Error Analysis of TT-Format Tensor Algorithms

Error Analysis of TT-format Tensor Algorithms

Dario Fasino and Eugene E. Tyrtyshnikov

Abstract The tensor train (TT) decomposition is a representation technique for arbitrary tensors, which allows efficient storage and computations. For a d-dimensional tensor with $d \geq 2$, that decomposition consists of two ordinary matrices and $d-2$ third-order tensors. In this paper we prove that the TT decomposition of an arbitrary tensor can be computed (or approximated, for data compression purposes) by means of a backward stable algorithm based on computations with Householder matrices. Moreover, multilinear forms with tensors represented in TT format can be computed efficiently with a small backward error.

1 Introduction

The *tensor train decomposition* is a representation technique which allows compact storage and efficient computations with arbitrary tensors. The origins of this representation can be traced back to a brief paper by I. Oseledets and E. Tyrtyshnikov dating 2009 [8], while its popularization is mainly due to the subsequent papers [6, 9]. Nowadays, the tensor train decomposition is a computationally powerful tool that offers viable and convenient alternatives to classical (e.g., Tucker, CP) tensor representations [1, 4], in particular for the approximation of solutions of high dimensional problems. As shown in, e.g., [5, 7], certain computations with large-scale structured matrices and vectors can be conveniently recast in terms of tensor train representations.

Dario Fasino
Department of Mathematics, Computer Science and Physics, University of Udine, Udine, Italy, e-mail: dario.fasino@uniud.it

Eugene E. Tyrtyshnikov
Institute of Numerical Mathematics of Russian Academy of Sciences, Russia, e-mail: eugene.tyrtyshnikov@gmail.com

Basically, a tensor train (TT) decomposition of a d-dimensional tensor A with size $n_1 \times n_2 \times \cdots \times n_d$ is a sequence $\mathsf{G}_1, \ldots, \mathsf{G}_d$ of tensors of order 2 or 3; the size of G_i is $r_{i-1} \times n_i \times r_i$ with $r_0 = r_d = 1$ (that is, G_1 and G_d are ordinary matrices) and

$$\mathsf{A}(i_1, i_2, \ldots, i_d) = \sum_{j_1=1}^{r_1} \cdots \sum_{j_{d-1}=1}^{r_{d-1}} \mathsf{G}_1(i_1, j_1)\mathsf{G}_2(j_1, i_2, j_2) \cdots \mathsf{G}_d(j_{d-1}, i_d). \tag{1}$$

We will denote by $\mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ the tensor identified by the right hand side of (1).

In this paper, we present a backward error analysis of two algorithms, originally devised in [9], which perform computations with tensors in TT-format. The first algorithm produces an exact or approximate TT decomposition $\mathsf{G}_1, \ldots, \mathsf{G}_d$ of an arbitrary d-dimensional tensor A given in functional form, depending on a tolerance parameter ε. If $\varepsilon = 0$ then the output of the algorithm is an exact TT decomposition, that is, $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$. If $\varepsilon > 0$ then $\mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ is an $\mathscr{O}(\varepsilon)$-approximation of A which can realize significant savings in memory space. The computational core of the algorithm is a suitable (approximate) matrix factorization that, in the original paper, relies on SVD computations. We prove that analogous performances and backward stability can be obtained by means of QR factorizations based on Householder transformations.

The second algorithm computes the *contraction* (i.e., multilinear form) of a given d-dimensional tensor A and vectors $v^{(1)}, \ldots, v^{(d)}$,

$$\alpha = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathsf{A}(i_1, i_2, \ldots, i_d) v_{i_1}^{(1)} \cdots v_{i_d}^{(d)}, \tag{2}$$

where A is known in TT format. By means of known error bounds for inner products in floating point arithmetic [3], we prove backward stability of the proposed algorithm under very general hypotheses on the evaluation order of the summations. More precisely, if $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ and no underflows or overflows are encountered then the output computed by the algorithm in floating point arithmetic is the exact contraction of $\widehat{\mathsf{A}} = \mathrm{TT}(\mathsf{G}_1 + \Delta\mathsf{G}_1, \ldots, \mathsf{G}_d + \Delta\mathsf{G}_d)$ and $v^{(1)}, \ldots, v^{(d)}$ where $|\Delta\mathsf{G}_i| \le (n_i + r_{i-1})u|G_i| + \mathscr{O}(u^2)$ and u is the machine precision.

After setting up some basic notations and concepts, in Section 3 we present the algorithm for computing the TT-representation of a tensor and analyze its numerical stability. Next, we discuss in Section 4 the computation in computer arithmetic of the multilinear form (2) with a tensor in TT-format. A final appendix contains a complementary result.

2 Notations and Preliminaries

We refer to [1, Ch. 12] and [4] for notations and fundamental concepts on tensors and basic multilinear operations. Vectors and matrices are denoted by lower-case and upper-case italic letters, respectively, and higher order tensors by upper-case

sans-serif letters: x, X, and X. A tensor X of order d is a multiway array of size $n_1 \times n_2 \times \cdots \times n_d$, where n_k is the size of the kth dimension or mode. A vector is a first-order tensor, and a matrix is a second-order tensor. The $(i_1, i_2, \ldots, i_d)$th entry of $\mathsf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is denoted by $\mathsf{X}_{i_1,i_2,\ldots,i_d}$ or, alternatively, $\mathsf{X}(i_1, i_2, \ldots, i_d)$ to avoid long, multiple subscripts. Lower case greek letters denote real numbers.

Throughout this paper, the prototypical tensor is a d-dimensional array A with dimensions $n_1 \times n_2 \times \cdots \times n_d$. In particular, the scalar d is used exclusively for the order of A, and the scalars $n_1, n_2, \ldots, n_d$ are reserved for A's dimensions. The *size* of A is the number of entries, $N(\mathsf{A}) = n_1 n_2 \cdots n_d$. The *vectorization* of A is the vector $\mathrm{vec}(\mathsf{A}) \in \mathbb{R}^{N(\mathsf{A})}$ whose ith entry is the ith entry of A according to the lexicographic ordering of the indices. The Matlab-style function `reshape` is defined in terms of the vectorization operator as follows. Let $m_1, m_2, \ldots, m_k$ be integers such that $N(\mathsf{A}) = m_1 m_2 \cdots m_k$. Then,

$$\mathsf{B} = \mathtt{reshape}(\mathsf{A}, [m_1, m_2, \ldots, m_k])$$

is the tensor $\mathsf{B} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_k}$ such that $\mathrm{vec}(\mathsf{A}) = \mathrm{vec}(\mathsf{B})$. In particular, for $k = 1, \ldots, d-1$,

$$A_k = \mathtt{reshape}(\mathsf{A}, \left[\textstyle\prod_{i=1}^{k} n_i, \prod_{i=k+1}^{d} n_i\right])$$

is the *k-th unfolding matrix* of A.

