BIT Numerical Mathematics (2025) 65:45
https://doi.org/10.1007/s10543-025-01086-5

BIT

On some orthogonalization schemes in Tensor Train format

Olivier Coulaud[1] · **Luc Giraud**[1] · **Martina Iannacito**[2]

Received: 20 June 2024 / Accepted: 16 October 2025

Abstract
In the framework of tensor spaces, we consider orthogonalization algorithms to generate an orthogonal basis of a tensor subspace from a set of linearly independent tensors. All variants, except for the Householder transformation, are straightforward extensions of well-known algorithms in matrix computation to tensors. In particular, we experimentally study the loss of orthogonality of six orthogonalization methods: Classical and Modified Gram-Schmidt with (CGS2, MGS2) and without (CGS, MGS) re-orthogonalization, the Cholesky-QR, and the Householder transformation. To overcome the curse of dimensionality, we represent tensors with a low-rank approximation using the Tensor Train (TT) formalism. Additionally, we introduce recompression steps in the standard algorithm outline through the TT-round method at a prescribed accuracy. After describing the structure and properties of the algorithms, we illustrate their loss of orthogonality with numerical experiments. Although no formal proof exists at this time, we observe very clearly that the well-established properties verified over decades of research by the round error analysis community in matrix computation appear to extend to the case of low-rank tensors, with the unit round-off replaced by the TT-round accuracy. The computational analysis for each orthogonalization scheme in terms of memory requirements and computational complexity, measured as a function of the number of TT-round operations, which happens to be the most computationally expensive operation, completes the study.

Keywords Classical Gram-Schmidt · Modified Gram-Schmidt · Householder transformation · Loss of orthogonality · Tensor Train format

Mathematics Subject Classification 15A69 (Multilinear algebra, tensor product) · 65G50(Roundoff error)

Dedicated to Professor Åke Björck on the occasion of his 90th birthday, in recognition of his inspiring and outstanding scientific contributions, and his constant, unwavering kindness as a person.

✉ Martina Iannacito
martina.iannacito@unibo.it

1 Concace, Inria Center at the University of Bordeaux, Talence, France

2 Dipartimento di Matematica and (AM)^2, Alma Mater Studiorum Università di Bologna, Piazza di Porta San Donato 5, I-40127 Bologna, Italy

Published online: 31 October 2025

1 Introduction

The solution of linear problems is at the center of many large-scale simulations in academic or industrial applications. Many numerical linear algebra algorithms rely on an orthonormal basis of the space in which the solution is sought; this is particularly the case in GMRES, one of the most popular Krylov subspace methods for solving linear systems, or all variants of the Arnoldi algorithms for computing eigenpairs [1, 2]. The orthonormal basis is built from a set of vectors that are explicitly orthonormalized by an orthogonalization procedure. Various orthogonalization algorithms have been proposed to perform this task over the years. Additionally, they allow for the computation of the matrix QR factorization. If the input consists of m vectors of $\mathbb{R}^n$ organized as the columns of a matrix $A \in \mathbb{R}^{n\times m}$, then the orthogonalization schemes can factorize A into the product of an orthogonal matrix $Q \in \mathbb{R}^{n\times m}$ and an upper triangular one $R \in \mathbb{R}^{m\times m}$. Among the most widely used numerical algorithms, we consider the Classical Gram-Schmidt (CGS) [3, 4], the Modified Gram-Schmidt (MGS) [3, 4], their variants with re-orthogonalization, named CGS2 and MGS2 [5–7], the Cholesky-QR [8] and the Householder transformations [9]. CGS and MGS are algorithms that implement the Gram-Schmidt method. The fundamental idea is to sequentially remove the projection of an input vector along the previously computed orthonormal vectors and eventually normalize it. The CGS2 and MGS2 procedures aim to improve the quality of the CGS and MGS basis vectors by orthogonalizing them once more in the same way as the basis computed with CGS and the MGS, respectively. The Cholesky-QR calculates the orthogonal basis by utilizing the Cholesky factorization of the Gram matrix, which is defined by the inner products of input vectors. The Householder transformation relies on orthogonal reflections constructed from the input vectors and used to reflect the canonical basis.

A crucial aspect of finite precision calculation for orthogonalization algorithms is the *loss of orthogonality* in the computed basis due to computational rounding errors. This issue has been extensively studied over the years, resulting in numerous findings. The research articles present many theoretical results that relate the loss of orthogonality to the linear dependency of the input vectors. The authors of [10–12] establish theoretical bounds for CGS and MGS loss of orthogonality, showing that the basis produced by MGS is better in terms of orthogonality than the CGS one. In [12] for CGS2 and MGS2, it is confirmed that this re-orthogonalization effectively improves the orthogonality of the computed basis. Bounds for the loss of orthogonality of the Householder transformation and the Cholesky-QR are proven in [13] and [8], respectively. The pioneering work of Åke Björck on the analysis of the numerical stability of orthogonalization algorithms forms the backbone of several contributions. Collectively, these theoretical results span decades old: from the 1960s results by Wilkinson and Björck to the more recent ones presented in 2006 by Barlow, Langou, and Smoktunowicz. Nevertheless, Björck's theoretical insights remain highly influential, continue to inspire contemporary research, and are likely to gain renewed relevance with the increasing prominence of multiple-precision arithmetic on future computing architectures.

All of the cited algorithms translate naturally into the tensor world. Starting from a set of m tensors of $\mathbb{R}^{n_1\times\cdots\times n_d}$, an orthogonal basis for the relative subspace of

dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ is produced. These orthogonalization schemes can work with dense tensors, but they are affected by the "curse of dimensionality", i.e., their storage and operation costs grow exponentially with the order of the tensor. Therefore, it is necessary to represent the tensor in a compressed format. In this work, we generalize the six orthogonalization algorithms previously mentioned to tensors using the Tensor Train (TT) formalism [14, 15]. In the scientific computing community working with tensors, the TT formalism is a common choice because of its properties. It is numerically stable, as it relies on QR and SVD, differently for Canonical Polyadic decomposition. For tensors of order-d and size n, it requires $\mathcal{O}(dnr^2)$ units of memory, assuming r as the maximum TT-rank. The storage is linear in the tensor order, differently from the Tucker decomposition. Finally, the TT-format can be described easily without introducing other mathematical objects, differently from the Hierarchical Tucker. However, the sequences operations between tensors in TT-formats reduce the benefit of this compressed representation. Therefore, we introduce additional compression steps by the `TT-round` function [14] in the orthogonalization schemes, knowing that they affect the orthogonality quality of the basis.

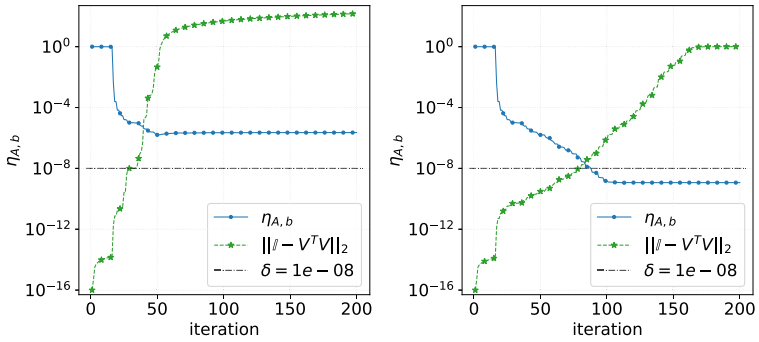
These orthogonalization schemes in TT-format find application in tensor algorithms for solving multilinear systems, such as TT-GMRES [16, 17], or for computing eigenvalues of multilinear operators, as in TT-FEAST [18]. The TT-MGS algorithm is used in TT-FEAST to estimate the number of eigenvalues in the considered contour, as existing matrix estimators cannot be extended to the tensor context. In TT-GMRES, we need to orthogonalize the Krylov basis built by the Arnoldi algorithm, and the choice of the orthogonalization scheme is crucial for ensuring the prescribed accuracy is reached. For example, we consider the discretization on a 16-point grid for each mode of the following PDE problem

$$\begin{cases} -\frac{1}{20}\Delta u - \frac{\partial u}{\partial x} - \frac{1}{250}\frac{\partial u}{\partial y} = 0 \quad \text{in} \quad \Omega = [-1, 1]^3 \,, \\ u_{\{y=1\}} = 1 \qquad \text{and} \qquad u_{\partial\Omega\setminus\{y=1\}} = 0 \,. \end{cases}$$

We solve it with TT-GMRES using TT-CGS and TT-MGS with rounding precision $\delta = 1e-8$.

Figure 1 displays the convergence history of the backward error together with the loss of orthogonality of the Arnoldi basis. We observe that the two backward errors stagnate at different levels. These experimental results are consistent with the matrix theoretical results found in [19], where the attainable accuracy for the backward error is linked to the orthogonalization scheme chosen. Finally, theoretical bounds on the loss of orthogonality form the backbone of convergence and stability proofs in matrix numerical analysis. The aim of this work is to describe the orthogonalization algorithms in TT-format and to experimentally investigate their loss of orthogonality. These purely experimental results in TT-format provide the first numerical evidence that orthogonalization algorithm properties persist in the TT-format. They represent a first step toward a rigorous theoretical analysis, which could support the extension of existing convergence proofs to tensor-based algorithms.

We should emphasize that the theoretical studies on the loss of orthogonality developed by the numerical linear algebra community are based on the analysis of per-



(a) TT-CGS (b) TT-MGS

Fig. 1 Convergence history of the backward error and loss of orthogonality for TT-GMRES using different orthogonalization schemes

turbations of elementary operations performed particularly on the elements of matrices and vectors. The proof techniques used are closely tied to the fundamental assumption that perturbations are component-wise. In our work, we are interested in situations where we do not have control over component-wise perturbations of TT-tensors but only over the norm of perturbations applied to them when they are rounded with a prescribed threshold. The identification of the most appropriate analytical tools for conducting this stability study with norm-wise perturbation represents a significant and ongoing research endeavor, which is beyond the scope of this work.

The rest of the paper is organized as follows. In Section 2, we introduce the notation and recall the most important properties of the TT-format. Section 3 begins with a description of the six orthogonalization schemes extended to the tensor context by the TT-formalism. We also address the complexity in terms of the number of `TT-round` applications, which is the most computationally expensive operation. In Section 3.4 we recall briefly the known theoretical bounds related to the loss of orthogonality of these schemes in classical matrix computation. The theoretical results are linked to the numerical experiments, collected in Section 4, of the same orthogonalization schemes extended with the TT-format. The similarities between the classical orthogonalization algorithms and their TT-versions are summarized in Section 5.

2 Notation and TT-format

To enhance readability, we utilize the following notation for the various mathematical objects described. Small Latin letters represent scalars and vectors (e.g., a), with the context clarifying the object's nature. Capital Latin letters denote matrices (e.g., A), while bold small Latin letters denote tensors (e.g., $\mathbf{a}$). Calligraphic capital letters represent sets (e.g., $\mathcal{A}$). We use the 'Matlab notation' to indicate all the indices along a mode with a colon (':'). For example, if we are given a matrix $A \in \mathbb{R}^{m\times n}$, then $A(:, i)$ represents the i-th column of A. The tensor product is denoted by $\otimes$, while the Euclidean inner product is denoted by $\langle\cdot,\cdot\rangle$ for both vectors and tensors. We use $||\cdot||$ to

denote the Euclidean norm for vectors and the Frobenius norm for matrices and tensors. The condition number of a matrix $A \in \mathbb{R}^{n\times n}$ is denoted by $\kappa(A) = \|A\|\|A^{-1}\|$.

Let $\mathbf{x}$ be a d-order tensor in $\mathbb{R}^{n_1\times\cdots\times n_d}$ and n_k the dimension of mode k for every $k \in \{1,\dots,d\}$. Storing the full tensor $\mathbf{x} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ has a memory cost of $\mathcal{O}(n^d)$ with $n = \max_{i\in\{1,\dots,d\}}\{n_i\}$. Therefore, various compression techniques have been proposed over the years to reduce the memory consumption [20–22]. For the purpose of this work, the most suitable tensor representation is the *Tensor Train* (TT) format [22]. The main concept of TT is to represent a d-order tensor as the contraction of d 3-order tensors. This contraction is a generalization of the matrix-vector product to tensors.

The Tensor Train representation of $\mathbf{x} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is

$$\mathbf{x} = \underline{\mathbf{x}}_1\underline{\mathbf{x}}_2\cdots\underline{\mathbf{x}}_d,$$

where $\underline{\mathbf{x}}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$ is called k-th *TT-core* for $k \in \{1,\dots,d\}$, with $r_0 = r_d = 1$. Note that $\underline{\mathbf{x}}_1 \in \mathbb{R}^{r_0\times n_1\times r_1}$ and $\underline{\mathbf{x}}_d \in \mathbb{R}^{r_{d-1}\times n_d\times r_d}$ reduce essentially to matrices, but for consistency in notation, we represent them as tensors. To ensure clarity, we denote the k-th TT-core of a tensor by the same bold letter underlined with a subscript k. The adopted notation for TT-cores is non-standard, differing from the commonly used notation, see for example [22]. This choice is made to distinguish the k-th TT-core, $\underline{\mathbf{x}}_k$, from the k-th TT-vector in a sequence, $\mathbf{x}_k$. The value r_k is called *k-th TT-rank*.

