On some orthogonalization schemes in Tensor Train format

Olivier Coulaud, Luc Giraud, Martina Iannacito

On some orthogonalization schemes in Tensor Train format

Olivier Coulaud,, Luc Giraud, Martina Iannacito,

RESEARCH REPORT
N° 9491
November 2022
Project-Team Concace

ISSN 0249-6399 ISRN INRIA/RR--9491--FR+ENG

On some orthogonalization schemes in Tensor Train format

Olivier Coulaud[*],, Luc Giraud[*], Martina Iannacito[†],

Project-Team Concace

Research Report n° 9491 — version 2 — initial version November 2022 — revised version January 2024 — 27 pages

Abstract: In the framework of tensor spaces, we consider orthogonalization kernels to generate an orthogonal basis of a tensor subspace from a set of linearly independent tensors. In particular, we experimentally study the loss of orthogonality of six orthogonalization methods, namely Classical and Modified Gram-Schmidt with (CGS2, MGS2) and without (CGS, MGS) re-orthogonalization, the Gram approach, and the Householder transformation. To overcome the curse of dimensionality, we represent tensors with a low-rank approximation using the Tensor Train (TT) formalism. In addition, we introduce recompression steps in the standard algorithm outline through the TT-rounding method at a prescribed accuracy. After describing the structure and properties of the algorithms, we illustrate their loss of orthogonality with numerical experiments. The theoretical bounds from the classical matrix computation round-off analysis, obtained over several decades, seem to be maintained, with the unit round-off replaced by the TT-rounding accuracy. The computational analysis for each orthogonalization kernel in terms of the memory requirements and the computational complexity measured as a function of the number of TT-rounding, which happens to be the most computationally expensive operation, completes the study.

Key-words: Classical Gram-Schmidt, Modified Gram-Schmidt, Householder transformation, loss of orthogonality, Tensor Train format

[*] Inria, Inria centre at the University of Bordeaux

[†] Group Science, Technology and Engineering, KU Leuven - Kulak, Departement of Electrical Engineering, KU Leuven, Belgium

À propos de schémas d'orthogonalisation dans le format Tensor Train

Résumé : Dans le contexte de l'espace tensoriel, nous considérons les noyaux d'orthogonalisation pour générer une base orthogonale d'un sous-espace tensoriel à partir d'un ensemble de tenseurs linéairement indépendants. En particulier, nous étudions numériquement la perte d'orthogonalité de six méthodes d'orthogonalisation, à savoir les méthodes de Gram-Schmidt classique et modifiée avec (CGS2, MGS2) et sans (CGS, MGS) réorthogonalisation, l'approche de Gram et la transformation de Householder. Pour lutter contre la malédiction de la dimensionnalité, nous représentons les tenseurs avec le formalisme Train de tenseurs (TT), et nous introduisons des étapes de recompression dans le schéma de l'algorithme standard par la méthode TT-rounding à une précision prescrite. Après avoir décrit la structure et les propriétés de l'algorithme, nous illustrons numériquement que les limites théoriques de la perte d'orthogonalité dans le calcul matriciel classique sont maintenues, l'arrondi unitaire étant remplacé par la précision de l'arrondi TT. L'analyse pour chaque noyau d'orthogonalisation de l'exigence de mémoire et de la complexité de calcul en termes d'arrondi TT, qui se trouve être l'opération la plus coûteuse en termes de calcul, complète l'étude.

Mots-clés : Gram-Schmidt classique, Gram-Schmidt modifiée, transformation de Householder, perte d'orthogonalité, Tensor Train format

Contents

1 Introduction

The solution of linear problems is at the heart 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, 6, 7], the Gram approach [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 Gram method 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, 11] 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 [11] 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 Gram method are proven in [12] and [8], respectively. Collectively, these theoretical results are several decades old. From Wilkinson's oldest paper in 1965 for the Householder transformation to the most recent method presented in 2006 by Barlow et al. for the CGS2 scheme.

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 kernels are used in iterative methods to solve linear systems structured with the tensor product or in generalization of the least-squares problem to the tensor space. 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 kernels previously mentioned to tensors, using the Tensor Train (TT) formalism [13, 14]. These kernels in TT-format can be used, for example, in TT-algorithms such as TT-GMRES [15]. However, the operation sequences between tensors in TT-formats reduce the benefit of this compressed representation. Therefore, we introduce additional compression steps by the `TT-round` function [13] in the orthogonalization schemes, knowing that they affect the orthogonality quality of the basis. The aim of this work is to describe the orthogonalization kernels generalized to the tensor with the TT-format and to experimentally investigate the loss of orthogonality of the computed basis. These numerical results in TT-format show a similarity with the theoretical results of classical numerical matrix computation.