The *k-mode product* of a tensor $\mathsf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ by a matrix $M \in \mathbb{R}^{n_k \times m}$, denoted by $\mathsf{A} \times_k M$, is an $(n_1 \times \cdots \times n_{k-1} \times m \times n_{k+1} \times \cdots \times n_d)$-tensor of which the entries are given by

$$(\mathsf{A} \times_k M)(i_1, \ldots, i_d) = \sum_{j=1}^{n_k} \mathsf{A}(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d) M_{j,i_k}.$$

The latter definition extends trivially to the case where M is a vector, by treating it as a $n_k \times 1$ matrix. The k-mode product satisfies the property

$$(\mathsf{A} \times_i B) \times_j C = (\mathsf{A} \times_j C) \times_i B\,, \tag{3}$$

so that notations like $\mathsf{A} \times_i B \times_j C$ can be used without ambiguity.

If A is a vector, a matrix or a tensor, we denote by $|A|$ the componentwise absolute value of A. Inequalities between tensors hold componentwise. Finally, the Frobenius inner product of two matrices $A, B \in \mathbb{R}^{m \times n}$ is $\langle A, B \rangle = \mathrm{trace}(A^T B) = \mathrm{vec}(A)^T \mathrm{vec}(B)$, and the associated matrix norm is $\|A\|_{\mathrm{F}} = \|\mathrm{vec}(A)\|_2$. The latter is extended in an obvious way to arbitrary tensors.

2.1 The Tensor Train Format for Multidimensional Arrays

A tensor train decomposition of $\mathsf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a sequence $\mathsf{G}_1, \ldots, \mathsf{G}_d$ of tensors, called *carriages* (or *cores* [6, 7]), such that the size of G_i is $r_{i-1} \times n_i \times r_i$ with $r_0 =$

$r_d = 1$ (that is, G_1 and G_d can be understood as ordinary matrices) and fulfilling the identity (1), that is, $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$.

Example 1. Let A be a $10 \times 20 \times 30 \times 40$ tensor with TT-ranks $5, 6, 7$. The TT-format of A is made of 4 carriages, whose dimensions are as follows:

G_1 : $1 \times 10 \times 5$
G_2 : $5 \times 20 \times 6$
G_3 : $6 \times 30 \times 7$
G_4 : $7 \times 40 \times 1$.

An alternative viewpoint on the decomposition (1) is

$$\mathsf{A}(i_1, i_2, \ldots, i_d) = G'_1(i_1) G'_2(i_2) \cdots G'_d(i_d),$$

where now $G'_k(i_k)$ is an $r_{k-1} \times r_k$ matrix depending on the integer parameter i_k. The numbers $r_1, \ldots, r_{d-1}$ are called *TT-ranks* of A. As shown in, e.g., [6, 9], r_k is bounded from below by $\mathrm{rank}(A_k)$. Moreover, a TT decomposition with $r_k = \mathrm{rank}(A_k)$ always exists and can be computed by the algorithm shown in [9], which is recalled in the next section. It is often the case that an exact or approximate TT-format of a given tensor yields considerable savings in terms of memory space with respect to other representation techniques.

Remark 1. It is sometimes convenient to assume that G_1 and G_d are not three-dimensional but two-dimensional with sizes $n_1 \times r_1$ and $r_{d-1} \times n_d$, respectively. For that reason, in what follows we will use indifferently the notations G_1 and G_1 to denote the first carriage.

3 Full-to-TT Compression

In this section we address backward stability properties of the compression algorithm from [9], which is recalled hereafter as Algorithm 3.1. This algorithm produces an exact or approximate TT decomposition $\mathsf{G}_1, \ldots, \mathsf{G}_d$ of a given tensor A with $d \geq 2$, depending on a tolerance ε. If $\varepsilon = 0$ then the output of the algorithm is an exact TT decomposition, that is $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ with $r_k = \mathrm{rank}(A_k)$ for $k = 1, \ldots, d-1$, see [9, Thm. 2.1]. If $\varepsilon > 0$ then $r_k \leq \mathrm{rank}(A_k)$ and $\mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ is an $\mathscr{O}(\varepsilon)$-approximation of A which can realize significant savings in memory space with respect to the exact TT-decomposition.

As a basic compression device we suppose to have at our disposal a generic, black-box algorithm having the interface

$$[M, N, r] = \texttt{compress}(A, \varepsilon) \tag{4}$$

which, for any matrix $A \in \mathbb{R}^{m \times n}$ and tolerance $\varepsilon \geq 0$ returns an integer r and two matrices $M \in \mathbb{R}^{m \times r}$ and $N \in \mathbb{R}^{r \times n}$ such that

$$\operatorname{rank}(M) = \operatorname{rank}(N) = r, \qquad \|MN - A\|_{\mathrm{F}} \leq \varepsilon \|A\|_{\mathrm{F}} \,. \tag{5}$$

In particular, if $\varepsilon = 0$ then $A = MN$ is a rank decomposition of A. In [9] this algorithm is realized by means of a truncated SVD of A. In that case, one has

$$\|A - MN\|_{\mathrm{F}} = \min_{\operatorname{rank}(X) \leq r} \|A - X\|_{\mathrm{F}}, \qquad r = \min_{\|A-X\|_{\mathrm{F}} \leq \varepsilon \|A\|_{\mathrm{F}}} \operatorname{rank}(X).$$

Consequently, if $\varepsilon > 0$ then TT-ranks $r_1, \ldots, r_{d-1}$ computed by the resulting procedure are in some sense optimal. However, other rank-revealing factorizations can be usefully adopted for the purpose of computing the TT-format of (an approximation of) the given tensor.