Given an index i_k, we denote the i_k-th matrix slice of $\underline{\mathbf{x}}_k$ with respect to mode 2 by $\underline{X}_k(i_k)$, i.e., $\underline{X}_k(i_k) = \underline{\mathbf{x}}_k(:,i_k,:)$. Each element of the TT-tensor $\mathbf{x}$ can be expressed as the product of d matrices, i.e.,

$$\mathbf{x}(i_1,\dots,i_d) = \underline{X}_1(i_1)\cdots\underline{X}_d(i_d)$$

with $\underline{X}_k(i_k) \in \mathbb{R}^{r_{k-1}\times r_k}$ for every $i_k \in \{1,\dots,n_k\}$ and $k \in \{2,\dots,d-1\}$, while $\underline{X}_1(i_1) \in \mathbb{R}^{1\times r_1}$ and $\underline{X}_d(i_d) \in \mathbb{R}^{r_{d-1}\times 1}$. It is important to note that $\underline{X}_1(i_1)$ and $\underline{X}_d(i_d)$ are actually vectors, but for the sake of consistency, they are written as matrices with a single row or column.

Storing a tensor in TT-format requires $\mathcal{O}(dnr^2)$ units of memory, where $n = \max_{i\in\{1,\dots,d\}}\{n_i\}$ and $r = \max_{i\in\{1,\dots,d\}}\{r_i\}$. The memory footprint grows linearly with the tensor order and quadratically with the maximal TT-rank. Therefore, knowing the maximal TT-rank is usually sufficient to estimate the TT-compression benefit. However, for greater accuracy, we introduce the compression ratio measure. If $\mathbf{x} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is a tensor in TT-format, then the compression ratio is the ratio between the storage cost of $\mathbf{x}$ in TT-format and the storage cost in dense format, i.e.,

$$\frac{\sum_{i=1}^{d} r_{i-1}n_i r_i}{\prod_{j=1}^{d} n_j} \tag{2.1}$$

where r_i is the i-th TT-rank of $\mathbf{x}$. As demonstrated by the compression ratio, to significantly benefit from this formalism, the TT-ranks r_i must remain bounded and small. However, some operations among tensors in TT-format, such as algebraic addition, can increase the TT-ranks. For instance, given two TT-tensors $\mathbf{x}$ and $\mathbf{y}$ with k-th TT-rank r_k and s_k respectively, then the k-th TT-rank of $\mathbf{x}+\mathbf{y}$ is equal to $r_k + s_k$, see [23].

To address the issue of the TT-rank growth, a rounding algorithm to reduce it was proposed in [22]. The `TT-round` algorithm takes a TT-vector $\mathbf{x}$ and a relative accuracy δ as inputs and returns a TT-tensor $\tilde{x}$, that is at a relative distance δ from $\mathbf{x}$, i.e.,

$$||\mathbf{x} - \tilde{x}|| \leq \delta ||\mathbf{x}||. \tag{2.2}$$

The TT-round function is fully described in [14]. For large-scale tensors, a randomized version of the TT-round function is described in [24, 25].

To evaluate the benefit of the `TT-round`, we introduce the *compression gain*, which is the ratio of the compression ratios, written as

$$\frac{\sum_{i=1}^{d} r_{i-1} n_i r_i}{\sum_{j=1}^{d} s_{j-1} n_j s_j} \tag{2.3}$$

where r_i and s_i are the i-th TT-rank of $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $\tilde{\mathbf{x}} = \texttt{TT-round}(\mathbf{x}, \delta)$, respectively. The computational cost of a `TT-round` over $\mathbf{x}$ in terms of floating point operations, is $\mathcal{O}(dnr^3)$, where $r = \max_{i \in \{1,\ldots,d\}}\{r_i\}$ and $n = \max_{i \in \{1,\ldots,d\}}\{n_i\}$, as stated in [22].

3 Orthogonalization schemes

In the following sections, we describe the classical orthogonalization schemes and we propose their extensions to the TT-format. The input of all the orthogonalization algorithms in TT-format is $\mathcal{A}$ a set of TT-vectors and an accuracy $\delta \in \mathbb{R}_+$ for the TT-round function.

In addition, we discuss the theoretical results for the loss of orthogonality in the classical matrix computation.

3.1 Classical and Modified Gram-Schmidt

The Gram-Schmidt process [3, 4] is a tool used in theoretical linear algebra to generate an orthonormal basis from a given set of vectors. Let $\mathcal{A} = \{a_1, \ldots, a_m\}$ be a set of m linearly independent vectors of $\mathbb{R}^n$, then the key idea of the Gram-Schmidt process is to incrementally construct an orthonormal basis of the space spanned by the elements of $\mathcal{A}$. At the i-th step, the i-th element a_i is made orthogonal to the previously computed $(i-1)$ orthonormal vectors $\{q_1, \ldots, q_{i-1}\}$, by subtracting from a_i its projection along q_j. The projection is given by the inner product of a_i and q_j for $j \in \{1, \ldots, i-1\}$. After normalization the new vector is q_i, the i-th vector of the final orthonormal basis. This mechanism is easily transported into the tensor framework. Therefore, instead of presenting the theory of the Gram-Schmidt procedure in the tensor notation, we illustrate the two different realizations of this theoretical tool only in the TT-format. We carefully emphasize the differences to the classical matrix implementations.

3.1.1 Classical schemes without re-orthogonalization

The Gram-Schmidt process can be directly implemented through *Classical Gram-Schmidt* (CGS), with its TT-version outlined in Algorithm 3.1. TT-CGS initializes $\mathbf{p}_i$ to $\mathbf{a}_i$ for every $i \in \{1, \ldots, m\}$, see Line 2. In the core loop, the algorithm subtracts from $\mathbf{p}_i$ the projection of $\mathbf{a}_i$ along the $(i-1)$ previously computed tensors $\mathbf{q}_j$ of the new orthogonal basis, as described in lines 4 and 5. Finally, $\mathbf{p}_i$ is normalized and added to the new orthonormal basis $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_i\}$. The i-th column of R is defined using the projections of $\mathbf{a}_i$ along $\mathbf{q}_j$ for $j \in \{1, \ldots, i-1\}$. By construction, R is consequently upper triangular. The norm of $\mathbf{p}_i$ computed in Line 8 is the i-th diagonal entry of R. These steps are present in both the tensor and matrix versions of the Classical Gram-Schmidt algorithm. However when dealing with compressed format tensors, it is important to ensure that the algorithm steps do not significantly reduce the compression quality. Therefore, it is crucial that the TT-ranks stay small. For example, after $(k-1)$ repetitions of Line 5, which involves $(k-1)$ subtractions, the TT-rank of $\mathbf{p}$ will be bounded by kr, if r is the maximum TT-rank of $\mathbf{p}_k$ and $\mathbf{q}_j$ for every $j \in \{1, \ldots, k-1\}$. To limit the growth of TT-rank, we compress $\mathbf{p}_i$ in Line 7 using the `TT-round` algorithm with accuracy δ. This is the most computationally expensive operation in orthogonalization algorithms. As a result, the complexity of TT-CGS depends on the number of `TT-round` calls and its complexity. This last is known to be $\mathcal{O}(dnr^3)$ where d is the order of the TT-vector rounded, n and r are the maximum of the mode size and of the TT-rank respectively. However, in TT-CGS and in the other studied orthogonalization methods, the TT-rank is not always known. Linear combinations of TT-vectors obtained from the `TT-round` algorithm with accuracy δ are rounded, resulting in a TT-rank that is not known a priori. Therefore, we estimate the complexity of the orthogonalization algorithms here and after in terms of the number of rounding operations. The complexity of TT-CGS is equal to m `TT-round` operations.

Algorithm 3.1 $\mathcal{Q}, R$ = TT-CGS($\mathcal{A}, \delta$)

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix

1: **for** $i = 1, \ldots, m$ **do**
2: $\quad \mathbf{p} = \mathbf{a}_i$
3: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
4: $\quad\quad R(i,j) = \langle \mathbf{a}_i, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
5: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
6: $\quad$ **end for**
7: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
8: $\quad R(i,i) = ||\mathbf{p}||$
9: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
10: **end for**

Algorithm 3.2 $\mathcal{Q}, R$ = TT-MGS($\mathcal{A}, \delta$)

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix

1: **for** $i = 1, \ldots, m$ **do**
2: $\quad \mathbf{p} = \mathbf{a}_i$
3: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}$ along $\mathbf{q}_j$
4: $\quad\quad R(i,j) = \langle \mathbf{p}, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{p}$ along $\mathbf{q}_j$
5: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
6: $\quad$ **end for**
7: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
8: $\quad R(i,i) = ||\mathbf{p}||$
9: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
10: **end for**

In the classical matrix framework, Classical Gram-Schmidt is known to suffer from a loss of orthogonality in the computed basis, as discussed later on, cf. [11]. The *Modified Gram-Schmidt* (MGS) algorithm introduces a small algorithmic change to Classical Gram-Schmidt, which guarantees better numerical orthogonality. In TT-MGS, we remove the projection of $\mathbf{p}_i$, rather than of $\mathbf{a}_i$, along $\mathbf{q}_j$ for every $j \in \{1, \ldots, i-1\}$. This can be seen by comparing Line 4 of Algorithm 3.1 and 3.2 respectively. This modification reduces error propagation, improving the algorithm's general stability in both the classical matrix and tensor cases, as we discussed in Section 3.4. The remaining steps of the two algorithms are identical.

Under the same assumptions stated previously to estimate the complexity, we conclude that the TT-MGS computational complexity in TT-format is equal to that of TT-CGS one, given by m `TT-round` calls.

3.1.2 Classical schemes with re-orthogonalization

CGS and MGS are known to have stability issues, as described in detail in Section 3.4. The closer the input vectors are to linear dependence, the more the algorithms propagate the rounding errors, spoiling the final orthogonality of the new basis. As reported in [12], several articles, such as [5, 6, 26], have addressed this issue by introducing re-orthogonalization steps. This involves repeatedly orthogonalizing the basis using the same approach. In [12], the authors showed theoretically that one re-orthogonalization step is sufficient to significantly improve the orthogonality of the new basis generated by CGS and MGS. We briefly introduce the concepts of CGS and MGS with re-orthogonalization, referred to as TT-CGS2 and TT-MGS2, respectively, in the tensor case. We emphasize the steps that are unique to the TT-version.

Algorithm 3.3 Q, R = TT-CGS2($\mathcal{A}$, δ)

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix

1: **for** $i = 1, \ldots, m$ **do**
2: $\quad \mathbf{p}_0 = \mathbf{a}_i$
▷ repeat twice the orthogonalization loop
3: $\quad$ **for** $k = 1, 2$ **do**
4: $\quad\quad \mathbf{p}_k = \mathbf{p}_{k-1}$
5: $\quad\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
6: $\quad\quad\quad R_k(i, j) = \langle \mathbf{p}_{k-1}, \mathbf{q}_j \rangle$
▷ subtract the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
7: $\quad\quad\quad \mathbf{p}_k = \mathbf{p}_k - R_k(i, j)\mathbf{q}_j$
8: $\quad\quad$ **end for**
9: $\quad\quad \mathbf{p}_k =$ `TT-round`$(\mathbf{p}_k, \delta)$
10: $\quad$ **end for**
11: $\quad R_2(i, i) = ||\mathbf{p}_2||$
12: $\quad \mathbf{q}_i = 1/R_2(i, i)\mathbf{p}_2$ ▷ normalize $\mathbf{p}_2$
13: **end for**
▷ compute the R factor from the repeated orthogonalization loop
14: $R = R_1 + R_2$

Algorithm 3.4 Q, R = TT-MGS2($\mathcal{A}$, δ)

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix

1: **for** $i = 1, \ldots, m$ **do**
2: $\quad \mathbf{p}_0 = \mathbf{a}_i$
▷ repeat twice the orthogonalization loop
3: $\quad$ **for** $k = 1, 2$ **do**
4: $\quad\quad \mathbf{p}_k = \mathbf{p}_{k-1}$
5: $\quad\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
6: $\quad\quad\quad R_k(i, j) = \langle \mathbf{p}_k, \mathbf{q}_j \rangle$
▷ subtract the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
7: $\quad\quad\quad \mathbf{p}_k = \mathbf{p}_k - R_k(i, j)\mathbf{q}_j$
8: $\quad\quad$ **end for**
9: $\quad\quad \mathbf{p}_k =$ `TT-round`$(\mathbf{p}_k, \delta)$
10: $\quad$ **end for**
11: $\quad R_2(i, i) = ||\mathbf{p}_2||$
12: $\quad \mathbf{q}_i = 1/R_2(i, i)\mathbf{p}_2$ ▷ normalize $\mathbf{p}_2$
13: **end for**
▷ compute the R factor from the repeated orthogonalization loop
14: $R = R_1 + R_2$