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 kernels and their TT-versions are summarized in Section 5.

2 Notation and TT-format

To enhance readability, we utilize the following notations 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 Frobenious 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 $k \in \{1,\ldots,d\}$. 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 $n = \max_{i\in\{1,\ldots,d\}}\{n_i\}$. Therefore, various compression techniques have been proposed over the years to reduce the memory consumption [16, 17, 18]. For the purpose of this work the most suitable tensor representation is the *Tensor Train* (TT) format [18]. 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 $k \in \{1,\ldots,d\}$, 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. The k-th TT-core of a tensor is denoted by the same bold letter underlined with a subscript 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.,

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

with $\underline{X}_k(i_k) \in \mathbb{R}^{r_{k-1}\times r_k}$ for every $i_k \in \{1,\ldots,n_k\}$ and $k \in \{2,\ldots,d-1\}$, 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 $n = \max_{i\in\{1,\ldots,d\}}\{n_i\}$ and $r = \max_{i\in\{1,\ldots,d\}}\{r_i\}$. 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.,

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

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 [19]. To address the issue of the TT-rank growth, a rounding algorithm to reduce it was proposed in [18]. The `TT-round` algorithm takes a TT-vector $\mathbf{x}$ and a relative accuracy δ as inputs and returns a TT-tensor $\tilde{\mathbf{x}}$, that is at a relative distance δ from $\mathbf{x}$, i.e., $||\mathbf{x} - \tilde{\mathbf{x}}|| \leq \delta ||\mathbf{x}||$. The TT-round function is fully described in [13]. For large-scale tensors, a randomized version of the TT-round function is described in [20, 21].

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

$$\frac{\sum_{i=1}^{d} r_{i-1} n_i r_i}{\sum_{j=1}^{d} s_{j-1} n_j s_j} \tag{2}$$

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 $\tilde{\mathbf{x}} = \texttt{TT-round}(\mathbf{x}, \delta)$. The computational cost of a `TT-round` over $\mathbf{x}$ in terms of floating point operations, is $\mathcal{O}(dnr^3)$, where $r = \max_{i\in\{1,\dots,d\}}\{r_i\}$ and $n = \max_{i\in\{1,\dots,d\}}\{n_i\}$, as stated in [18].

3 Orthogonalization schemes

In the following sections, we describe the classical orthogonalization kernels and we propose their extensions to the TT-format. The input of all the orthogonalization kernels 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 $\mathcal{A} = \{a_1, \dots, a_m\}$ 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 $\{q_1, \dots, q_{i-1}\}$, 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 $j \in \{1, \dots, i-1\}$. 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 1. TT-CGS initializes $\mathbf{p}_i$ to $\mathbf{a}_i$ for every $i \in \{1, \dots, m\}$, 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 $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_i\}$. The i-th column of R is defined using the projections of $\mathbf{a}_i$ along $\mathbf{q}_j$ for $j \in \{1, \dots, i-1\}$. 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 $j \in \{1, \dots, k-1\}$. 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 1 $\mathcal{Q}, R$ = TT-CGS($\mathcal{A}$, δ)

```
    input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
    output: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, $R$ the upper triangular matrix
 1: for $i = 1, \dots, m$ do
 2:     $\mathbf{p} = \mathbf{a}_i$
 3:     for $j = 1, \dots, i-1$ do
    ▷ compute the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
 4:         $R(i,j) = \langle \mathbf{a}_i, \mathbf{q}_j \rangle$
    ▷ remove the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
 5:         $\mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
 6:     end for
 7:     $\mathbf{p}$ = TT-round($\mathbf{p}, \delta$)
 8:     $R(i,i) = ||\mathbf{p}||$
 9:     $\mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$          ▷ normalize $\mathbf{p}$
10: end for
```

