim, 2009.
- [44] C. MOLER AND C. VAN LOAN, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.
- [45] I. V. OSELEDETS, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.
- [46] I. V. OSELEDETS, *DMRG approach to fast linear algebra in the TT-format*, Comput. Methods Appl. Math, 11 (2011), pp. 382–393.
- [47] I. V. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
- [48] I. V. OSELEDETS, *Constructive representation of functions in low-rank tensor formats*, Constr. Approx., 37 (2013), pp. 1–18.
- [49] I. V. OSELEDETS AND S. V. DOLGOV, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739.
- [50] 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.
- [51] I. V. OSELEDETS AND E. E. TYRTYSHNIKOV, *Tensor Tree Decomposition Does Not Need a Tree*, preprint 2009-4, INM RAS, Moscow, 2009.
- [52] I. V. OSELEDETS AND E. E. TYRTYSHNIKOV, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.
- [53] S. ÖSTLUND AND S. ROMMER, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett., 75 (1995), pp. 3537–3540.
- [54] D. PEREZ-GARCIA, F. VERSTRAETE, M. WOLF, AND J. CIRAC, *Matrix product state representations*, Quantum Inform. Comput., 7 (2007), pp. 401–430.
- [55] T. ROHWEDDER AND A. USCHMAJEV, *Local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Numer. Anal., 51 (2013), pp. 1134–1162.
- [56] D. V. SAVOSTYANOV AND I. V. OSELEDETS, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, in 7th International Workshop on Multidimensional (nD) Systems (nDs), IEEE, Piscataway, NJ, 2011.
- [57] E. TADMOR, *The exponential accuracy of Fourier and Chebychev differencing methods*, SIAM J. Numer. Anal., 23 (1986), pp. 1–10.
- [58] L. R. TUCKER, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
- [59] T. VON PETERSDORFF AND C. SCHWAB, *Numerical solution of parabolic equations in high dimensions*, ESAIM: Math. Model. Numer. Anal., 38 (2004), pp. 93–127.
- [60] S. R. WHITE, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.