In CGS2, described in Algorithm 3.3, the input TT-vector $\mathbf{a}_i$ is orthogonalized with respect to the previously computed orthogonal TT-vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$, by subtracting from $\mathbf{a}_i$ its projection along $\mathbf{q}_j$. The projection of $\mathbf{a}_i$ along $\mathbf{q}_j$ defines the (j, i) element of the first matrix R_1. These first $(i-1)$ iterations, given in Line 5 of Algorithm 3.3, define the TT-vector $\mathbf{p}_1$. Then, in Line 9, $\mathbf{p}_1$ is rounded. Up to this point, TT-CGS2 functions identically to TT-CGS. However, in TT-CGS2, the TT-vector $\mathbf{p}_1$ is orthogonalized again against $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$, after rounding. This results in $\mathbf{p}_2$. The projections along $\mathbf{q}_j$ of $\mathbf{p}_1$ determine the (j, i) component of the second matrix R_2. Once $\mathbf{p}_2$ is fully defined, it is rounded and normalized, defining the i-th orthogonal TT-vector $\mathbf{q}_i$, as stated in lines 11 and 12 of Algorithm 3.3. The $\mathbf{p}_2$ norm at the i-th iteration determines the (i, i)-th diagonal component of R_2. The R-factor from the QR decomposition computed by CGS2 is obtained by adding R_1 and R_2. The distinction between MGS2 and CGS2 is evident in Line 6 of Algorithm 3.4 and 3.3, respectively. In the first orthogonalization loop, that is, when $k = 1$ in Line 3 of both methods, the classical Gram-Schmidt version projects $\mathbf{a}_i$ along $\mathbf{q}_j$ defining $\mathbf{p}_1$, while the modified Gram-Schmidt version projects $\mathbf{p}_1$ along $\mathbf{q}_j$ to update it. In the second orthogonalization loop, i.e., when $k = 2$ in Line 3 of both algorithms, TT-CGS2 removes the projection of $\mathbf{p}_1$ along $\mathbf{q}_j$ to define $\mathbf{p}_2$. In the TT-MGS2 version, $\mathbf{p}_2$ is the TT-vector projected along $\mathbf{q}_j$ and updated. The remaining steps, including the `TT-round` and the construction of R_1, R_2 and their sum R, are identical between TT-CGS2 and TT-MGS2. It is important to note that the rounding steps are only performed in the TT-version of MGS2 and CGS2.

The computational complexity of TT-CGS2 and TT-MGS2 is estimated as $2m$ `TT-round` operations, based on the previously used hypothesis. During the m iterations, the two temporary TT-vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are rounded.

3.2 Cholesky-QR

In their article [8], the authors propose an algorithm for generating an orthogonal basis, based on the Cholesky factorization of the Gram matrix associated with the input set of linearly independent vectors. This algorithm is typically referred to as the Cholesky-QR algorithm. We briefly describe the main ideas of this orthogonalization scheme in the classical matrix framework and we describe in detail the implementation of the Cholesky-QR in the TT-format, TT-CholQR. In fact, the tensor realization of this procedure is extremely close to that of the matrix one, so a description of the tensor case and its differences from the classical matrix one is sufficient to ensure a good understanding.

Given a set of vectors $\mathcal{A} = \{a_1, \ldots, a_m\}$ with $a_i \in \mathbb{R}^n$, the Gram matrix $G \in \mathbb{R}^{m \times m}$ is defined by the inner product $G(i, j) = \langle a_i, a_j \rangle$ for every $i, j \in \{1, \ldots, m\}$. Equivalently, let a_i be the i-th column of the matrix $A \in \mathbb{R}^{n \times m}$, then in the matrix computation the Gram matrix is written as

$$G = A^\top A. \tag{3.1}$$

If the elements of $\mathcal{A}$ are linearly independent, then G is symmetric positive definite. As a consequence, its Cholesky factorization exists and is written as $G = LL^\top$, where $L \in \mathbb{R}^{m\times m}$ is a lower triangular matrix. If we now denote the transpose of L by R, then the Gram matrix gets

$$G = R^\top R. \tag{3.2}$$

Comparing Equations (3.1) and (3.2), we conclude that R is the R-factor from the QR decomposition of A, i.e., it expresses the same information of A in a different basis. The matrix Q from the QR decomposition of A is written as $Q = AR^{-1}$ where $R = L^\top$. The columns of Q form an orthogonal basis $Q = \{q_1, \ldots, q_m\}$, whose j-th element is strictly speaking a linear combination of the first j elements of $\mathcal{A}$, i.e.,

$$q_j = \sum_{k=1}^{j} R^{-1}(k, j)a_k.$$

Remark 3.1 Note that, by construction, the condition number of G is the square of that of A. Consequently, if the condition number of A associated with the set of input vectors $\mathcal{A}$ is greater than the inverse of the square root of the working precision of the arithmetic considered , e.g., $u_{64} \approx 10^{-16}$ for 64-bit computation, the associated Gram matrix G is numerically singular and its Cholesky decomposition is no longer defined. This is the main practical drawback of the method.

This procedure generates an orthonormal basis starting from a set of linear independent vectors, which is naturally extended to TT-vectors. As described in Algorithm 3.5, given a set of TT-vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ with $\mathbf{a}_i \in \mathbb{R}^{n_1\times\cdots\times n_d}$, we construct the Gram matrix $G \in \mathbb{R}^{m\times m}$ by the tensor inner product and we compute its Cholesky factorization to obtain the lower triangular matrix $L \in \mathbb{R}^{m\times m}$. As in the matrix case, $R \in \mathbb{R}^{m\times m}$ the transpose of L expresses the same information as the TT-vectors of $\mathcal{A}$, but with respect to a different basis. Following the matrix approach, we retrieve this basis, i.e., the orthonormal set Q whose element $\mathbf{q}_i \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is defined as

$$\mathbf{q}_i = \sum_{k=1}^{i} R^{-1}(k, i)\mathbf{a}_k.$$

Remark 3.2 In the matrix framework, the orthogonal vector q_j is obtained from the elements of $\mathcal{A}$ by back-substitution using the matrix R. However, this approach does not easily translated to the tensor framework, where the inverse of R must be explicitly computed.

As for the other orthogonalization techniques, we avoid memory problems by monitoring the TT-ranks and eventually rounding. Indeed, assuming that all the TT-vectors of $\mathcal{A}$ have TT-ranks bounded by r, then the i-th TT-vector constructed in Line 11 has a maximum TT-rank bounded by ir. Since this value grows linearly with m, in Line 13 we introduce a rounding step with prescribed accuracy δ.

Algorithm 3.5 $Q, R = \text{TT-CholQR}(\mathcal{A}, \delta)$

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix

for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, i$ **do**
  ▷ construct the Gram matrix through the inner product of the input TT-vectors
    $G(i, j) = G(j, i) = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$
  end for
end for
$L = \texttt{cholesky}(G)$ ▷ compute the Cholesky factorization
$R = L^\top$ and $R^{-1} = \texttt{invert}(R)$ ▷ define the R factor of the QR-factorization
for $i = 1, \ldots, m$ **do**
  $\mathbf{p} = R^{-1}(i, 1)\mathbf{a}_1$
  for $j = 2, \ldots, i$ **do**
  ▷ construct the i-th new basis TT-vector as a linear combination of the $(i-1)$ input TT-vector
    $\mathbf{p} = \mathbf{p} + R^{-1}(i, j)\mathbf{a}_j$
  end for
  $\mathbf{q}_i = \texttt{TT-round}(\mathbf{p}, \delta)$ ▷ round the TT-vector before adding it to the basis
end for

As in Sections 3.1 and 3.2, note that the complexity of the TT-CholQR algorithm is given by m `TT-round` operations. However, in this particular case, we can even estimate the cost of each single rounding step, and thus of the entire algorithm. Indeed, the maximum TT-rank of the rounded TT-vector $\mathbf{q}_i$ is bounded by $i\,r$, under the assumption that the maximum TT-rank and the maximum mode size of $\mathbf{a}_i$ are bounded by $r \in \mathbb{N}$ and $n \in \mathbb{N}$ respectively. Consequently, the computational cost, i.e., the number of floating point operations, of each rounding operation, the most expensive step in the entire algorithm, is known and is equal to $\mathcal{O}(dni^3r^3)$; summing over $i \in \{1, \ldots, m\}$, we conclude that the cost of the TT-CholQR algorithm is $\mathcal{O}(dnm^4r)$ floating point operations.

3.3 Householder reflections

In the classical matrix framework, Householder transformations are commonly used to generate an orthogonal basis or to compute the QR-factorization due to their stability properties. Given a set of m vectors, $a_k \in \mathbb{R}^n$, at each step k, the Householder orthogonalization applies the first $k-1$ Householder reflectors to the input vectors, i.e., $a_j^{(k-1)} = H_{k-1} \cdots H_1 a_j$, for $j \in \{k, \ldots, m\}$. The vector $a_k^{(k-1)}$ is used to compute the next reflector, H_k, that moves $a_k^{(k-1)}$ on a linear combination of the first k elements of the canonical basis. The Householder vector defining H_k is $u_k = a_k^{(k-1)} - \sum_{j=1}^{k} R(j, k)e_j$ where

$$R(j, k) = \langle a_k^{(k-1)}, e_j \rangle \qquad \text{and} \qquad R(k, k) = \pm \sqrt{\left\| a_k^{(k-1)} \right\|^2 - \sum_{j=1}^{k-1} R(j, k)^2}. \tag{3.3}$$

Once m Householder reflectors are computed, the k-th element of the orthogonal basis is generated applying in reverse order the first k Householder reflectors to the k-th vector of the canonical basis of $\mathbb{R}^n$.

Remark 3.3 In practice, H_k is constructed using only the last $(n-k+1)$ entries of $a_k^{(k-1)}$, moving it along $e_1 \in \mathbb{R}^{n-k+1}$.This reduced approach cannot be replicated in the tensor framework, because objects are expressed in compressed format.

Householder TT-vector. Let $\mathcal{A}$ be a set of m TT-vectors, $\mathbf{a}_i \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, and $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ the canonical basis of a subspace of dimension m in $\mathbb{R}^{n_1 \times \cdots \times n_d}$. The i-th element of the canonical basis is $\mathbf{e}_i = e_{i_1} \otimes \cdots \otimes e_{i_d}$ where e_{i_k} is the i_k-th canonical basis vector of $\mathbb{R}^{n_k}$ and

$$i = i_1 + \sum_{\alpha=2}^{d} (i_\alpha - 1) m_\alpha \qquad \text{with} \qquad m_\alpha = \prod_{\beta=1}^{\alpha-1} n_\beta. \tag{3.4}$$

Following Equation 3.3, we define the k-th Householder TT-vector, $\mathbf{u}_k \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, as

$$\mathbf{u}_k = \mathbf{a}_k^{(k-1)} - \sum_{j=1}^{k} R(j,k)\mathbf{e}_j$$

where $R(j,k) = \langle \mathbf{a}_k^{(k-1)}, \mathbf{e}_j \rangle$ and $R(k,k) = \pm\sqrt{||\mathbf{a}_k^{(k-1)}||^2 - \sum_{\ell=1}^{k-1} R(\ell,k)^2}$. The (k,k)-th component of R takes a positive sign if $\langle \mathbf{a}_k^{(k-1)}, \mathbf{e}_k \rangle > 0$; otherwise, it takes a negative sign. This choice extends the stability-preserving idea given in [27] to the tensor framework. The construction of a Householder TT-vector is summarized in Algorithm 3.6.