Algorithm 2 $\mathcal{Q}, R$ = TT-MGS($\mathcal{A}$, δ)

```
    input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
    output: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, $R$ the upper triangular matrix
 1: for $i = 1, \dots, m$ do
 2:     $\mathbf{p} = \mathbf{a}_i$
 3:     for $j = 1, \dots, i-1$ do
    ▷ compute the projection of $\mathbf{p}$ along $\mathbf{q}_j$
 4:         $R(i,j) = \langle \mathbf{p}, \mathbf{q}_j \rangle$
    ▷ remove the projection of $\mathbf{p}$ along $\mathbf{q}_j$
 5:         $\mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
 6:     end for
 7:     $\mathbf{p}$ = TT-round($\mathbf{p}, \delta$)
 8:     $R(i,i) = ||\mathbf{p}||$
 9:     $\mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$          ▷ 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. [10]. 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 $j \in \{1, \dots, i-1\}$. This can be seen by comparing line 4 of Algorithm 1 and 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 TT-MGS computational complexity in TT-format is equal to 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 [11], several articles, such as [5, 6, 22], have addressed this issue by introducing re-orthogonalization steps. This involves repeatedly orthogonalizing the basis using the same approach. In [11], 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 $\mathcal{Q}, R$ = TT-CGS2($\mathcal{A}$, δ)

```
    input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
    output: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, $R$ the upper triangular matrix
 1: for $i = 1, \dots, m$ do
 2:     $\mathbf{p}_0 = \mathbf{a}_i$
        ▷ repeat twice the orthogonalization loop
 3:     for $k = 1, 2$ do
 4:         $\mathbf{p}_k = \mathbf{p}_{k-1}$
 5:         for $j = 1, \dots, i-1$ do
                ▷ compute the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
 6:             $R_k(i,j) = \langle \mathbf{p}_{k-1}, \mathbf{q}_j \rangle$
                ▷ subtract the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
 7:             $\mathbf{p}_k = \mathbf{p}_k - R_k(i,j)\mathbf{q}_j$
 8:         end for
 9:         $\mathbf{p}_k$ = TT-round($\mathbf{p}_k, \delta$)
10:     end for
11:     $R_2(i,i) = ||\mathbf{p}_2||$
12:     $\mathbf{q}_i = 1/R_2(i,i)\mathbf{p}_2$        ▷ normalize $\mathbf{p}_2$
13: end for
    ▷ compute the R factor from the repeated orthogonalization loop
14: $R = R_1 + R_2$
```

Algorithm 4 $\mathcal{Q}, R$ = TT-MGS2($\mathcal{A}$, δ)

```
    input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
    output: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, $R$ the upper triangular matrix
 1: for $i = 1, \dots, m$ do
 2:     $\mathbf{p}_0 = \mathbf{a}_i$
        ▷ repeat twice the orthogonalization loop
 3:     for $k = 1, 2$ do
 4:         $\mathbf{p}_k = \mathbf{p}_{k-1}$
 5:         for $j = 1, \dots, i-1$ do
                ▷ compute the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
 6:             $R_k(i,j) = \langle \mathbf{p}_k, \mathbf{q}_j \rangle$
                ▷ subtract the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
 7:             $\mathbf{p}_k = \mathbf{p}_k - R_k(i,j)\mathbf{q}_j$
 8:         end for
 9:         $\mathbf{p}_k$ = TT-round($\mathbf{p}_k, \delta$)
10:     end for
11:     $R_2(i,i) = ||\mathbf{p}_2||$
12:     $\mathbf{q}_i = 1/R_2(i,i)\mathbf{p}_2$        ▷ 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, the input TT-vector $\mathbf{a}_i$ is orthogonalized with respect to the previously computed orthogonal TT-vectors $\{\mathbf{q}_1, \dots, \mathbf{q}_{i-1}\}$, 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, 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 $\{\mathbf{q}_1, \dots, \mathbf{q}_{i-1}\}$, 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. 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 4 and 3, respectively. In the first orthogonalization loop, that is, when $k = 1$ in line 3 of both methods, the classical Gram-Schmidt version projects $\mathbf{a}_i$ is 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 Gram approach

In their article [8], the authors propose an algorithm for generating an orthogonal basis from a set of m linearly independent vectors of $\mathbb{R}^n$ with $m \ll n$. We will refer to this algorithm as Gram's algorithm. This scheme is based on the Gram matrix, which under the hypothesis that m is significantly small. The key idea is to decompose the small Gram matrix by its Cholesky factorization and use it to generate the orthogonal basis. We briefly describe the main ideas of this orthogonalization scheme in the classical matrix framework and we describe in detail the implementation of the Gram algorithm in the TT-format. In fact, the tensor realization of this procedure is extremely close to 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 $\mathcal{A} = \{a_1, \dots, a_m\}$ 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 $i, j \in \{1, \dots, m\}$. 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

$$G = A^\top A. \tag{3}$$

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

$$G = R^\top R. \tag{4}$$

Comparing Equations (3) and (4), 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 $\mathcal{Q} = \{q_1, \dots, q_m\}$, 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 this 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 5, given a set of TT-vectors $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ 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 $\mathcal{Q}$ 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. As in Sections 3.1 and 3.2, note that the complexity of the TT-Gram