Observe that Algorithm 3.1 is based on a finite iteration. Each iteration is entirely based on matrix computations. In particular, if the argument A is a matrix then the output consists of the two matrices computed by `compress`. In the pseudocode here below, the intermediate matrices $B_1, \ldots, B_{d-1}$ are introduced to improve readability; in a more implementation-oriented description, these variables can share the same name and memory space.

Algorithm 3.1 Full-to-TT compression, iterative version

Input: tensor A of size $n_1 \times n_2 \times \cdots \times n_d$ and local accuracy bound ε
Output: tensor carriages $\mathsf{G}_1, \ldots, \mathsf{G}_d$

1: **function** $[\mathsf{G}_1, \ldots, \mathsf{G}_d] = \textsc{Full\_to\_TT}(\mathsf{A}, \varepsilon)$
2: $n := N(\mathsf{A})$
3: $A_1 := \texttt{reshape}(\mathsf{A}, [n_1, n/n_1])$
4: $[\mathsf{G}_1, B_1, r_1] := \texttt{compress}(A_1, \varepsilon)$
5: **for** $k = 2 \ldots d-1$ **do**
6: $n_R := n_R / n_k$
7: $B_{k-1} := \texttt{reshape}(B_{k-1}, [r_{k-1} n_k, n_R])$
8: $[\mathsf{G}_k, B_k, r_k] := \texttt{compress}(B_{k-1}, \varepsilon)$
9: $\mathsf{G}_k := \texttt{reshape}(\mathsf{G}_k, [r_{k-1}, n_k, r_k])$
10: **end for**
11: $\mathsf{G}_d := B_{d-1}$
12: **end function**

Example 2. Suppose that Algorithm 3.1 is applied to the fourth order tensor A in Example 1. Then Algorithm 3.1 performs three for-loops; the dimensions of the matrices B_1 and B_2 before and after being reshaped in step 7 are as follows:

B_1 : 5×24000 and 100×1200
B_2 : 6×1200 and 180×40

At the third iteration the matrix B_3 is 7×40 and is renamed G_4 without being reshaped.

Algorithm 3.2 is a recursive version of Algorithm 3.1, namely, the two algorithms compute the same function, but Algorithm 3.2 calls itself internally. Termination is ensured by the fact that the order of the tensor argument diminishes by one at each

recursive call. In fact, if $d = 2$ then the sought decomposition is obtained immediately by one call to `compress`, as in the iterative version. When $d > 2$, the matrix B_1 computed in step 4 together with the carriage G_1 is the first unfolding matrix of the $(d-1)$-dimensional tensor B, whose TT decomposition is computed by the recursive call in step 9. In that step the carriage G_2 is obtained as a matrix of dimension $(r_1 n_2) \times r_2$. In the subsequent steps that matrix is reshaped into a tensor of dimension $r_1 \times n_2 \times r_2$ and the computation is complete.

Algorithm 3.2 Full-to-TT compression, recursive version

Input: tensor A of size $n_1 \times n_2 \times \cdots \times n_d$ and local accuracy bound ε
Output: tensor carriages $\mathsf{G}_1, \ldots, \mathsf{G}_d$

1: **function** $[\mathsf{G}_1, \ldots, \mathsf{G}_d] = \text{FULL\_TO\_TT}(A, \varepsilon)$
2: $n := N(\mathsf{A})$
3: $A_1 := \texttt{reshape}(\mathsf{A}, [n_1, n/n_1])$
4: $[\mathsf{G}_1, B_1, r_1] := \texttt{compress}(A_1, \varepsilon)$
5: **if** $d = 2$ **then**
6: $\mathsf{G}_2 := B_1$
7: **else**
8: $\mathsf{B} := \texttt{reshape}(B_1, [r_1 n_2, n_3, \ldots, n_d])$ ▷ the order of B is $d-1$
9: $[\mathsf{G}_2, \ldots, \mathsf{G}_d] := \texttt{Full\_to\_TT}(\mathsf{B}, \varepsilon)$ ▷ recursive call
10: $r_2 := N(\mathsf{G}_2)/(r_1 n_2)$
11: $\mathsf{G}_2 := \texttt{reshape}(\mathsf{G}_2, [r_1, n_2, r_2])$ ▷ G_2 is tensorized
12: **end if**
13: **end function**

Remark 2. Apart of reshaping, the matrix B_k computed in Algorithm 3.1 coincides with the matrix B_1 computed in the $(k-1)$-th recursive call of Algorithm 3.2.

In the subsequent analysis we assume that, in exact arithmetic, the function `compress` satisfies the following property: For any input matrix A, the output matrices M and N fulfil the identities

$$M^T M = I, \qquad M^T (A - MN) = O. \tag{6}$$

Equations (5) and (6) can be met if `compress` is obtained from a truncated SVD as in [9] or from a (truncated) QR factorization. Indeed, the equations in (6) are fulfilled if and only if the matrix A admits a factorization

$$A = (M\ M') \begin{pmatrix} N \\ N' \end{pmatrix}$$

where the matrix $Q = (M\ M')$ has orthonormal columns and N has full rank, in which case we have $\|A - MN\| = \|M'N'\| = \|N'\|$ and $\|N\| = \|M^T A\| \le \|A\|$ in any unitarily invariant norm. The sufficiency of that condition is obvious. To prove necessity, consider the residual matrix $R = A - MN$ and let $R = M'N'$ be a factorization where the columns of M' are an orthonormal basis for the column space of R and N'

has full rank. Since $M^T M' N' = O$ the columns of M' must belong to the kernel of M^T, hence the matrix $Q = (M\ M')$ has orthonormal columns.

The following theorem is a reworking of Theorem 2.2 from [9] to emphasize the role of the hypotheses placed on `compress`. A shorter proof is included for later reference.

Theorem 1. *Suppose that the function* `compress` *in* (4) *fulfills the conditions* (5) *and* (6). *Let* $\mathsf{T} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ *where the tensor carriages* $\mathsf{G}_1, \ldots, \mathsf{G}_d$ *are computed from Algorithm 3.2 in exact arithmetic. Then* $\|\mathsf{A} - \mathsf{T}\|_{\mathrm{F}} \leq \varepsilon \sqrt{d-1}\, \|\mathsf{A}\|_{\mathrm{F}}$.