Algorithm 3.6 $\mathbf{u}, r = \texttt{TTH-vec}(\mathbf{a}, \mathcal{E}_k, \delta)$

input: $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ a TT-vector, $\mathcal{E}_k = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ a subset of the canonical tensor space basis in TT-format, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $\mathbf{u}$ the Householder TT-vector, r a column of the R-factor

$\mathbf{w} = \mathbf{a}$
for $j = 1, \ldots, k-1$ **do**
▷ compute the component of $\mathbf{a}$ along the j-th canonical basis TT-vector
  $r(j) = \langle \mathbf{a}, \mathbf{e}_j \rangle$
▷ set to zero the component of $\mathbf{a}$ along the j-th canonical basis TT-vector $\mathbf{e}_j$
  $\mathbf{w} = \mathbf{w} - r(j)\mathbf{e}_j$
end for
$\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
▷ subtract from the norm of $\mathbf{a}$ the contribution of the components set to zero
$r(k) = \text{sign}(\langle \mathbf{a}, \mathbf{e}_k \rangle)\sqrt{||\mathbf{a}||^2 - \|r\|^2}$
$\mathbf{w} = \mathbf{w} - r(k)\mathbf{f}_i$
$\mathbf{z} = \texttt{TT-round}(\mathbf{w}, \delta)$
$\mathbf{u} = (1/||\mathbf{z}||)\mathbf{w}$

We want to ensure that the TT-rank of $\mathbf{u}_k$ remains sufficiently small because it plays a crucial role in the Householder transformation, and its TT-rank has a significant

impact on the complexity entire process. Thus, we introduce two rounding steps. The first appears in Line 6, since after removing the first $(k-1)$ components of $\mathbf{a}_k^{(k-1)}$, the TT-rank of $\mathbf{u}_k$ is bounded by $(r_a + k - 1)$, assuming r_a as an upper bound on the TT-rank of $\mathbf{a}_k^{(k-1)}$. The second is performed after subtracting the $R(k,k)$ in Line 9 to minimize as much as possible the TT-rank of $\mathbf{u}_k$.

As in the matrix case, the Householder reflection, $\mathbf{H}_k$, is applied to a TT-vector, $\mathbf{x}$, using the Householder TT-vector that defines it, $\mathbf{u}_k$, computing

$$\mathbf{H}_k(\mathbf{x}) = \mathbf{x} - 2\langle \mathbf{x},\ \mathbf{u}_k\rangle \mathbf{u}_k.$$

Householder orthogonal basis. Once m Householder TT-vectors are computed, we construct a new set $\mathcal{Q}$ of orthonormal TT-vectors. The k-th element of $\mathcal{Q}$, $\mathbf{q}_k$, is obtained by applying the first k Householder reflections in reverse order to $\mathbf{e}_i$ from the canonical basis $\mathcal{E}$. To maintain the maximum TT-rank of $\mathbf{q}_i$ limited and prevent memory overflow, each $\mathbf{q}_i$ is rounded to an accuracy δ.

The complete TT-Householder algorithm is reported in Algorithm 3.7. Note that in Line 8 we perform `TT-round` of $\mathbf{a}_k^{(k-1)}$ before generating the associated Householder TT-vector. This rounding tackles the potential growth of the TT-rank of $\mathbf{a}_k$, due to the application of the previous $(k-1)$ Householder reflections. The TT-Householder algorithm requires $4m$ `TT-round` operations, including two for each Householder TT-vector, one for each TT-vector $\mathbf{a}_k^{(k-1)}$ after the $(k-1)$ reflections, and one for each $\mathbf{q}_k$ orthogonal TT-vector. As a result, the TT-Householder algorithm is computationally more expensive than all the other orthogonalization methods. Specifically, it is 4 times more expensive than CGS and MGS, and twice as expensive as CGS2 and MGS2.

Algorithm 3.7 $\mathcal{Q}, R$ = TT-Householder($\mathcal{A}, \delta$)

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
output: $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, R the upper triangular matrix
1: let $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ be the canonical basis of a subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$
2: set $\mathbf{a}_j^{(0)} = \mathbf{a}_j$ for $j \in \{1, \ldots, d\}$
3: **for** $k = 1, \ldots, m$ **do**
▷ construct the k-th Householder TT-vector
4: $\mathbf{u}_k, R(:k,k) = \texttt{TTH-vec}(\mathbf{a}_k^{(k-1)}, \mathcal{E}_k, \delta)$ with $\mathcal{E}_k = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$
5: **for** $j = k, \ldots, m$ **do**
6: $\mathbf{a}_j^{(k)} = \mathbf{a}_j^{(k-1)} - 2\langle \mathbf{a}_j^{(k-1)},\ \mathbf{u}_k\rangle \mathbf{u}_k$ ▷ update the j-th element of $\mathcal{A}$
7: **end for**
▷ round the TT-vector that will define the successive Householder reflection
8: $\mathbf{a}_{k+1}^{(k)} = \texttt{TT-round}(\mathbf{a}_{k+1}^{(k)}, \delta)$
9: **end for**
10: **for** $i = 1, \ldots, m$ **do** ▷ compute the new orthogonal basis
11: $\mathbf{q}_i = \mathbf{e}_i$
12: **for** $j = i, \ldots, 1$ **do**
13: $\mathbf{q}_i = \mathbf{q}_i - 2\langle \mathbf{q}_i,\ \mathbf{u}_j\rangle \mathbf{u}_j$ for ▷ reflect the i-th element of $\mathcal{E}$
14: **end for**
15: $\mathbf{q}_i = \texttt{TT-round}(\mathbf{q}_i, \delta)$
16: **end for**

3.4 Stability comparison

A central issue for the orthogonalization algorithms is the loss of orthogonality, i.e., how much the rounding errors propagate and affect the orthogonality of the computed basis. The loss of orthogonality of an orthogonalization scheme applied to the set $\mathcal{A}_m = \{a_1, \ldots, a_m\}$ is defined by the L2-norm of the difference between the identity matrix of size m and the Gram matrix defined by the m vectors generated by the orthogonalization algorithm. We give the definition more formally. Let $\mathcal{Q}_m = \{q_1, \ldots, q_m\}$ be a set of m vectors, obtained from an orthogonalization scheme applied to the m vectors of the input set $\mathcal{A}_m$. Let $q_i \in \mathcal{Q}_m$ be the i-th column of the matrix $Q_m \in \mathbb{R}^{n\times m}$ for every $i \in \{1, \ldots, m\}$, then the Gram matrix associated with the set $\mathcal{Q}_m$ is $Q_m^\top Q_m$. Note that the (i, j) element of $Q_m^\top Q_m$ is the inner product of q_i and q_j, i.e., $Q_m^\top Q_m(i, j) = \langle q_i, q_j \rangle$ for every $i, j \in \{1, \ldots, m\}$. Then, the loss of orthogonality of the considered algorithm for a basis of size m is equal to

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \tag{3.5}$$

where the norm considered is the L2 norm for matrices. In the classical matrix framework, an orthogonalization scheme is said to be *numerically stable* if the loss of orthogonality of the basis it computes is of the order of the unit round-off u of the working arithmetic. The following theoretical results, which hold for the six orthogonalization schemes in classical linear algebra, provide a base line for the comparison with the numerical results obtained in the tensor framework, discussed in Section 4.

In [12, Theorem 1], the authors prove that the loss of orthogonality for a basis obtained by CGS, given $\mathcal{A}_m = \{a_1, \ldots, a_m\}$ a set of m vectors, is bounded by a positive constant times the unit round-off u of the working arithmetic, times the squared condition number of the matrix $A_m \in \mathbb{R}^{n\times m}$ whose j-th column is $a_j \in \mathbb{R}^n$ for $j \in \{1, \ldots, m\}$, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\big(u\kappa^2(A_m)\big) \tag{3.6}$$

as long as $\kappa^2(A_m)u \ll 1$. In [11], an upper bound for the loss of orthogonality of MGS is provided. The loss of orthogonality for a basis of m vectors produced by MGS from $\{a_1, \ldots, a_m\}$ is upper bounded by a constant times the unit round-off u times the condition number of A_m as previously defined from $\mathcal{A}_m$ elements, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\big(u\kappa(A_m)\big) \tag{3.7}$$

as long as $\kappa(A_m)u \ll 1$. The authors of [8, Theorem 4.1], who proposed the Cholesky-QR orthogonalization scheme, also estimated an upper bound for the loss of orthogonality of their orthogonalization technique. The loss of orthogonality of a basis of m vectors produced by the Cholesky-QR scheme from $\{a_1, \ldots, a_m\}$ satisfies the same upper bound as CGS, given in (3.6). The Householder orthogonalization algorithm is known for its stability. The loss of orthogonality of a basis of m vectors produced by Householder transformations from $\{a_1, \ldots, a_m\}$ is bounded by a constant

Table 1 Computational costs in floating point operations and in `TT-round` operations, and bounds for the loss of orthogonality, theoretical with respect to the unit round-off u and conjectured ones with respect to the rounding accuracy δ, for an input set of m vectors and TT-vectors respectively.

	Matrix		TT-vectors	
Algorithm	*Computational cost in fp operations*	$\left\|\mathbb{I}_m - Q_m^\top Q_m\right\|_2$	*Computational cost in* `TT-round`	$\left\|\mathbb{I}_m - Q_m^\top Q_m\right\|_2$
CholQR	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_m))$	m	$\mathcal{O}(\delta\kappa^2(A_m))$
CGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_m))$	m	$\mathcal{O}(\delta\kappa^2(A_m))$
MGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa(A_m))$	m	$\mathcal{O}(\delta\kappa(A_m))$
CGS2	$\mathcal{O}(4nm^2)$	$\mathcal{O}(u)$	$2m$	$\mathcal{O}(\delta)$
MGS2	$\mathcal{O}(4nm^2)$	$\mathcal{O}(u)$	$2m$	$\mathcal{O}(\delta)$
Householder	$\mathcal{O}(2nm^2-2m^3/3)$	$\mathcal{O}(u)$	$4m$	$\mathcal{O}(\delta)$

times the round-off unit, i.e.,

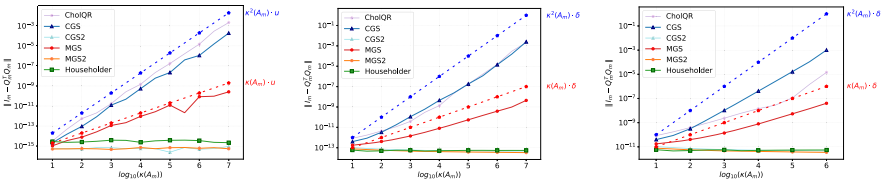
$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}(u) \tag{3.8}$$

as proven in [28]. When introducing a further orthogonalization step in the classical and modified Gram-Schmidt, defining CGS2 and MGS2, their loss of orthogonality improves considerably, reaching Householder quality. As proven in [10, 12], the loss of orthogonality of CGS2 and MGS2 satisfies the bound given in Equation (3.8), under the hypothesis $\kappa^2(A_m)u \ll 1$ for CGS2, while it holds for MGS2 if $\kappa(A_m)u \ll 1$.

Table 1 presents a summary of all the loss of orthogonality bounds. The left columns present the theoretical bounds established in the matrix computation framework, along with their associated computational complexity measured in floating-point operations. On the right, we provide analogous results for TT computations: the computational complexity is expressed in terms of the number of TT-rounds, and the bounds on the loss of orthogonality are conjectural. These conjectures will be further investigated through experiments in the remainder of this paper.

In Figure 2, we display, in the matrix framework, the loss of orthogonality of the computed basis when the condition number of the initial vectors is increased. The left most graph in Figure 2 corresponds to IEEE fp64 computation and illustrates the theoretical results presented in the first column of Table 1. To motivate the bounds proposed for TT computations, we also perform experiments where fake normwise perturbations are introduced during the computation. In these experiments, any time we compute or update a vector x, we perturb it and substitute it in the subsequent steps of the orthogonalization algorithm with its perturbed version $\tilde{x}$. The perturbed vector $\tilde{x}$ is designed to satisfy a condition analogous to Equation (2.2), which characterizes TT-round perturbation effects.

As conjectured, the loss of orthogonality is $\mathcal{O}(\delta)$ and remains independent of the condition number of the vector set for the Householder, CGS2, and MGS2 algorithms. In contrast, for MGS, the loss of orthogonality grows linearly with the condition number, while for Cholesky-QR and CGS, the increase is quadratic.



(a) $u = 1.11 \cdot 10^{-16}$ (b) $\delta = 10^{-14}$ (c) $\delta = 10^{-12}$

Loss of orthogonality with classical algorithms

Fig. 2 Loss of orthogonality in matrix computation with normwise perturbation of various magnitude.