Algorithm 5 $\mathcal{Q}, R$ = TT-Gram($\mathcal{A}$, δ)

```
    input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ a set of TT-vectors, $\delta \in \mathbb{R}_+$ a relative rounding accuracy
    output: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ the set of orthogonal TT-vectors, $R$ the upper triangular matrix
 1: for $i = 1, \dots, m$ do
 2:     for $j = 1, \dots, i$ do
    ▷ construct the Gram matrix through the inner product of the input TT-vectors
 3:         $G(i,j) = G(j,i) = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$
 4:     end for
 5: end for
 6: $L$ = cholesky($G$)                                   ▷ compute the Cholesky factorization
 7: $R = L^\top$ and $R^{-1}$ = invert($R$)               ▷ define the R factor of the QR-factorization
 8: for $i = 1, \dots, m$ do
 9:     $\mathbf{p} = R^{-1}(i,1)\mathbf{a}_1$
10:     for $j = 2, \dots, i$ do
    ▷ construct the $i$-th new basis TT-vector as a linear combination of the $(i-1)$ input TT-vector
11:         $\mathbf{p} = \mathbf{p} + R^{-1}(i,j)\mathbf{a}_j$
12:     end for
13:     $\mathbf{q}_i$ = TT-round($\mathbf{p}, \delta$)          ▷ round the TT-vector before adding it to the basis
14: end for
```

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 $i \in \{1, \dots, m\}$, we conclude that the cost of the TT-Gram 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 due to their stability properties, as explained in the following sections. We will briefly present the theoretical construction of a Householder transformation and how it can be used to generate an orthogonal basis. The following section provides a detailed description of how the Householder transformation is extended to the tensor context, with specific attention given to the implementation of the Householder orthogonalization scheme in TT-format.

The Householder reflector is used to move a vector $x \in \mathbb{R}^n$ along a chosen direction, which is typically an element of the canonical basis or a linear combination of them. The construction of the Householder reflector in the general case is illustrated. Let $x \in \mathbb{R}^n$ be the vector we want to reflect along the normalized vector $y \in \mathbb{R}^n$, the *Householder reflection or transformation* is a linear operator $\boldsymbol{\mathcal{H}} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\boldsymbol{\mathcal{H}}(x) = \|x\|y \qquad \text{with} \qquad \|y\| = 1.$$

The Householder reflector is represented with respect to the canonical basis of $\mathbb{R}^n$ by the matrix $H \in \mathbb{R}^{n\times n}$ such that $H = \mathbb{I}_n - 2u \otimes u$ where $u \in \mathbb{R}^n$ is the *Householder vector* defined as

$$u = \frac{x - z}{\|x - z\|} \qquad \text{with} \qquad z = \|x\|y. \tag{5}$$

The Householder reflection matrix H is unitary and defined entirely by the Householder vector u. Additionally, the action of a Householder reflector is computed by one inner product with the Householder vector u and one algebraic vector summation. For a given vector $w \in \mathbb{R}^n$ and a Householder reflector $H = \mathbb{I}_n - 2u \otimes u$, the image of w through H is

$$Hw = w - 2\langle w, u\rangle u. \tag{6}$$

If u is defined as in Equation (5), then it can be verified that $Hx = \|x\|y$. It is important to note that

$$\begin{aligned}\|x - z\|^2 &= \langle x - \|x\|y,\, x - \|x\|y\rangle \\ &= \|x\|^2 - 2\|x\|\langle x, y\rangle + \|x\|^2\|y\|^2 \\ &= 2\big(\|x\|^2 - \|x\|\langle x, y\rangle\big)\end{aligned}$$

since $\|y\| = 1$ by hypothesis. Using this result and Equation (6), we can obtain

$$\begin{aligned}Hx &= x - 2\langle x, u\rangle u \\ &= x - \frac{2}{2\big(\|x\|^2 - \|x\|\langle x, y\rangle\big)}\langle x, x - \|x\|y\rangle(x - \|x\|y) \\ &= x - \frac{1}{\big(\|x\|^2 - \|x\|\langle x, y\rangle\big)}\big(\|x\|^2 - \|x\|\langle x, y\rangle\big)(x - \|x\|y) \\ &= x - (x - \|x\|y) \\ &= \|x\|y\end{aligned}$$