Proof. We proceed by induction on d. When $d = 2$ the claim follows immediately from (5). For $d > 2$, consider the matrices G_1 and B_1 computed in step 4. By construction, the first unfolding matrix of T admits the factorization $T_1 = G_1 U_1$, for some $r_1 \times (n_2 \cdots n_d)$ matrix U_1. Since $G_1^T (A_1 - G_1 B_1) = O$ and $G_1^T G_1 = I$ by hypothesis, we have

$$\begin{aligned}
\|\mathsf{A} - \mathsf{T}\|_{\mathrm{F}}^2 &= \|A_1 - G_1 B_1 + G_1 B_1 - G_1 U_1\|_{\mathrm{F}}^2 \\
&= \|A_1 - G_1 B_1\|_{\mathrm{F}}^2 + \|G_1 (B_1 - U_1)\|_{\mathrm{F}}^2 + 2\langle G_1^T (A_1 - G_1 B_1), B_1 - U_1 \rangle \\
&= \|A_1 - G_1 B_1\|_{\mathrm{F}}^2 + \|B_1 - U_1\|_{\mathrm{F}}^2 \,.
\end{aligned}$$

Consider the $(d-1)$-dimensional tensor

$$\mathsf{U} = \mathtt{reshape}(U_1, [r_1 n_2, n_3, \ldots, n_d]) \,,$$

whose first unfolding matrix is U_1. Then,

$$\|\mathsf{A} - \mathsf{T}\|_{\mathrm{F}}^2 \leq \varepsilon^2 \|\mathsf{A}\|_{\mathrm{F}}^2 + \|\mathsf{B} - \mathsf{U}\|_{\mathrm{F}}^2 \,,$$

where B is obtained in step 8 of Algorithm 3.2. By construction $\mathsf{U} = \mathrm{TT}(\mathsf{G}_2, \ldots, \mathsf{G}_d)$, that is, U is the tensor whose exact TT-decomposition is given by the tensor carriages obtained from Algorithm 3.2 with input B. By inductive hypothesis we have

$$\|B_1 - U_1\|_{\mathrm{F}}^2 = \|\mathsf{B} - \mathsf{U}\|_{\mathrm{F}}^2 \leq (d-2)\varepsilon^2 \|\mathsf{B}\|_{\mathrm{F}}^2 \,.$$

Moreover,

$$\|\mathsf{B}\|_{\mathrm{F}} = \|B_1\|_{\mathrm{F}} = \|G_1^T A_1\|_{\mathrm{F}} \leq \|A_1\|_{\mathrm{F}} = \|\mathsf{A}\|_{\mathrm{F}} \,,$$

and the claim follows. □

Remark 3. It is not difficult to prove that the hypothesis $M^T M = I$ in (6) can be replaced by $\|M\|_2 \|M^+\|_2 = 1$, where M^+ is the Moore–Penrose inverse of M, without affecting the claim of Theorem 1. In the proof, one has simply to replace $\|G_1(B_1 - U_1)\|_{\mathrm{F}} = \|B_1 - U_1\|_{\mathrm{F}}$ by $\|G_1(B_1 - U_1)\|_{\mathrm{F}} \leq \|G_1\|_2 \|B_1 - U_1\|_{\mathrm{F}}$ and $\|B_1\|_{\mathrm{F}} = \|G_1^T A_1\|_{\mathrm{F}} \leq \|A_1\|_{\mathrm{F}}$ by $\|B_1\|_{\mathrm{F}} = \|G_1^+ A_1\|_{\mathrm{F}} \leq \|G_1^+\|_2 \|A_1\|_{\mathrm{F}}$.

This fact allows to introduce scaling factors in the computed carriages, e.g., in order to balance their norms, with no changes in the error estimate.

3.1 Backward Stability Analysis

The forthcoming Theorem 2 provides a backward stability estimate for Algorithm 3.2 which, in the exact arithmetic case, reduces to the error estimate given in Theorem 1 and, in the floating point arithmetic case, outlines the effects of the tolerance ε, the loss of orthogonality of the matrix M in (4) and the numerical stability of the function `compress` on the backward error of the computed TT decomposition.

Actually, the following analysis is mainly aimed to deal with two issues arising in the practical usage of Algorithm 3.2:

1. We compute an "exact" ($\varepsilon = 0$) TT decomposition in computer arithmetics and we desire a bound on the backward error of the computed result.
2. We want to compute a "low rank" approximation ($\varepsilon > 0$) of the given tensor but the function `compress` does not meet the conditions (6) as it happens if, e.g., it is based not on (pivoted) QR but on a different rank-revealing factorization where the spectral conditioning of M is greater than 1, see e.g., [1, §5.4.5]. In that case, we would like to quantify to what extent the approximation bound in Theorem 1 is degraded.

These two issues can be tackled together by assuming that the function `compress` fulfills the following hypotheses in place of (6): For any $A \in \mathbb{R}^{m\times n}$ there exists an exact decomposition $A + \Delta A = \widehat{Q}\widehat{R}$ with

$$\widehat{Q} = (M\ M') \in \mathbb{R}^{m\times m}, \qquad \widehat{R} = \begin{pmatrix} N \\ N' \end{pmatrix} \in \mathbb{R}^{m\times n}, \tag{7}$$

such that $\|M'N'\|_{\mathrm{F}} \le \varepsilon \|A\|_{\mathrm{F}}$ where ε is the user-specified tolerance; and there exist two functions $\eta_1 = \eta_1(m,n,r)$ and $\eta_2 = \eta_2(m,r)$ such that

$$\|\Delta A\|_{\mathrm{F}} \le \eta_1 \|A\|_{\mathrm{F}}, \qquad \|Q - \widehat{Q}\|_2 \le \eta_2 \tag{8}$$

for some orthogonal matrix $Q \in \mathbb{R}^{m\times m}$.

If computations are performed in the usual IEEE standard computer arithmetic with machine precision u, the existence of two functions η_1 and η_2 being $\mathscr{O}(u)$, fulfilling (8) and having a moderate growth in m,n,r can be inferred from known facts concerning the error analysis of Householder QR factorization [1, Ch. 5], [2, §19.3]. For example, if the matrix $\widehat{Q}$ in (7) is computed by means of $r \le \min\{m,n\}$ Householder transformations then [2, pp. 359–361] brings in the estimates

$$\eta_1(m,n,r) = \mathscr{O}(mru), \qquad \eta_2(m,r) = \mathscr{O}(mr^{3/2}u).$$

These estimates are deemed as rather conservative, and practical experience suggests e.g., that η_1 is a linear polynomial in m and r, see [2, p. 368]. In any case, all quantities η_1 and η_2 occurring in what follows are assumed to be "sufficiently small" so that quadratic and higher order terms can be neglected. We stress that conditions (7) and (8) reduce to (6) when $\eta_1 = \eta_2 = 0$.