4 Numerical tensor experiments

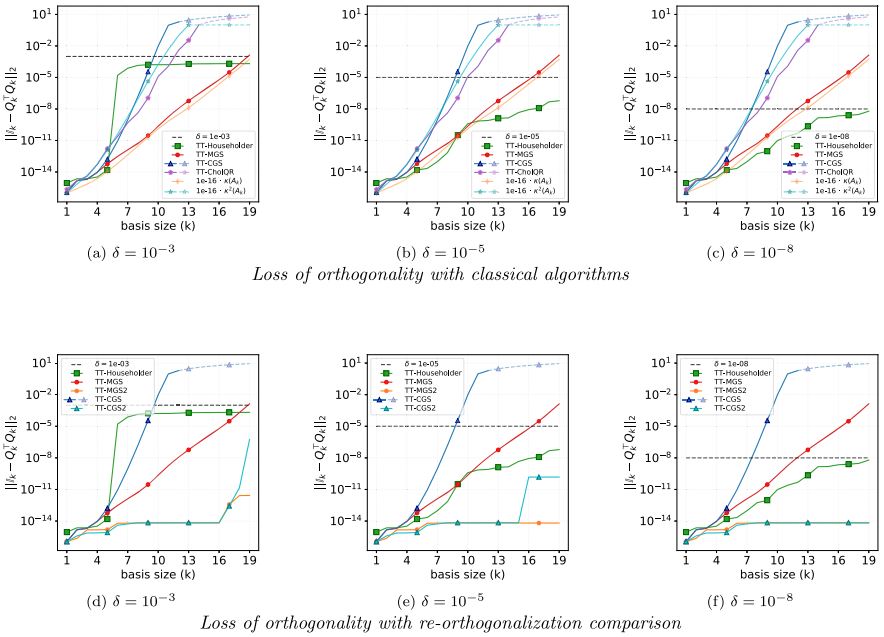
Sections 3.1 - 3.3 describe four orthogonalization methods that produce an orthonormal basis of TT-vectors, given a set of TT-vectors and a rounding accuracy δ. This section analyzes two sets of results obtained from the orthogonalization schemes, highlighting similarities and differences with the known theoretical results in matrix computation. In all the experiments, the input set of TT-vectors $\mathcal{A}_m = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ is generated using a Krylov process. Starting with a TT-vector of ones, $\mathbf{x}_1 \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, we iteratively compute $\mathbf{x}_{j+1} = -\mathbf{\Delta}_d \mathbf{a}_j$ where $\mathbf{\Delta}_d$ is the TT-matrix representing the discretization of the Laplacian operator of order d with Dirichlet boundary conditions, see [29], and $\mathbf{a}_j$ is the normalized output of the `TT-round` algorithm applied to $\mathbf{x}_j$ with the accuracy replaced by a maximum TT-rank equal to 1, see [14] for further details. As a result, $\mathbf{a}_j$, the j-th element of $\mathcal{A}_m$, has a TT-rank of 1 for every $j \in \{1, \ldots, m\}$. This TT-rank constraint facilitates the analysis of the memory requirement. The elements of $\mathcal{A}$ are generated as a sequence of m normalized (and rounded) Krylov TT-vectors. Therefore, if the first k of them are vectorized and arranged as columns of the matrix A_k, the condition number $\kappa(A_k)$ grows for $k \in \{1, \ldots, m\}$. The $\mathcal{A}_k$ denotes the subset of $\mathcal{A}_m$ defined by its first k TT-vectors. The two experiments differ in the problem dimension, with the first having a dimension of 3 and the second having dimension 6, but they have the same mode size $n = 15$. Further details are provided in the following sections.

4.1 Numerical loss of orthogonality

This section examines the numerical results of two sets of experiments from the perspective of the loss of orthogonality. The purpose is to highlight the similarities with the classical matrix orthogonalization methods. Specifically, we investigate the loss of orthogonality $\left\| \mathbb{I}_k - Q_k^\top Q_k \right\|_2$, where the (i, j) element of $Q_k^\top Q_k$ is computed by the inner product of the i-th and j-th TT-vector of the orthogonal basis, i.e.,

$$Q_k^\top Q_k(i, j) = \langle \mathbf{q}_i, \mathbf{q}_j \rangle$$

for $\mathbf{q}_i, \mathbf{q}_j \in Q$ for $i, j \in \{1, \ldots, k\}$ for $k \in \{1, \ldots, m\}$, where Q denotes the set of TT-vectors produced by the considered orthogonalization algorithm.



(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms

(d) $\delta = 10^{-3}$ (e) $\delta = 10^{-5}$ (f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

Fig. 3 Loss of orthogonality and condition number for $m = 20$ TT-vectors of order $d = 3$ and mode size $n = 15$. The curves get dashed and partially transparent when they get greater than 1.

4.2 Order-3 experiments

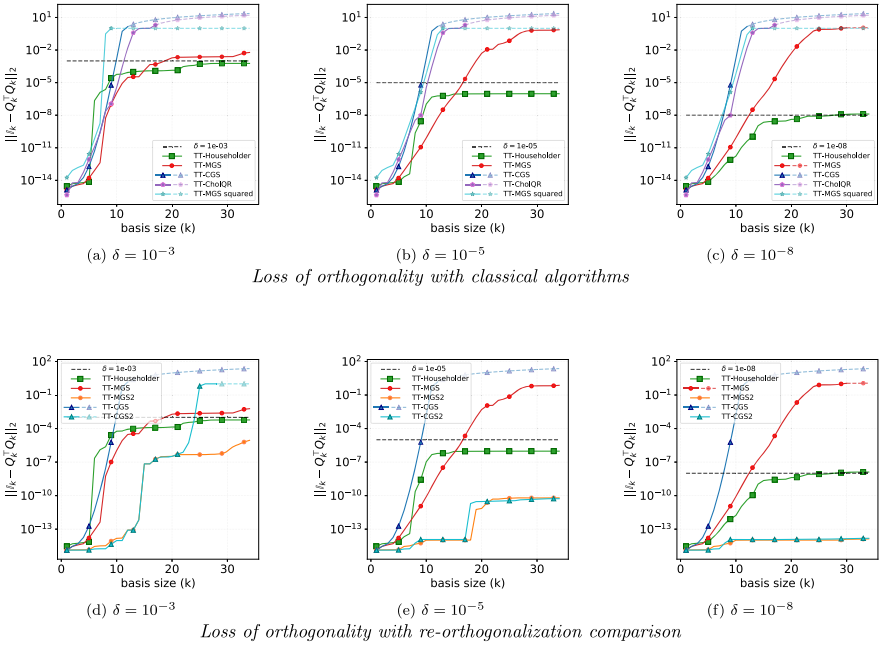
In the first experiment, we set the order $d = 3$, the size mode $n_i = 15$ for $i \in \{1, 2, 3\}$, and the input number of TT-vectors $m = 20$. Figure 3 reports the results of this first group of experiments for the six different schemes and three different rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$. The use of a low dimensional problem allows us to convert each TT-vector $\mathbf{a}_j$ into a dense format and vectorize it. These vectors are stored as the j-th column of $A_m \in \mathbb{R}^{n^3 \times m}$. Consequently, we can estimate the condition number of $A_k \in \mathbb{R}^{n^3 \times k}$, which is the submatrix of A_m formed by its first k columns. Figure 3 displays the loss of orthogonality with colored continuous curves, and the constant rounding accuracy δ with a dashed black line in all plots. The condition number $\kappa(A_k)$ and its squared value scaled by $u \approx 10^{-16}$ are also shown with colored continuous lines, as long as they are smaller than 1. For ease of comparison with the slope of the loss of orthogonality of TT-MGS and TT-CGS, the condition number curves are scaled. All the curves are dependent on the basis size k. As previously mentioned, the TT-vectors are generated as a sequence of normalized Krylov TT-vectors. As the value of k increases, the elements of $\mathcal{A}_k$ become more linearly dependent, resulting in an increase in associated condition number $\kappa(A_k)$.

To aid interpretation, we present three plots in Figure 3a, 3b and 3c, which show the loss of orthogonality of standard methods in tensor format: TT-Householder, TT-MGS, TT-CGS, TT-CholQR. Figures 3d, 3e and 3f show the loss of orthogonality of re-orthogonalization methods: TT-MGS2 and TT-CGS2, compared to TT-Householder,

TT-MGS and TT-CGS. All six plots in Figure 3 exhibit similar behaviors. The loss of orthogonality of the TT-Householder method in green stagnates around the rounding accuracy δ, as shown in Figure 3a. This is consistent with the matrix theoretical expectation, stated in Equation (3.8), where the unit round-off u is replaced by the `TT-round` accuracy δ. The loss of orthogonality of TT-MGS method in red grows with the same slope as the condition number $\kappa(A_k)$ in dashed green, matching the matrix upper bound stated in (3.7). Finally, both the TT-CGS and and TT-CholQR loss of orthogonality curves cross the rounding accuracy dashed line faster than TT-MGS. This curve follows the squared condition number $\kappa^2(A_k)$, as long as $\kappa^2(A_k) < 10^2$. The loss of orthogonality curves for TT-CGS and TT-CholQR stagnates below 10^2 for $k > 10$ approximately, while $\kappa^2(A_k)$ continues to grow. Upon analyzing the second line plots, it is evident that Figure 3d, 3e and 3f demonstrate a significant improvement in the loss of orthogonality when a second re-orthogonalization loop is introduced. It is noteworthy that the loss of orthogonality of TT-CGS2 is close to the machine precision, approximately 10^{-14} for $\delta \in \{10^{-3}, 10^{-5}\}$, increasing after for $k \geq 15$. For $\delta = 10^{-8}$, it remains around 10^{-14}. Therefore, as long as the elements of $\mathcal{A}_k$ are not highly collinear, TT-CGS2 outperforms TT-CGS, TT-MGS and TT-Householder, but not TT-MGS2. For the rounding accuracy $\delta \in \{10^{-5}, 10^{-8}\}$, the loss of orthogonality of TT-MGS2 remains around 10^{-14}, while for $\delta = 10^{-3}$ the loss of orthogonality jumps from 10^{-14} to 10^{-11}, where it appears to remain constant, when $k > 16$. Overall ,TT-MGS2 is the best performing algorithm among all the others. The results for TT-CGS2 and TT-MGS2 are consistent with the matrix theory presented in Section 3.4. It is hypothesized that the jumps occur when the condition number $\kappa(A_k)$, or its square, multiplied by the rounding accuracy is no longer sufficiently smaller than 1.

4.3 Order-6 experiments

To further validate our results and to study their applicability to large-scale problems, we introduce a second experimental framework. We set the problem order to $d = 6$ with size mode $n_i = 15$ for $i \in \{1, \ldots, 6\}$, and generate $m = 35$ TT-vectors, defining the set $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_{35}\}$. For the rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, we compute the loss of orthogonality for the four orthogonalization schemes, presented in Section 3.1–3.3. Figure 4 displays the loss of orthogonality of these experiments. Due to the problem order $d = 6$ and size $n = 15$, the curve of the condition number of the matrix $A_k \in \mathbb{R}^{n^6 \times k}$ is not included. To compensate for the absence of the condition number curve, we display the square of the TT-MGS loss of orthogonality values with a dashed line. This line should exhibit the same slope as the CGS loss of orthogonality (and consequently of the squared condition number $\kappa(A_k)$), if the matrix theory extends to the TT-framework. Similarly to the previous case, the first line of plots displays standard orthogonalization algorithms in TT-format. The second line shows results from methods with re-orthogonalization. Figures 4a, 4b and 4c demonstrate that the TT-Householder orthogonalization algorithm produces a basis with a loss of orthogonality that stagnates around the rounding accuracy δ for every value in $\{10^{-3}, 10^{-5}, 10^{-8}\}$ after the basis size exceeds approximately 10. This supports the intuition that the bound expressed in Equation (3.8) still holds true in the



(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms

(d) $\delta = 10^{-3}$ (e) $\delta = 10^{-5}$ (f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

Fig. 4 Loss of orthogonality for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$. The curves get dashed and partially transparent when they get greater than 1.

tensor framework, with the unit round-off u replaced by the `TT-round` accuracy δ. In Figures 4a and 4b, when the basis size is smaller than about 15, the TT-MGS loss of orthogonality is smaller than the Householder one. However, when the basis includes more then 15 TT-vectors, the relation reverses. For more accurate computation, specifically for $\delta = 10^{-8}$, the Householder loss of orthogonality outperforms the TT-MGS loss of orthogonality when the basis size is around 5. Note that the TT-MGS loss of orthogonality increases linearly and then stabilizes at different level for each rounding accuracy. It reaches 10^{-2} for $\delta = 10^{-3}$ and 1 for $\delta \in \{10^{-5}, 10^{-8}\}$. The TT-CGS and TT-CholQR loss of orthogonality curves reaches stabilization almost immediately when the basis size is greater than 10 for all the rounding accuracy values, as shown in Figure 4. This is likely due to the poor condition number of the input, rendering the assumptions of matrix theory invalid. Specifically, the condition number multiplied by the rounding accuracy smaller than 1. Furthermore, Figures 4b and 4c demonstrate that the loss of orthogonality of TT-CGS and TT-CholQR follows the square of loss of orthogonality of the TT-MGS. This supports the idea that even in the TT-format, the loss of orthogonality of TT-MGS increases as the condition number, while the loss of orthogonality of TT-CholQR and TT-CGS increases as the squared condition number grows. Additionally, Figures 4d, 4e and 4f display the loss of orthogonality of TT-MGS2 and TT-CGS2 compared to the previously analyzed results of TT-Householder, TT-MGS and TT-CGS. TT-MGS2 outperforms all the other methods for all considered rounding accuracies. Its loss of orthogonality stagnates around 10^{-5} for $\delta = 10^{-3}$, around 10^{-10} for $\delta = 10^{-5}$ and around 10^{-13} for $\delta = 10^{-8}$. In Figures 4d–4f,

the TT-MGS2 curve shows a larger jump for greater values of δ, for $15 \leq k \leq 20$ with $\delta \in \{10^{-3}, 10^{-5}\}$ and for $k \sim 10$ with $\delta = 10^{-8}$. TT-CGS2 outperforms the TT-Householder, TT-CGS and TT-MGS, similarly to TT-MGS2, as long as the TT-vectors are not highly collinear (i.e., for $k < 20$ when $\delta = 10^{-3}$) for $\delta \in \{10^{-5}, 10^{-8}\}$. These results support the conclusion made for the $d = 3$ experiments, that the bounds for the loss of orthogonality of TT-CGS2 and TT-MGS2 established in the classical matrix framework still hold, possibly under revised assumptions.