Householder transformations are commonly used to compute the QR factorization of a matrix, but they can also be applied when a set of vectors needs to be converted into an orthogonal basis. We will briefly examine the two possibilities. When given a matrix $A \in \mathbb{R}^{n\times m}$, we construct m Householder reflections. The k-th reflection moves the k-th column of A along a linear combination of the first k canonical basis vectors. In other words, the k-th Householder transformation sets the last $(n-k)$ entries of the k-th column of A to zero. As a result, after m Householder reflections, the matrix A becomes upper triangular. We will now provide a more detailed illustration of how the algorithm iteratively proceeds. To begin with, let $a_1 \in \mathbb{R}^n$ represent the first column of A. The first step is to reflect it along the first canonical basis vector e_1 by constructing the Householder reflector H_1 such that $H_1 a_1 = \|a_1\| e_1$. The first Householder vector $u_1 \in \mathbb{R}^n$ is then

$$u_1 = a_1 \pm \|a_1\| e_1$$

and normalized. For stability reasons (cf. [23]), the sign of the norm of a_1 is determined by the sign of the first component of a_1, which is positive if $a_1(1) > 0$ and negative otherwise. The first Householder reflector H_1 is applied to all the columns of A, resulting in $\tilde{a}_j = H_1 a_j$ for $j \in \{1, \dots, m\}$. From now on, $\tilde{a}_j$ denotes the j-th column of A updated by all the previously defined $(j-1)$ Householder transformations for $j \in \{2, \dots, m\}$. It is important to note that the first Householder transformation moves the first column of A along a multiple of the first canonical basis vector e_1, i.e., setting the last $(n-1)$ entries of the first column of A to zero. Next, we reflect the second column of A along a linear combination of the first two canonical basis vectors e_1 and e_2. We define $u_2 \in \mathbb{R}^n$ as the Householder vector that defines the second Householder reflector H_2. This second Householder reflector updates the j-th column of A for $j \in \{2, \dots, m\}$ a second time. At this point, only the first two entries of $\tilde{a}_2$ are different from zero. The k-th Householder reflection H_k moves $\tilde{a}_k \in \mathbb{R}^n$ the k-th column of A, updated by the first $(k-1)$ Householder reflections along a linear combination of the first k elements of the canonical basis of $\mathbb{R}^n$, i.e., $H_k \tilde{a}_k = \sum_{\ell=1}^{k} \alpha_\ell e_\ell$ with $\sqrt{\sum_{\ell=1}^{k} \alpha_\ell^2} = ||\tilde{a}_k||$. Prior to normalization, the k-th Householder vector, $u_k \in \mathbb{R}^n$, is defined as

$$u_k = \tilde{a}_k - \sum_{\ell=1}^{k} \beta_\ell e_\ell, \tag{7}$$

where

$$\beta_\ell = \tilde{a}_k(\ell) \qquad \text{and} \qquad \beta_k = \pm\sqrt{||\tilde{a}_k||^2 - \sum_{\ell=1}^{k-1} \beta_\ell^2},$$

for every $\ell \in \{1, \dots, k-1\}$. To ensure stability (cf. [23]), β_k is positive if $\tilde{a}_k(k)$ is positive, and negative otherwise. The vector u_k is then normalized.

Remark 3.3. *By construction, the first $(k-1)$ entries of u_k are zeros, the last $(n-k)$ entries are equal to the corresponding ones of $\tilde{a}_k$. The k-th component of u_k is obtained by subtracting the quantity β_k from the k-th component of a_k, that is $u_k(k) = \tilde{a}_k(k) - \beta_k$. In the context of matrices, the Householder QR factorization has a simplified construction due to the property that only the last $(n-k+1)$ entries of $\tilde{a}_k$ have a determinant role. The k-th Householder vector, $\hat{u}_k \in \mathbb{R}^{n-k+1}$, is defined from the norm of $\tilde{a}_k$, with the first $(k-1)$ entries being zeros. In other words,*

$$\hat{u}_k = \gamma_k e_1 \qquad \textit{where} \qquad \gamma_k = \sqrt{\sum_{j=k}^{n} \big(\tilde{a}_k(j)\big)^2}$$