Lemma 1. *In the preceding notations and hypotheses, neglecting higher order terms in ε, η_1, and η_2 we have*

1. $\|A-MN\|_{\mathrm{F}} \le (\varepsilon+\eta_1)\|A\|_{\mathrm{F}}$
2. $\|M^T(A-MN)\|_{\mathrm{F}} \le \eta_1\|A\|_{\mathrm{F}}$
3. $\|N\|_{\mathrm{F}} \le (1+\eta_1+\eta_2)\|A\|_{\mathrm{F}}$.

Proof. Firstly, we have

$$\|A-MN\|_{\mathrm{F}} = \|A+\Delta A-\Delta A-MN\|_{\mathrm{F}} = \|M'N'-\Delta A\|_{\mathrm{F}} \le (\varepsilon+\eta_1)\|A\|_{\mathrm{F}}\,,$$

and the first part follows.

Let $Q,\widehat{Q}\in\mathbb{R}^{m\times m}$ be the matrices appearing in (7) and (8). Let $Q=(Q_1\ Q_2)$ and $\Delta Q=\widehat{Q}-Q=(\Delta_1\ \Delta_2)$ be partitioned consistently with $\widehat{Q}$ as in (7). Then, $\|\Delta_{1,2}\|_2\le\eta_2$ and

$$\begin{aligned}\|M^TM'\|_2 &= \|(Q_1+\Delta_1)^T(Q_2+\Delta_2)\|_2\\ &\le \|\Delta_1^TQ_2\|_2+\|Q_1^T\Delta_2\|_2+\|\Delta_1^T\Delta_2\|_2\\ &\le \|\Delta_1\|_2+\|\Delta_2\|_2+\|\Delta_1^T\Delta_2\|_2\le\eta_2(2+\eta_2)\,.\end{aligned}$$

Neglecting quadratic terms we get $\|M^TM'\|_2\lesssim 2\eta_2$, $\|N'\|_{\mathrm{F}}\lesssim\varepsilon\|A\|_{\mathrm{F}}$ and

$$\begin{aligned}\|M^T(A-MN)\|_{\mathrm{F}} &= \|M^T(M'N'-\Delta A)\|_{\mathrm{F}}\\ &\le \|M^TM'\|_2\|N'\|_{\mathrm{F}}+\|M\|_2\|\Delta A\|_{\mathrm{F}}\lesssim(2\eta_2\varepsilon+\eta_1)\|A\|_{\mathrm{F}}\,.\end{aligned}$$

This proves the second inequality in the claim. Finally,

$$\begin{aligned}\|N\|_{\mathrm{F}}\le\|\widehat{R}\|_{\mathrm{F}} &= \|\widehat{Q}^{-1}(A+\Delta A)\|_{\mathrm{F}}\\ &= \|\widehat{Q}^{-1}QQ^T(A+\Delta A)\|_{\mathrm{F}}\\ &\le \|\widehat{Q}^{-1}(\widehat{Q}-\Delta Q)\|_2\|A+\Delta A\|_{\mathrm{F}}\lesssim(1+\eta_1)(1+\eta_2)\|A\|_{\mathrm{F}}\,,\end{aligned}$$

and the proof is complete. □

Theorem 2. *Let $\mathsf{T}=\mathrm{TT}(\mathsf{G}_1,\dots,\mathsf{G}_d)$ where the carriages $\mathsf{G}_1,\dots,\mathsf{G}_d$ are computed under the aforementioned hypotheses. Let η_1 and η_2 denote the largest values of the like named coefficients occurring in all recursive calls of Algorithm 3.2 on the specific input A. Moreover, let $\hat{\varepsilon}=\varepsilon+\eta_1$. Neglecting higher order terms in ε, η_1, and η_2,*

$$\|\mathsf{A}-\mathsf{T}\|_{\mathrm{F}}\le(1+\eta_1+2\eta_2)^{d-2}\big((d-2)\eta_1+\sqrt{d-1}\,\hat{\varepsilon}\big)\|\mathsf{A}\|_{\mathrm{F}}\,.$$

Proof. We proceed by induction, as in the the proof of Theorem 1. The $d=2$ case is an immediate consequence of Lemma 1(1):

$$\|\mathsf{A}-\mathsf{T}\|_{\mathrm{F}}=\|A_1-G_1B_1\|_{\mathrm{F}}\le(\varepsilon+\eta_1)\|A_1\|_{\mathrm{F}}=\hat{\varepsilon}\|\mathsf{A}\|_{\mathrm{F}}.$$

For $d > 2$, the inductive argument begins similarly to the proof of Theorem 1. Indeed, by hypotheses and Lemma 1 we have $\|G_1\|_2 \le 1+\eta_2$ and

$$\begin{aligned}\|\mathsf{A}-\mathsf{T}\|_{\mathrm{F}}^2 &= \|A_1 - G_1B_1 + G_1B_1 - G_1U_1\|_{\mathrm{F}}^2 \\ &= \|A_1 - G_1B_1\|_{\mathrm{F}}^2 + \|G_1(B_1-U_1)\|_{\mathrm{F}}^2 + 2\langle G_1^T(A_1-G_1B_1), B_1-U_1\rangle \\ &\le \hat{\varepsilon}^2\|\mathsf{A}\|_{\mathrm{F}}^2 + (1+\eta_2)^2\|B_1-U_1\|_{\mathrm{F}}^2 + 2\eta_1\|\mathsf{A}\|_{\mathrm{F}}\|B_1-U_1\|_{\mathrm{F}}\,.\end{aligned}$$

The last inequality follows by Cauchy–Schwartz inequality and Lemma 1(2).