4.4 Memory usage estimation

This section aims to analyze the effects of the orthogonalization process on the TT-rank and memory requirement based on experimental results. The growth of the TT-ranks, of the compression ratio (defined in Equation (2.1)), and the compression gain (cf. Equation (2.3)) curve of the orthogonal basis are investigated in the second set of experiments with $d = 6$, $n_i = 15$, and $m = 35$. This setting can be considered a large-scale one, and we use as rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$.

4.4.1 Householder transformation

The Householder algorithm 3.7 applies the `TT-round` to three sets of TT-vectors: the Householder TT-vector $\mathbf{u}_k$, the TT-vector $\mathbf{a}_k$ (to which k Householder transformations are applied) and the orthogonal TT-vector $\mathbf{q}_k$ (obtained from the canonical basis TT-vectors with i successive Householder transformations). It is important to study the evolution of the maximum TT-rank, of the compression ratio, and gain for each of these three groups of TT-vector. Figure 5 displays the maximum TT-rank, the compression ratio, and the compression gain of $\mathbf{u}_k$, $\mathbf{a}_k$ (after the k-th reflection), and $\mathbf{q}_k$ for every $k \in \{1, \ldots, 35\}$ and all values of the rounding accuracy δ. The maximum TT-rank and the compression ratio of $\mathbf{u}_k$, $\mathbf{a}_k$ and $\mathbf{q}_k$ increase with increasing basis sizes, k, as expected, due to the growing number of terms in their computation. Notably, the maximum TT-rank of $\mathbf{q}_k$ exceeds that of $\mathbf{u}_k$ for a basis size greater than 10. This property is important because in most practical vector computations, only the Householder TT-vectors $\mathbf{u}_k$ are stored, and the orthogonal basis TT-vector $\mathbf{q}_k$ is usually not explicitly formed. Figures 5a,5b and 5c show that the maximum TT-rank of $\mathbf{a}_k$ after the k-th Householder reflection is extremely low, especially when compared to those of $\mathbf{q}_k$ and $\mathbf{u}_k$. Furthermore, the maximum TT-rank of $\mathbf{a}_k$ increases every time the index k is a multiple of the mode size $n = 15$, as shown in Figure 5a and 5b. This pattern is also observed in Figure 5c for $\delta = 10^{-8}$, although it is less consistent. We attempted to explore this phenomenon theoretically, but we were unable to provide a clear and convincing explanation.

The compression ratio closely follows the same trend as the maximum TT-rank, but it allows us to monitor memory growth as a percentage. Figures 5d, 5eand 5e indicate that the compression ratio of $\mathbf{q}_k$ plateaus at approximately 10^{-1}, while that of $\mathbf{u}_k$ plateaus at around 10^{-2}. In Figure 5f, it is unclear whether the compression ratio of $\mathbf{q}_k$ and $\mathbf{u}_k$ plateaus at 1 and 10^{-1} respectively. In Figures 5g, 5i and 5h, the gain curves of $\mathbf{u}_k$, $\mathbf{q}_k$ and $\mathbf{a}_k$ exhibit same behavior. Specifically, we examine the gain of

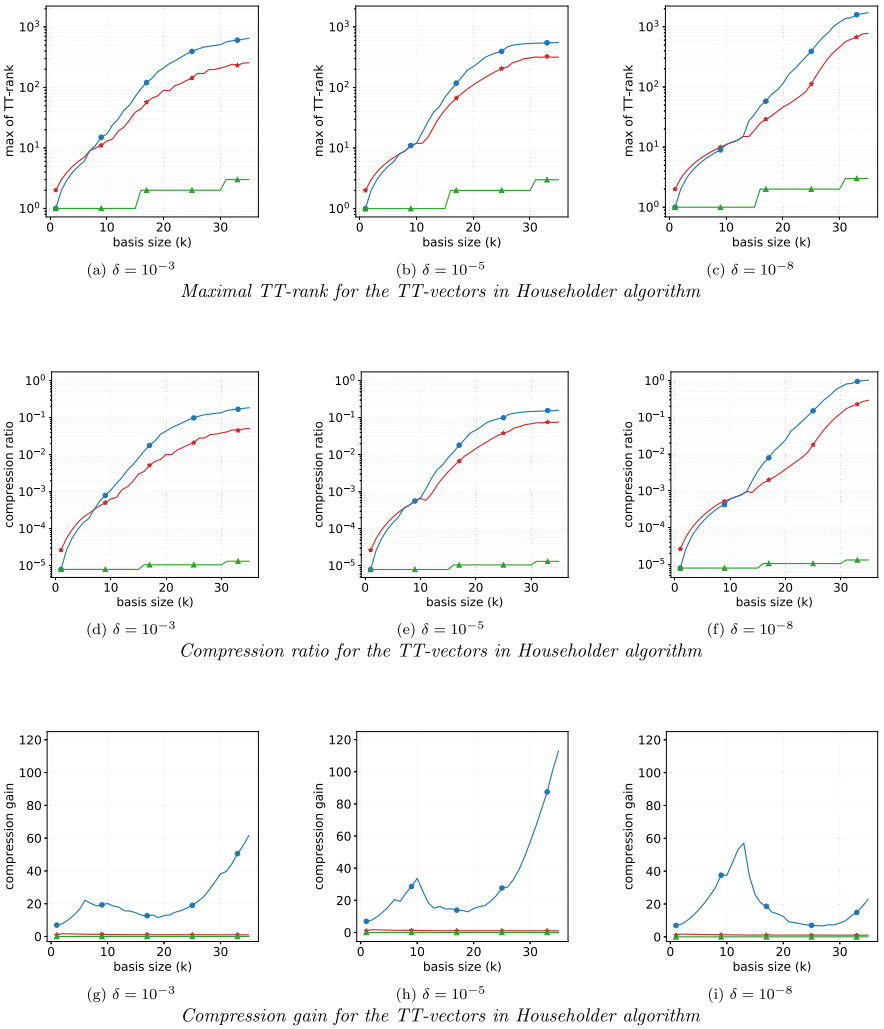


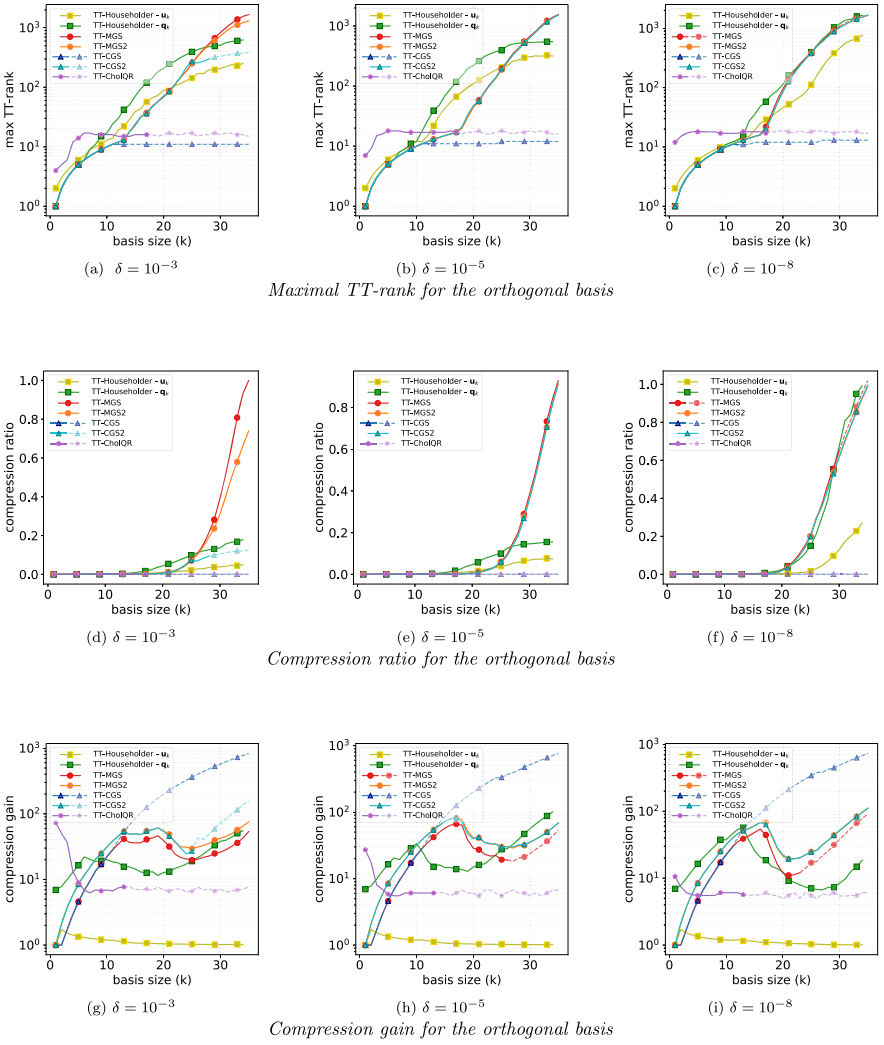
Fig. 5 Memory requirement for the Householder TT-vector $\mathbf{u}_k$ (—★—), basis TT-vector $\mathbf{q}_k$ (—●—) and the reflected TT-vector $\mathbf{a}_k$ (—▲—), assuming as input $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$

compressing the k-th Householder TT-vector $\mathbf{u}_k$ and the input TT-vector $\mathbf{a}_k$ (after the k-th reflection). These curves are almost constant and low during the iterations. The several compression steps of the associated TT-vectors during the previous iterations are likely the cause. In contrast, the compression gain for $\mathbf{q}_k$ is significantly larger. The compression gain curve of $\mathbf{q}_k$ increases during the first 10 iterations for all rounding accuracies, decreases slightly after, and seems to raise again. It is worth noting that the compression gain curve of the Householder basis reaches its highest value at around 110 when $\delta = 10^{-5}$. This indicates that after the compression, slightly less than 1% of the memory used to store the same tensor in TT-format before the compression is required.

4.4.2 Orthogonalization scheme comparison

After describing the TT-Householder algorithm's memory requirements, we compare the four orthogonalization schemes from a memory consumption perspective. To make a fair comparison, we consider both memory consumption perspective and loss of orthogonality, which is studied in Section 4.1. Figure 6 displays the maximum TT-rank, the compression ratio and the gain for the orthogonal TT-vectors $\mathbf{q}_k$ generated by the orthogonalization schemes plus the Householder TT-vectors $\mathbf{u}_k$, for different accuracies δ. It is important to note that $\mathbf{q}_k$ are not computed in many applications. The curves in the corresponding figure become dashed and partially transparent for every rounding accuracy δ when the corresponding loss of orthogonality exceeds 10^{-1}. In all Figures 6a, 6b, and 6c show that the maximum TT-rank of the orthogonal TT-vectors computed by TT-CGS and TT-CholQR schemes stagnates around 10. However, there is no clear theoretical justification for this phenomenon. The maximal TT-rank of $\mathbf{q}_k$ from the TT-CholQR algorithm is theoretically bounded by k multiplied by the maximal TT-rank of $\mathbf{a}_j$ for $j \in \{1, \ldots, k\}$. When generating $\mathbf{a}_j$, they are rounded with a maximal TT-rank equal to 1. In our experimental framework, the maximal TT-rank of $\mathbf{q}_k$ is bounded by k. Conversely, the maximal TT-rank of $\mathbf{q}_k$ from TT-CGS is bounded by $1 + k(k-1)/2$ knowing that the maximal TT-rank of $\mathbf{a}_i$ is bounded by 1 for $i \in \{1, \ldots, m\}$ in our experiments. In terms of the maximal TT-rank, TT-CholQR outperforms TT-MGS and TT-Householder for basis sizes greater than 10. However, TT-CGS sets a lower bound for the maximal TT-rank. It is important to note that the loss of orthogonality of TT-CGS and TT-CholQR becomes greater than 10^{-1} around $k = 10$. On the other hand, the TT-MGS maximal TT-rank curve becomes dashed around $k = 20$, while the Householder curve never does. This means that the loss of orthogonality arrives much later for TT-MGS or may not arrive at all for TT-Householder. In Figure 6a, the maximal TT-rank of $\mathbf{q}_k$ from TT-MGS exceeds the maximal TT-rank of the Householder TT-vector $\mathbf{u}_k$ and the Householder orthogonal TT-vector $\mathbf{q}_k$ when the basis size k reaches 20 and 25, respectively. Figure 6b shows a comparable relationship between the maximal TT-rank for Householder and for MGS generated orthogonal TT-vectors. However, the turning point occurs at different basis sizes: 25 for the Householder TT-vector $\mathbf{u}_k$ and 30 for $\mathbf{q}_k$. For the last rounding accuracy value $\delta = 10^{-8}$, the maximal TT-rank of the MGS orthogonal TT-vector reaches the Householder $\mathbf{q}_k$ when the basis size exceeds 15, surpassing the maximal TT-rank of the Householder $\mathbf{u}_k$ at approximately the same basis size. These results are displayed in Figure 6c.