with $e_1 \in \mathbb{R}^{n-k+1}$ *for every* $k \in \{1, \dots, m\}$. *The* k*-th Householder transformation,* $H_k \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$, *is defined based on* $\hat{u}_k$ *and is only applied on the last* $(n-k+1)$ *components of* $\tilde{a}_j$, *where* $j \in \{k, \dots, m\}$ *and* $k \in \{1, \dots, m\}$. *It is important to note that this reduced approach cannot be replicated in the tensor framework, where objects are expressed in compressed format. Therefore, we present it in the most general way, which is not the same as the approach used for in matrix computation.*

After applying the k-th Householder transformation to $\tilde{a}_j$ for $j \in \{k, \dots, m\}$, the vector $\tilde{a}_k$ will have its last $(n-k)$ component equal to zero. The application of m Householder reflections to A results in the upper triangular matrix R, which is the R factor of the QR decomposition. To calculate the Q factor, we multiply the m Householder reflection matrices. For most applications, it is sufficient to form the Householder vectors and know the Householder transformations implicitly, as given in Equation (6). However, if we want to produce an orthogonal basis from a generic set of m vectors $\mathcal{A} = \{a_1, \dots, a_m\}$ with $a_k \in \mathbb{R}^n$, we must take an additional step. When computing the QR factorization, the k-th Householder transformation, H_k, is defined by the k-th Householder vector, u_k, as given in Equation (7) for every $k \in \{1, \dots, m\}$. To generate the set of orthonormal vectors $\mathcal{Q} = \{q_1, \dots, q_m\}$, we apply the first k Householder transformations, H_k, to the k-th canonical basis vector, e_k, in reverse order. That is

$$q_k = H_1 \cdots H_k e_k.$$

We extend this approach to the tensor case with some modifications, as in the previous sections. Let $\mathcal{A}$ be a set of m TT-vectors $\mathbf{a}_i \in \mathbb{R}^{n_1\times\cdots\times n_d}$ for every $i \in \{1, \dots, m\}$. To construct the Householder transformations, we first define a *canonical* basis for a tensor subspace of dimension m of $\mathbb{R}^{n_1\times\cdots\times n_d}$. In order to do so, we fix $\mathcal{N}_i = \{1, \dots, n_i\}$ for every $i \in \{1, \dots, d\}$ and $n = \prod_{j=1}^{d} n_j$. We then define the function $\psi : \mathcal{N}_1 \times \cdots \times \mathcal{N}_d \to \{1, \dots, n\}$ such that

$$\psi(i_1, \dots, i_d) = i_1 + \sum_{\alpha=2}^{d} (i_\alpha - 1) m_\alpha \qquad \text{with} \qquad m_\alpha = \prod_{\beta=1}^{\alpha-1} n_\beta.$$

Since ψ is invertible, we denote its inverse by $\phi : \{1, \dots, n\} \to \mathcal{N}_1 \times \cdots \times \mathcal{N}_d$ such that $\phi(i) = (i_1, \dots, i_d)$. As a consequence, $\psi(\phi(i)) = i$ and $\phi(\psi(i_1, \dots, i_d)) = (i_1, \dots, i_d)$ for $i \in \{1, \dots, n\}$ and $i_k \in \{1, \dots, n_k\}$ with $k \in \{1, \dots, d\}$. The basis for the subspace of dimension m of $\mathbb{R}^{n_1\times\cdots\times n_d}$ is fixed as $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_m\}$ with

$$\mathbf{e}_i = e_{i_1} \otimes \cdots \otimes e_{i_d} \qquad \text{with} \qquad (i_1, \dots, i_d) = \phi(i), \tag{8}$$

where e_{i_k} is the i_k-th canonical basis vector of $\mathbb{R}^{n_k}$ for $k \in \{1, \dots, d\}$. The index i is used to denote the i-th element of the canonical basis, that is $i = \psi(i_1, \dots, i_d)$.