Let $\mathsf{A} = \mathsf{A}_d$ and $\mathsf{T} = \mathsf{T}_d$. For $k = 2,\dots,d-1$, let $\mathsf{A}_k, \mathsf{T}_k$ be the k-dimensional tensors defined as follows: A_k and T_k are the argument and the result of the $(d-k)$-th recursive call of the algorithm,[1]

$$(\mathsf{G}_{d-k+1},\dots,\mathsf{G}_d) = \texttt{Full\_to\_TT}(\mathsf{A}_k,\varepsilon)\,, \qquad \mathsf{T}_k = \mathrm{TT}(\mathsf{G}_{d-k+1},\dots,\mathsf{G}_d)\,.$$

For example, in the proof of Theorem 1 A_{d-1} is denoted by B and T_{d-1} by U. With these notations, the preceding inequality yields

$$\|\mathsf{A}_{k+1}-\mathsf{T}_{k+1}\|_{\mathrm{F}}^2 \le \hat{\varepsilon}^2\|\mathsf{A}_{k+1}\|_{\mathrm{F}}^2 + (1+\eta_2)^2\|\mathsf{A}_k-\mathsf{T}_k\|_{\mathrm{F}}^2 + 2\eta_1\|\mathsf{A}_{k+1}\|_{\mathrm{F}}\|\mathsf{A}_k-\mathsf{T}_k\|_{\mathrm{F}}\,.$$

Let $\rho = 1+\eta_1+\eta_2$. From Lemma 1(3) we have $\|\mathsf{A}_k\|_{\mathrm{F}} \le \rho\|\mathsf{A}_{k+1}\|_{\mathrm{F}}$. For $k = 1,\dots,d-1$ let $e_k = \|\mathsf{A}_{k+1}-\mathsf{T}_{k+1}\|/\|\mathsf{A}_{k+1}\|_{\mathrm{F}}$. Neglecting product terms in η_1, η_2 and other small quantities we arrive at the recurrence

$$\begin{aligned}e_k^2 &\le \hat{\varepsilon}^2 + \rho^2(1+\eta_2)^2e_{k-1}^2 + 2\eta_1\rho e_{k-1} \\ &\le (\alpha e_{k-1}+\eta_1)^2 + \hat{\varepsilon}^2\end{aligned}$$

where $\alpha = 1+\eta_1+2\eta_2$ and $e_1 \le \hat{\varepsilon}$. From Lemma 5 in Appendix we obtain

$$e_k \le \alpha^{k-1}\left((k-1)\eta_1 + \sqrt{k}\,\hat{\varepsilon}\right)\,,$$

and the claim follows by setting $k = d-1$. □

It is worth noting that in the exact case (that is, when $\eta_1 = \eta_2 = 0$) the inequality in the previous claim reduces to that of Theorem 1.

4 Computing Multilinear Forms

The computation of the multilinear form (or contraction)

$$\alpha = \mathsf{A} \times_1 v^{(1)} \times_2 v^{(2)} \times_3 \cdots \times_d v^{(d)}$$

[1] With a little abuse of notation, in the ensuing equation we identify G_{d-k+1} with its first unfolding matrix.

where $v^{(i)} \in \mathbb{R}^{n_i}$, occurs e.g., in the computation of multidimensional integrals on cartesian product grids [9]. If $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$ then the preceding expression can be rewritten as

$$\alpha = \sum_{i_1,\ldots,i_d} \sum_{j_1,\ldots,j_{d-1}} \mathsf{G}_1(i_1, j_1)\mathsf{G}_2(j_1, i_2, j_2) \cdots \mathsf{G}_d(j_{d-1}, i_d) v_{i_1}^{(1)} \cdots v_{i_d}^{(d)} .$$

Assuming $n_1 = \ldots = n_d = n$ and $r_1 = \ldots = r_{d-1} = r$, the right hand side can be computed in $\mathscr{O}(dnr^2)$ floating point operations using Algorithm 3 from [9], which is described hereafter as Algorithm 4.1.

Algorithm 4.1 Fast TT contraction algorithm

Input: tensor train $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$, vectors $v^{(1)}, \ldots, v^{(d)}$
Output: $\alpha = \mathsf{A} \times_1 v^{(1)} \times_2 \cdots \times_d v^{(d)}$
1: **function** $\alpha = \mathrm{TT\_CONTRACTION}(\mathsf{G}_1, \ldots, \mathsf{G}_d, v^{(1)}, \ldots, v^{(d)})$
2: $t := \mathsf{G}_1 \times_1 v^{(1)}$
3: **for** $k = 2 \ldots d$ **do**
4: $W := \mathsf{G}_k \times_1 t$
5: $t := W^T v^{(k)}$
6: **end for**
7: $\alpha := t$.
8: **end function**

After completion of the k-th cycle of the for-loop, W is an $n_k \times r_k$ matrix and t is an r_k-vector. In particular, when $k = d$ step 4 is a matrix-vector multiplication as $r_d = 1$, so W becomes a vector and step 5 is an inner product of two n_d-vectors. Note that the computation of $W = \mathsf{G}_k \times_1 t$ followed by $t = W^T v^{(k)}$ yields a particular algorithm to compute $\mathsf{G}_k \times_1 t \times_2 v^{(k)}$ which can be implemented using $n_k r_k$ inner products with r_{k-1}-vectors followed by r_k inner products with n_k-vectors. Owing to the identity (3), an alternative algorithm for computing the same expression (with almost the same number of floating point operations) is $W := \mathsf{G}_k \times_2 v^{(k)}$ followed by $t := W^T t$.

In what follows, $\mathbb{F}$ denotes a set of (computer) floating point numbers endowed by the usual IEEE standard arithmetics, and u denotes the corresponding unit roundoff. Moreover, we use the notation $\mathrm{fl}(\cdot)$ with an argument that is an expression to denote the computed value of that expression. The next two lemmas are borrowed from [3].

Lemma 2. *Given $x, y \in \mathbb{F}^n$, any order of evaluation of the inner product $x^T y$ produces an approximation α such that $|\alpha - x^T y| \leq nu|x|^T|y|$, if no underflows or overflows are encountered.*

Lemma 3. *Given $A \in \mathbb{F}^{m\times n}$ and $x \in \mathbb{F}^n$, let $\widehat{y} = \mathrm{fl}(Ax)$ be the approximation to Ax obtained by computing m inner products of n-vectors, each of them being performed in an arbitrary evaluation order. If no underflow or overflow occurs, then there exists a matrix $\widehat{A} \in \mathbb{R}^{m\times n}$ such that*

$$\widehat{y} = \widehat{A}x, \qquad \widehat{A}_{ij} = (1+\varepsilon_{ij})A_{ij}, \qquad |\varepsilon_{ij}| \le nu\ .$$

On the basis of these two lemmas it is not hard to obtain the result hereafter.