In analyzing the memory requirements of the TT-Householder algorithm, we also examine the compression ratio of the TT-vectors that form the orthogonal basis generated by the four orthogonalization methods. The compression ratio curves in Figures 6d, 6e and 6f have the same slopes as their corresponding maximal TT-rank curves, but they clearly demonstrate the memory needs. The orthogonal TT-vectors obtained from the TT-CGS and TT-CholQR schemes require only about 1% of the memory needed to store the full format tensors, as shown in Figure 6d, 6e, and 6f. However, when the basis size exceeds 10 and the TT-vectors become more collinear, the resulting basis from these schemes become very poor in terms of orthogonality. Both Figures 6d and 6e indicate that storing the basis TT-vectors generated by the TT-



(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

Maximal TT-rank for the orthogonal basis

(d) $\delta = 10^{-3}$ (e) $\delta = 10^{-5}$ (f) $\delta = 10^{-8}$

Compression ratio for the orthogonal basis

(g) $\delta = 10^{-3}$ (h) $\delta = 10^{-5}$ (i) $\delta = 10^{-8}$

Compression gain for the orthogonal basis

Fig. 6 Comparison of the orthogonal basis memory requirement for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$. The curves get dashed and partially transparent when their corresponding loss of orthogonality gets greater than the prescribed rounding accuracy δ.

Householder scheme requires approximately 20% of the memory needed to store those tensors in full format. Similarly only 10% of the entire memory required for full format storage is necessary to store the Householder TT-vectors $\mathbf{u}_k$. Finally, for $\delta = 10^{-8}$, the cost of storing the Householder basis TT-vectors $\mathbf{q}_k$ is the same as storing them in full format, as shown in Figure 6f. However, based on the same figure, it is evident that storing the Householder TT-vectors, even for $\delta = 10^{-8}$, requires only about 30% of the memory needed to store the same tensors in full format. This feature makes the TT-Householder algorithm highly appealing, as it is usually adequate to store only the

Householder TT-vectors. The TT-Householder algorithm becomes even more advantageous when compared to the compression ratio curves of the TT-MGS, TT-MGS2 and TT-CGS2 algorithms. For all rounding accuracy values, the compression ratio curve of the TT-MGS and TT-MGS2 always reaches 1, as shown in Figures 6d, 6e, and 6f. This implies that the memory required by the TT-vectors from these schemes is the same as the memory needed to store the orthogonal basis tensors in full format. This consideration also applies for the compression ratio of the orthogonal basis generated by the TT-CGS2 for $\delta \in \{10^{-5}, 10^{-8}\}$. However, for $\delta = 10^{-3}$, the TT-vectors from TT-CGS2 only require 20% of the memory needed to store the same tensors in dense format. Figures 6g, 6h, and 6i show the compression gain curves for the different rounding accuracy values δ with a similar behavior. The compression gain of TT-CholQR has a peak at the beginning and then stabilizes around 10 starting from $k = 10$. This means that during compression, the j-th basis TT-vector reduces the memory requirement by 10 times for $j \geq k$. The gain curves of TT-MGS, TT-MGS2 and TT-CGS2 have a similar shape. They increase up to $k = 15$, then drop for the next 5 iterations before rising again around $k = 22$. The gain curve of TT-Householder basis follows a similar pattern, growing to a peak before decreasing and then rising again during the last iterations. As previously observed, the Householder TT-vector gain curve slightly rises at the beginning, then it drops down and stagnates at a very low value from $k > 10$. Finally, the TT-CGS gain curve increases as the dimension k of the basis increases. When the last basis TT-vector is rounded, only 0.001 of the memory used to store the TT-vector before rounding is required. However, as indicated by the dash style, both the TT-CGS and TT-CholQR bases have almost completely lost the orthogonality already at $k = 10$.

4.5 Summary

In terms of memory footprint, the TT-Householder orthogonalization scheme, along with TT-CGS2 and TT-MGS2, is often the best option due to its stability property and the option to store only the TT-vectors $\mathbf{u}_i$. Figures 6d, 6e, and 6f demonstrate that when the input TT-vectors are not highly collinear ($k < 15$), TT-MGS2 and TT-CGS2 achieve a compression ratio similar to that of TT-Householder. For a basis size between 15 and 25 (or 20 for $\delta = 10^{-8}$), TT-Householder is more memory-expensive than TT-MGS, TT-MGS2 and TT-CGS2. Finally, as the input TT-vectors become more linearly dependent, the memory requirements of the TT-MGS2 and TT-CGS2 bases become greater or equal to those of both the TT-Householder basis and Householder TT-vector. However, in terms of orthogonality preservation, TT-MGS2 outperforms TT-Householder for every rounding accuracy, while TT-CGS2 outperforms TT-Householder for $\delta \in \{10^{-5}, 10^{-8}\}$. In terms of computationa cost, both TT-CGS2 and TT-MGS2 are cheaper than TT-Householder, requiring only $2m$ `TT-round` instead of $4m$.

5 Concluding remarks

In the framework where data representation accuracy is decoupled from computational accuracy, as previously proposed in [17, 30, 31], we investigate the loss of orthogonality of six orthogonalization algorithms in the tensor format. The Tensor Train [15] is the compressed format used to represent tensors. The orthogonalization methods considered are Classical and Modified Gram-Schmidt (CGS, MGS), their versions with re-orthogonalization (CGS2, MGS2), the Cholesky-QR approach, and the Householder transformation. As in the matrix case, the choice of the orthogonalization scheme among TT-Householder, TT-CGS2, and TT-MGS2 depends strongly on the purpose and on the available computing resources. TT-Householder requires less memory, but it is computationally more expensive and its orthogonality stagnates around the rounding accuracy. On the other hand, TT-MGS2 produces a basis of better orthogonality quality, as long as the input TT-vectors are not too collinear, and it is computationally cheaper than TT-Householder. The same considerations hold also for TT-CGS2, under the same hypothesis.

The computed orthogonalization qualities of these schemes are essentially the same in TT format calculation as they are in finite arithmetic for matrix computation, with the unit rounding error replaced by the TT rounding threshold. These similar behavioral results extend from matrix computation to TT-tensors; however, the technical analysis tools for formal proof use for rounding error analysis do not naturally extend to the TT-analysis. Identifying the appropriated analysis tools and establishing these formal proofs constitutes an area for future research.

Appendix A Non-standard inner product

In classical linear algebra, the six orthogonalization algorithms can be reformulated taking into account an inner product induced by a symmetric and positive definitive matrix, M. Thanks to its properties, M can be factorized as $M = (M^{\frac{1}{2}})^T M^{\frac{1}{2}}$. Thus, the inner product induced by M

$$\langle x, y\rangle_M = \langle M^{\frac{1}{2}}x, M^{\frac{1}{2}}y\rangle = x^T My$$

for every x and y vectors of compatible size. The norm associated to this induced inner product is $\|x\|_M = \|M^{\frac{1}{2}}x\|$.

In the context of multilinear operator, a TT-matrix, $\mathcal{M}$, is symmetric and positive definitive properties if

$$\langle \mathcal{M}\mathbf{x}, \mathbf{x}\rangle > 0 \qquad \text{and} \qquad \mathcal{M}(\mathbf{i}, \mathbf{j}) = \mathcal{M}(\mathbf{j}, \mathbf{i})$$

for every TT-vector $\mathbf{x}$ of compatible size, and $\mathbf{i}, \mathbf{j}$ multi-index of appropriate length. The inner product induced by $\mathcal{M}$ is

$$\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{M}} = \langle \mathcal{M}\mathbf{x}, \mathbf{y}\rangle$$

where the right-most inner product is the standard one. Similarly, every occurrence of the Frobenius norm is replaced as well by the norm induced by $\mathcal{M}$, computed as

$$\|\mathbf{x}\|_{\mathcal{M}}^2 = \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{M}}.$$

We use the inner product, as we are not aware of an existing tensor technique that factorizes a TT-matrix as $\mathcal{M} = (\mathcal{M}^{\frac{1}{2}})^T \mathcal{M}^{\frac{1}{2}}$. Every occurrence of the standard inner product and the Frobenius norm in the six TT-orthogonalization algorithms is replaced by the inner product and the norm induced by $\mathcal{M}$. The only non-trivial modification appears in the Householder algorithm. This scheme relies on an orthogonal basis to compute the Householder reflectors and the new orthogonal set of TT-vectors. The canonical basis is used, when the standard inner product is considered. However, a basis orthogonal with respect to the inner product induced by $\mathcal{M}$ is needed, because of the non-standard inner product chosen. As suggested in [32], we replace the canonical basis in the TT-Householder algorithm with a set of TT-vectors orthogonal with respect to $\mathcal{A}$ generated with TT-MGS2 at accuracy $1e-15$. This is clearly a limit for the performances of TT-Householder.

We measure the loss of orthogonality as $\|\mathbb{I}_k - Q^T M Q_k\|$ where the (i, j)-th element of $Q^T M Q_k \in \mathbb{R}^{k \times k}$ is

$$\left(Q^T M Q_k\right)(i, j) = \langle \mathbf{q}_i, \mathbf{q}_j \rangle_{\mathcal{M}}$$

for $\mathbf{q}_i, \mathbf{q}_j$ elements in the basis produced by one of the six algorithms, for increasing basis size k.

A.1 Experiments

To support our statement, we investigate experimentally the loss of orthogonality of the six algorithms considering two different non-standard inner products. For both the cases, we generate the input set of $m = 64$ TT-vectors of order 3 and mode size $n = 15$, using the same Krylov process described in Section 4.1. The rounding accuracy δ takes values in $\{1e-3, 1e-5, 1e-8\}$.

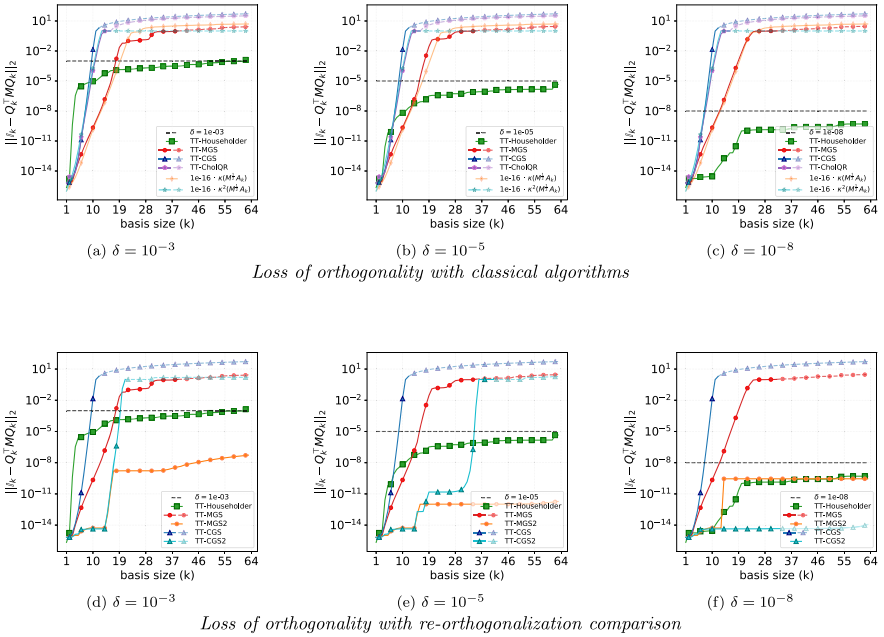
A.1.1 Multidiagonal case

The first TT-matrix we consider is

$$\mathcal{M}_1 = D_1 \otimes \mathbb{I} \otimes \mathbb{I} + \mathbb{I} \otimes D_2 \otimes \mathbb{I} + \mathbb{I} \otimes \mathbb{I} \otimes D_3$$

where $D_i = \text{diag}(d_i)$ with $d_i \in \mathbb{R}^n$ a randomly generated vector from the uniform distribution of $(0, 1)$ for $i = 1, 2, 3$. By construction, this TT-matrix is symmetric and positive definite.