As stated in Remark 3.3, the vector u_k has zeros for its first $(k-1)$ components, the corresponding components of $\tilde{a}_k$ for its last $(n-k)$ entries, and the difference between the k-th component of $\tilde{a}_k$ and the quantity β_k (defined in Equation (7)) for its k-th component. This structure needs to be transported in the tensor case. However, if the elements of $\mathcal{A}$ are in TT-format, it is not possible to directly access the tensor components. We need to recover them either by multiplying the TT-cores with the correct index or by computing the inner product with the element of $\mathcal{E}$. The k-th Householder TT-vector is $\mathbf{u}_k \in \mathbb{R}^{n_1\times\cdots\times n_d}$ defined as

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

where $\tilde{\mathbf{a}}_k$ is the result of $(k-1)$ Householder reflections applied to $\mathbf{a}_k$, $R(j,k) = \langle \tilde{\mathbf{a}}_k, \mathbf{e}_j \rangle$ for every $j \in \{1,\dots,k\}$ and $R(k,k) = \pm\sqrt{||\tilde{\mathbf{a}}(k)_k||^2 - \sum_{\ell=1}^{k-1} R(\ell,k)^2}$ as described in lines 7 and 9 of Algorithm 6, respectively. The k-th component of r_k takes a positive sign if $\langle \tilde{\mathbf{a}}_k, \mathbf{e}_k \rangle > 0$; otherwise, it takes negative sign. This extends the stability preserving idea given in [23] to the tensor framework. The j-th component of r_k corresponds to the (j,k) component of the R factor. As previously mentioned, it is important to ensure that the TT-rank of $\mathbf{u}_k$ remains small for every $k \in \{1,\dots,m\}$. Assuming that the maximum TT-rank of $\tilde{\mathbf{a}}_k$ is bounded by r_a, then after removing the first $(k-1)$ components of $\tilde{\mathbf{a}}_k$ (after line 7), the TT-rank of $\mathbf{u}_k$ is bounded by $(r_a + k - 1)$. At this step, we make the first `TT-round` call on the Householder TT-vector $\mathbf{u}_k$, whose TT-rank decreases depending on the accuracy value δ. Then we subtract the k-th component of r_k, which results in further growth in the TT-rank of $\mathbf{u}_k$. As the $\mathbf{u}_k$ TT-vector plays a crucial role in the Householder transformation and its TT-rank has a significant impact on the entire process, we perform an additional `TT-round` step over $\mathbf{u}_k$ at accuracy δ.

Algorithm 6 $\mathbf{u}, r = \texttt{TTH-vec}(\mathbf{a}, \mathcal{F}, \delta)$

```
    input: $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ a TT-vector, $\mathcal{F} = \{\mathbf{f}_1, \dots, \mathbf{f}_i\}$ 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
 1: $s = 0$
 2: $\mathbf{w} = \mathbf{a}$
 3: for $j = 1, \dots, i-1$ do
    ▷ compute the component of $\mathbf{a}$ along the $j$-th canonical basis TT-vector
 4:     $r(j) = \langle \mathbf{a}, \mathbf{f}_j \rangle$
 5:     $s = s + \big(r(j)\big)^2$
    ▷ set to zero the component of $\mathbf{a}$ along the $j$-th canonical basis TT-vector $\mathbf{f}_j$
 6:     $\mathbf{w} = \mathbf{w} - r(j)\mathbf{f}_j$
 7: end for
 8: $\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
    ▷ subtract from the norm of $\mathbf{a}$ the contribution of the components set to zero
 9: $r(i) = \mathrm{sign}(\langle \mathbf{a}, \mathbf{f}_i \rangle)\sqrt{||\mathbf{a}||^2 - s}$
10: $\mathbf{w} = \mathbf{w} - r(i)\mathbf{f}_i$
11: $\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
12: $\mathbf{u} = (1/||\mathbf{z}||)\mathbf{w}$
```

After describing the construction of a Householder TT-vector, which is summarized in Algorithm 6, the focus shifts on generation of the orthonormal TT-vector set from a generic TT-vector set $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, as depicted in Algorithm 8. To reflect the k-th TT-vector of $\mathcal{A}$ along a linear combination of the first k elements of $\mathcal{E}$, we generate the k-th Householder TT-vector, $\mathbf{u}_k$, using Algorithm 6. This TT-vector defines the Householder transformation, $\mathbf{H}_k$, implicitly. The first k components of r_k are stored in the k-th column of the upper triangular matrix, $R \in \mathbb{R}^{m \times m}$. We apply $\mathbf{H}_k$ implicitly using $\mathbf{u}_k$ to form $\tilde{\mathbf{a}}_j$ for every $j \in \{k, \dots, m\}$, as expressed in line 6 of Algorithm 8. This follows the same approach as in the matrix case, where

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

Algorithm 7 applies a given Householder reflection to a specific input vector. It is important to note that $(k-1)$ transformations are performed on $\mathbf{a}_k$, computing $\tilde{\mathbf{a}}_k$, before generating the k-th reflector. This can potentially lead to a much larger maximum TT-rank of $\mathbf{a}_k$ that its initial value. To keep the TT-rank of $\tilde{\mathbf{a}}_k$ reasonably small, a `TT-round` is performed before generating the associated Householder TT-vector (see line 9).