Theorem 3. *Given $\mathsf{G} \in \mathbb{F}^{\ell\times m\times n}$, $x \in \mathbb{F}^{\ell}$, and $y \in \mathbb{F}^{m}$, let $\widehat{z} = \mathrm{fl}(\mathrm{fl}(\mathsf{G}\times_1 x)\times_2 y) \in \mathbb{F}^{n}$ be the finite precision approximation to $z = \mathsf{G}\times_1 x\times_2 y$ obtained after $mn+n$ inner products, each of them being performed in an arbitrary evaluation order. If no underflow or overflow occurs then there exists a tensor $\Delta\mathsf{G} \in \mathbb{F}^{\ell\times m\times n}$ such that*

$$\widehat{z} = (\mathsf{G}+\Delta\mathsf{G})\times_1 x\times_2 y, \qquad |\Delta\mathsf{G}| \le (\ell+m)u|\mathsf{G}| + \mathscr{O}(u^2)\ .$$

Proof. Introduce the auxiliary matrix $M = \mathsf{G}\times_1 x$ and let $\widehat{M} = \mathrm{fl}(\mathsf{G}\times_2 x)$ be its finite precision counterpart. In exact arithmetic,

$$M_{jk} = \sum_{i=1}^{\ell} \mathsf{G}_{ijk}x_i\ .$$

Owing to Lemma 2, for every j,k there exists ε_{jk} such that

$$\widehat{M}_{jk} - M_{jk} = \varepsilon_{jk}\sum_{i=1}^{\ell}|\mathsf{G}_{ijk}x_i|\ , \qquad |\varepsilon_{jk}| \le \ell u\ .$$

Letting $\eta_{ijk} = \mathrm{sign}(\mathsf{G}_{ijk}x_i)\varepsilon_{jk}$ we obtain

$$\begin{aligned}\widehat{M}_{jk} &= \sum_i \mathsf{G}_{ijk}x_i + \varepsilon_{jk}|\mathsf{G}_{ijk}x_i| \\ &= \sum_i (1+\eta_{ijk})\mathsf{G}_{ijk}x_j\ .\end{aligned}$$

Obviously $|\eta_{ijk}| \le \ell u$. By Lemma 3, for $k = 1,\dots,n$ we have

$$\begin{aligned}\widehat{z}_k = [\mathrm{fl}(\widehat{M}^T y)]_k &= \sum_j (1+\eta_{jk})\widehat{M}_{jk}y_j \\ &= \sum_j\sum_i (1+\eta_{jk})(1+\eta_{ijk})\mathsf{G}_{ijk}x_iy_j\end{aligned}$$

for some constants η_{jk} with $|\eta_{jk}| \le mu$. In conclusion, $\widehat{z} = (\mathsf{G}+\Delta\mathsf{G})\times_1 x\times_2 y$ where

$$\Delta\mathsf{G}_{ijk} = \xi_{ijk}\mathsf{G}_{ijk}\ , \qquad |\xi_{ijk}| = |(1+\eta_{jk})(1+\eta_{ijk})-1| \le (\ell+m)u + \ell mu^2\ ,$$

and the proof is complete. □

Note that the previous theorem applies also when $\ell = 1$ or $n = 1$, where G reduces to a matrix. Recalling the conventional notation $r_0 = 1$, we obtain immediately the following consequence.

Corollary 1. *Let $\widehat{\alpha}$ be the result of Algorithm 4.1 computed in machine arithmetics from input $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1,\dots,\mathsf{G}_d)$. Then, there exist tensors $\Delta G_1,\dots,\Delta G_d$ such that*

$$\widehat{\alpha} = \widehat{\mathsf{A}} \times_1 v^{(1)} \times_2 v^{(2)} \cdots \times_d v^{(d)}, \qquad \widehat{\mathsf{A}} = \mathrm{TT}(\mathsf{G}_1 + \Delta\mathsf{G}_1, \ldots, \mathsf{G}_d + \Delta\mathsf{G}_d)\,,$$

and $|\Delta\mathsf{G}_i| \leq (n_i + r_{i-1})u|\mathsf{G}_i| + \mathscr{O}(u^2)$.

The previous result allows us to interpret the rounding errors in the computation of α as due to perturbations in the carriages $\mathsf{G}_1, \ldots, \mathsf{G}_d$, not in A. The forthcoming Theorem 4 provides a backward error bound in terms of a perturbation in A. To that goal, we need the the following lemma whose proof derives from an elementary inductive argument and will not be shown.

Lemma 4. *Let* $\xi_1, \ldots \xi_k$ *be numbers such that* $s = \sum_{i=1}^k |\xi_i| < 1$. *Then,*

$$\prod_{i=1}^{k}(1+\xi_i) = 1 + \theta, \qquad |\theta| \leq \frac{s}{1-s}\,.$$

In particular, if $s \leq \frac{1}{2}$ *then* $|\theta| \leq 2s$.

Theorem 4. *Let* $\widehat{\alpha}$ *be the result of Algorithm 4.1 computed in machine arithmetic from input* $\mathsf{A} = \mathrm{TT}(\mathsf{G}_1, \ldots, \mathsf{G}_d)$, *and let* $s = \sum_{i=1}^d (n_i + r_{i-1})u$. *If* $s \leq \frac{1}{2}$ *then*

$$\widehat{\alpha} = \widehat{\mathsf{A}} \times_1 v^{(1)} \times_2 v^{(2)} \cdots \times_d v^{(d)}$$

where $|\widehat{\mathsf{A}} - \mathsf{A}| \leq 2s\,\mathrm{TT}(|\mathsf{G}_1|, \ldots, |\mathsf{G}_d|) + \mathscr{O}(u^2)$.