Figure 7 displays the loss of orthogonality in function of the basis size, varying from 1 to m, using the inner product induced by $\mathcal{M}_1$. In Figures 7a, 7b and 7c, we display also $\kappa_2(M_1^{\frac{1}{2}} A_k)$ where $M_1 = \text{mat}(\mathcal{M}_1)$, and the j-th column of A_k is the j-th TT-vector input vectorized. These experimental results are consistent with those



(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms

(d) $\delta = 10^{-3}$ (e) $\delta = 10^{-5}$ (f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

Fig. 7 Loss of orthogonality and condition number with non-standard inner product for $m = 64$ TT-vectors of order $d = 3$ and mode size $n = 15$. The loss of orthogonality curves get dashed and partially transparent when they get greater than 1, while the condition number curves do when they get larger than 1.

presented in Section 4.1. They are coherent also with the theoretical results in matrix format presented in [33] for CGS, MGS with and without reorthogonalization, and in [32] for the Householder. The worsened TT-Householder performances depend likely on the input basis of TT-vectors, which is generated with TT-MGS2 using rounding accuracy equal to 10^{-15}.

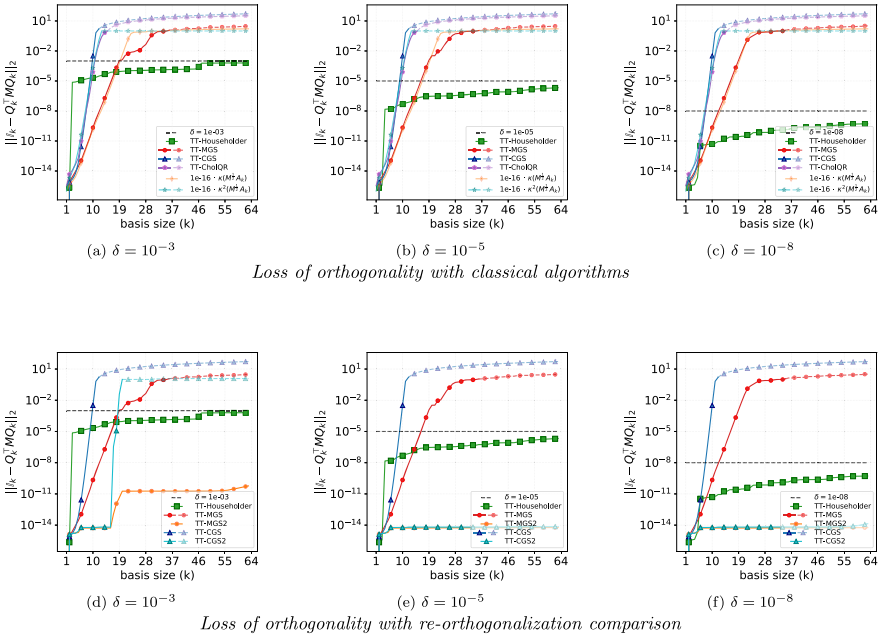
A.1.2 Laplacian-like case

The second TT-matrix is

$$\mathcal{M}_2 = T_1 \otimes \mathbb{I} \otimes \mathbb{I} + \mathbb{I} \otimes T_2 \otimes \mathbb{I} + \mathbb{I} \otimes \mathbb{I} \otimes T_3$$

where every $T_i = (i+1)\text{tridiag}(-b_i, a_i, -b_i)$ with a_i and b_i randomly generated numbers from the uniform distribution in $(1, 2)$ and $(0, 1)$, respectively, for $i = 1, 2, 3$. By construction, this TT-matrix is symmetric and positive definite, and not a diagonal operator.

As in previous section, we illustrate in Figure 8 the loss of orthogonality in function of the basis size, varying from 1 to m, using the inner product induced by $\mathcal{M}_2$. Figures 8a, 8b and 8c present $\kappa_2(M_2^{\frac{1}{2}} A_k)$ where $M_2 = \text{mat}(\mathcal{M}_2)$, and the j-th column of A_k is the j-th TT-vector input vectorized. The remarks made in previous section are still true.



(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms

(d) $\delta = 10^{-3}$ (e) $\delta = 10^{-5}$ (f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

Fig. 8 Loss of orthogonality and condition number with non-standard inner product for $m = 64$ TT-vectors of order $d = 3$ and mode size $n = 15$. The loss of orthogonality curves get dashed and partially transparent when they get greater than 1, while the condition number curves do when they get larger than 1.

Acknowledgements The authors thank the reviewers for their careful reading of the manuscript and their insightful comments. They express their sincere gratitude to Professor Elias Jarlebring, Managing Editor, and Professor Gunilla Kreiss, Editor-in-Chief of *BIT*, for proposing the publication of this work in the special issue dedicated to Professor Åke Björck. The authors also thank the Guest Editors of the special issue, Professor Lothar Reichel and Professor Emeritus Lars Elden, for kindly accepting this paper for inclusion in the special issue. Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr).

Funding Open access funding provided by Alma Mater Studiorum - Università di Bologna within the CRUI-CARE Agreement. The third author is member of the INdAM Research Group GNCS. Moreover, her work was partially supported by the European Union - NextGenerationEU under the National Recovery and Resilience Plan (PNRR) - Mission 4 Education and research - Component 2 from Research to Business - Investment 1.1 Notice Prin 2022 - DD N. 104 of 2/2/2022, entitled "Low-Rank Structures and Numerical Methods in Matrix and Tensor Computations and their Application", code 20227PCCKZ – CUP J53D23003620006.

Declarations

Conflicts of Interest This study does not have any conflicts to disclose.

References

1. Saad, Y.: *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, (2003) https://doi.org/10.1137/1.9780898718003
2. Saad, Y.: Numerical Methods for Large Eigenvalue Problems. Society for Industrial and Applied Mathematics (2011). https://doi.org/10.1137/1.9781611970739
3. Gram, J.P.: "Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate". In: *Journal f ü r die reine und angewandte Mathematik (Crelles Journal)* 1883.94 (1883), pp. 41–73
4. Schmidt, E.: "Zur Theorie der linearen und nichtlinearen Integralgleichungen". In: *Mathematische Annalen* 63.4 (Dec. 1907), pp. 433–476. https://doi.org/10.1007/BF01449770
5. Abdelmalek, N.N.: "Round-off error analysis for Gram-Schmidt method and solution of linear least squares problems". In: *BIT Numerical Mathematics* 11.4 (Dec. 1971), pp. 345–367. https://doi.org/10.1007/BF01939404
6. Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W.: "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization". In: *Mathematics of Computation* 30.136 , pp. 772–795 (1976)
7. Parlett, B.: *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, (1980)
8. Stathopoulos, A., Wu, K.: "A Block Orthogonalization Procedure with Constant Synchronization Requirements". In: *SIAM Journal on Scientific Computing* 23.6 , pp. 2165–2182. (2002) https://doi.org/10.1137/S1064827500370883
9. Householder, A.S.: "Unitary Triangularization of a Nonsymmetric Matrix". In: *J. ACM* 5.4 (Oct. 1958), pp. 339–342. https://doi.org/10.1145/320941.320947
10. Smoktunowicz, A., Barlow, J.L., Langou, J.: "A note on the error analysis of classical Gram–Schmidt". In: *Numerische Mathematik* 105.2 (Dec. 2006), pp. 299–313. https://doi.org/10.1007/s00211-006-0042-1
11. Björck, Å.: "Solving linear least squares problems by Gram-Schmidt orthogonalization". In: *BIT Numerical Mathematics* 7.1 (Mar. 1967), pp. 1–21. https://doi.org/10.1007/BF01934122
12. Giraud, L., Langou, J., Rozložník, M., Eshof, J.v.d.: "Rounding error analysis of the classical Gram-Schmidt orthogonalization process". In: *Numerische Mathematik* 101.1 (July 2005), pp. 87–100. https://doi.org/10.1007/s00211-005-0615-4
13. Wilkinson, J.H.: "Modern Error Analysis". In: *SIAM Review* 13.4 (Oct. 1971), pp. 548–568. https://doi.org/10.1137/1013095
14. Oseledets, I.V.: "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 , pp. 2295–2317. (2011) https://doi.org/10.1137/090752286
15. Oseledets, I.V., Tyrtyshnikov, E.E.: "Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions". In: *SIAM Journal on Scientific Computing* 31.5 , pp. 3744–3759. (2009) https://doi.org/10.1137/090748330
16. Dolgov, S.V.: "TT-GMRES: solution to a linear system in the structured tensor format". In: *Russian Journal of Numerical Analysis and Mathematical Modelling* 28.2 , pp. 149–172. (2013) https://doi.org/10.1515/rnam-2013-0009
17. Coulaud, O., Giraud, L., Iannacito, M.: "A note on TT-GMRES for the solution of parametric linear systems". In: *ETNA - Electronic Transactions on Numerical Analysis* , pp. 163–187. ISSN: 1068-9613. (2025) https://doi.org/10.1553/etna_vol62s163
18. Coulaud, O., Giraud, L., Iannacito, M., Issa. M.: *Solving eigenvalue problems in high dimensions using contour integration and Tensor Train format*. Tech. rep. RR-9586. Inria, Apr. 2025, p. 19
19. Paige, C.C., Rozloznik, M., Strakoš, Z.: "Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES". In: *SIAM Journal on Matrix Analysis and Applications* 28.1 , pp. 264–284. (2006) https://doi.org/10.1137/050630416

20. De Lathauwer, L., De Moor, B., Vandewalle, J.: "A Multilinear Singular Value Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 , pp. 1253–1278. (2000) https://doi.org/10.1137/S0895479896305696
21. Grasedyck, L.: "Hierarchical Singular Value Decomposition of Tensors". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 , pp. 2029–2054. (2010) https://doi.org/10.1137/090764189
22. Oseledets, I.V.: "DMRG Approach to Fast Linear Algebra in the TT-Format". In: *Computational Methods in Applied Mathematics* 11.3 , pp. 382–393. (2011) https://doi.org/10.2478/cmam-2011-0021
23. Gelß, P.: "The Tensor-Train Format and Its Applications". PhD thesis. Freien Universitat Berlin, 2017
24. Al Daas, H., Ballard, G., Cazeaux, P., Hallman, E., Miedlar, A., Pasha, M., Reid, T.W., Saibaba, A.K.: "Randomized Algorithms for Rounding in the Tensor-Train Format". In: *SIAM Journal on Scientific Computing* 45.1 , A74–A95. (2023) https://doi.org/10.1137/21M1451191
25. Che, M., Wei, Y.: "Randomized algorithms for the approximations of Tucker and the tensor train decompositions". In: *Advances in Computational Mathematics* 45.1 (Feb. 2019), pp. 395–428. https://doi.org/10.1007/s10444-018-9622-8
26. Hoffmann, W.: "Iterative algorithms for Gram-Schmidt orthogonalization". In: *Computing* 41.4 (Dec. 1989), pp. 335–348. https://doi.org/10.1007/BF02241222
27. Trefethen, L.N., Bau, D.: *Numerical Linear Algebra*. SIAM, (1997)
28. Wilkinson, J.H.: *The algebraic eigenvalue problem*. en. Numerical Mathematics and Scientific Computation. Oxford, England: Clarendon Press, Jan. (1965)
29. Kazeev, V.A., Khoromskij, B.N.: "Low-Rank Explicit QTT Representation of the Laplace Operator and Its Inverse". In: *SIAM Journal on Matrix Analysis and Applications* 33.3 , pp. 742–758. (2012) https://doi.org/10.1137/100820479
30. Agullo, E., Coulaud, O., Giraud, L., Iannacito, M., Marait, G., Schenkels, N.: *The backward stable variants of GMRES in variable accuracy*. Research Report RR-9483. Inria, Sept. , pp. 1–77 (2022)
31. Iannacito, M.: "Numerical linear algebra and data analysis in large dimensions using tensor format". PhD thesis. Inria Center at the University of Bordeaux, France, Dec. (2022)
32. Shao, M.: "Householder Orthogonalization with a Nonstandard Inner Product". In: *SIAM Journal on Matrix Analysis and Applications* 44.2 , pp. 481–502. (2023) https://doi.org/10.1137/21M1414814
33. Rozložník, M., Tůma, M., Smoktunowicz, A., Kopal, J.: "Numerical stability of orthogonalization methods with a non-standard inner product". In: *BIT Numerical Mathematics* 52.4 (Dec. 2012), pp. 1035–1058. ISSN: 1572-9125. https://doi.org/10.1007/s10543-012-0398-9