Proof. In what follows, we interpret $\mathsf{G}_1(i,j)$ and $\mathsf{G}_d(i,j)$ as $\mathsf{G}_1(1,i,j)$ and $\mathsf{G}_d(i,j,1)$, respectively. Let $\varepsilon_i = (n_i + r_{i-1})u$ for $i = 1, \ldots, d$. By Corollary 1, there exist constants $\xi_{ijk}^{(\ell)}$ such that $\widehat{\alpha}$ is the exact contraction of the tensor $\mathrm{TT}(\mathsf{G}_1 + \Delta\mathsf{G}_1, \ldots, \mathsf{G}_d + \Delta\mathsf{G}_d)$ and vectors $v^{(1)}, \ldots, v^{(d)}$ where

$$\Delta\mathsf{G}_\ell(i,j,k) = \mathsf{G}_\ell(i,j,k)\xi_{ijk}^{(\ell)}, \qquad |\xi_{ijk}^{(\ell)}| \leq \varepsilon_\ell + \mathscr{O}(u^2)\,.$$

Assuming $j_0 = j_d = 1$, for every multi-index $i = (i_1, i_2, \ldots, i_d)$ we have

$$\begin{aligned}\widehat{\mathsf{A}}(i) &= \sum_{j_1,\ldots,j_{d-1}} \mathsf{G}_1(i_1,j_1)\mathsf{G}_2(j_1,i_2,j_2)\cdots G_d(j_{d-1},i_d)\prod_{\ell=1}^{d}(1+\xi_{j_{\ell-1}i_\ell j_\ell}^{(\ell)}) \\ &= \sum_{j_1,\ldots,j_{d-1}} \mathsf{G}_1(i_1,j_1)\mathsf{G}_2(j_1,i_2,j_2)\cdots \mathsf{G}_d(j_{d-1},i_d)(1+\theta_x)\end{aligned}$$

with $x = (i_1, \ldots, i_d, j_1, \ldots, j_{d-1})$ and $|\theta_x| \leq 2s$ by Lemma 4. By triangle inequality,

$$|\widehat{\mathsf{A}}(i) - \mathsf{A}(i)| \leq \sum_{j_1,\ldots,j_{d-1}} |\mathsf{G}_1(i_1,j_1)||\mathsf{G}_2(j_1,i_2,j_2)|\cdots|\mathsf{G}_d(j_{d-1},i_d)||\theta_x|\,,$$

and the claim follows. □

Appendix

Hereafter, we prove a technical lemma which yields an upper bound for the growth of a sequence occurring within the proof of Theorem 2.

Lemma 5. *Let $\{e_k\}$ be a sequence of nonnegative numbers such that $e_1 \le \gamma$ and for $k \ge 2$*

$$e_k^2 \le (\alpha e_{k-1} + \beta)^2 + \gamma^2$$

for some nonnegative constants α, β, γ with $\alpha \ge 1$. Then for all $k \ge 1$

$$e_k \le \alpha^{k-1}\big((k-1)\beta + \sqrt{k}\,\gamma\big)\ .$$

Proof. Define the auxiliary notations $\hat{e}_k = \alpha^{1-k} e_k$, $\hat{\beta}_k = \alpha^{1-k}\beta$ and $\hat{\gamma}_k = \alpha^{1-k}\gamma$. Note that $\hat{\beta}_k \le \beta$ and $\hat{\gamma}_k \le \gamma$ for $k \ge 1$. Then,

$$\begin{aligned} \hat{e}_k^2 = \alpha^{2-2k} e_k^2 &\le (\alpha^{2-k} e_{k-1} + \alpha^{1-k}\beta)^2 + \alpha^{2-2k}\gamma^2 \\ &= (\hat{e}_{k-1} + \hat{\beta}_k)^2 + \hat{\gamma}_k^2\ . \end{aligned}$$

Firstly we prove that for all $k \ge 1$

$$\hat{e}_k \le \textstyle\sum_{j=2}^{k} \hat{\beta}_k + \sqrt{\sum_{j=1}^{k} \hat{\gamma}_j^2}\ .$$

Indeed, the claim is trivially verified when $k = 1$. By an inductive argument, for $k \ge 2$ we have

$$\begin{aligned} \hat{e}_k^2 &\le \Big(\textstyle\sum_{j=2}^{k} \hat{\beta}_j + \sqrt{\sum_{j=1}^{k-1} \hat{\gamma}_j^2}\Big)^2 + \hat{\gamma}_k^2 \\ &= \big(\textstyle\sum_{j=2}^{k} \hat{\beta}_j\big)^2 + \sum_{j=1}^{k} \hat{\gamma}_j^2 + 2\big(\sum_{j=2}^{k} \hat{\beta}_j\big)\sqrt{\sum_{j=1}^{k-1} \hat{\gamma}_j^2} \\ &\le \Big(\textstyle\sum_{j=2}^{k} \hat{\beta}_j + \sqrt{\sum_{j=1}^{k} \hat{\gamma}_j^2}\Big)^2. \end{aligned}$$

Going back to the sequence $\{e_k\}$ we have for all $k \ge 1$

$$\begin{aligned} e_k = \alpha^{k-1}\hat{e}_k &\le \alpha^{k-1}\Big(\textstyle\sum_{j=2}^{k} \hat{\beta}_j + \sqrt{\sum_{j=1}^{k} \hat{\gamma}_j^2}\Big) \\ &\le \alpha^{k-1}\big((k-1)\beta + \sqrt{k}\,\gamma\big) \end{aligned}$$

and we are done. □

Acknowledgements The first author acknowledges the support received by INDAM-GNCS, Italy, for his research. The work of the second author was supported by the Russian Scientific Foundation project 14-11-00806.

References

1. Golub, G.H., Van Loan, C.: Matrix Computations—4th edition. The John Hopkins University Press (2013)
2. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. SIAM (2002)
3. Jeannerod, C.-P., Rump, S.M.: Improved error bounds for inner products in floating-point arithmetic. SIAM J. Matrix Anal. Appl. **34**, 338–344 (2013)
4. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
5. Lee, N., Cichocki, A.: Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format. SIAM J. Matrix Anal. Appl. **36**(3), 994–1014 (2015)
6. Oseledets, I.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
7. Oseledets, I., Dolgov, S.V.: Solution of linear systems and matrix inversion in the TT-format. SIAM J. Sci. Comput. **34**(5), A2718–A2739 (2012)
8. Oseledets, I., Tyrtyshnikov, E.: Recursive decomposition of multidimensional tensors. Dokl. Math. **80**(1), 460–462 (2009)
9. Oseledets, I., Tyrtyshnikov, E.: TT-cross approximation for multidimensional arrays. Lin. Alg. Appl. **432**(1), 70–88 (2010)