Algorithm 7 $\mathbf{b} = \texttt{apply-H-vec}(\mathbf{a}, \mathbf{u})$

```
    input: $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ a TT-vector to project, $u \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ the Householder TT-vector
    output: $\mathbf{b}$ the TT-vector resulting from the Householder reflector defined by $\mathbf{u}$ applied to $\mathbf{a}$
 1: $\mathbf{b} = \mathbf{a} - 2\langle \mathbf{a}, \mathbf{u} \rangle \mathbf{u}$        ▷ apply the Householder reflection defined by $\mathbf{u}$ to $\mathbf{a}$
```

The conclusive part of the TT-Householder transformation algorithm 8 generates a new set $\mathcal{Q}$ of orthonormal TT-vectors. The i-th element of $\mathcal{Q}$, $\mathbf{q}_i$, is obtained by applying the first i Householder reflections in reverse order to $\mathbf{e}_i$ from the canonical basis $\mathcal{E}$ (see line 15 of Algorithm 8). 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, as shown in line 17 of Algorithm 8.

Algorithm 8 $\mathcal{Q}, R = \text{TT-Householder}(\mathcal{A}, \delta)$

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

The TT-Householder algorithm requires $4m$ `TT-round` operations, including two for each Householder TT-vector, one for each TT-vector $\tilde{\mathbf{a}}_k$ after the $(k-1)$-th reflection, and one for each $\mathbf{q}_k$ orthogonal TT-vector. Therefore, 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.

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 $\mathcal{A}_m = \{a_1, \dots, a_m\}$ 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 $\mathcal{Q}_m = \{q_m, \dots, q_m\}$ 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

$$||\mathbb{I}_m - Q_m^\top Q_m||_2. \tag{9}$$

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 [11, 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.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\big(u\kappa^2(A_m)\big) \tag{10}$$

as long as $\kappa^2(A_m)u \ll 1$. In [10], 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.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\big(u\kappa(A_m)\big) \tag{11}$$

as long as $\kappa(A_m)u \ll 1$. The authors of [8, Theorem 4.1], who proposed the Gram 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 Gram scheme from $\{a_1, \ldots, a_m\}$ satisfies the same upper bound as CGS, given in (10). The Householder orthogonalization algorithm is known 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 times the round-off unit, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\big(u\big) \tag{12}$$

as proven in [24]. 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 [11, 25], the loss of orthogonality of CGS2 and MGS2 satisfies the bound given in Equation (12), 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.

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 [26], and $\mathbf{a}_j$ is the normalized output of the `TT-round` algorithm applied to $\mathbf{x}_j$. 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 $\|\mathbb{I}_k - Q_k^\top Q_k\|_2$, 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.,

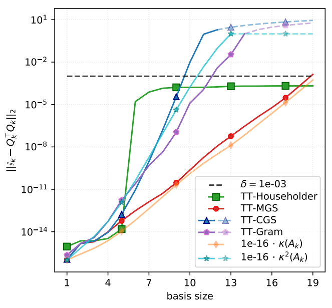
$$Q_k^\top Q_k(i, j) = \langle \mathbf{q}_i, \mathbf{q}_j \rangle$$

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

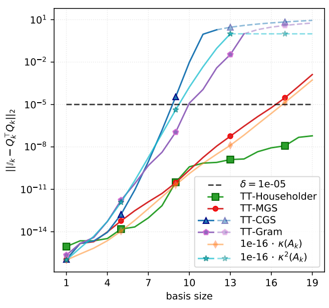
4.2 Order-3 experiments

In the 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 1 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 1 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 $\kappa(A_k)$.

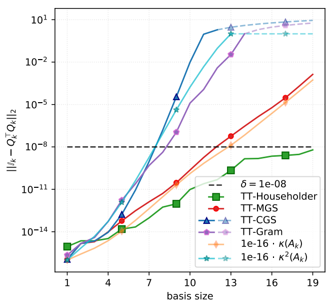
To aid interpretation, we present three plots in Figure 1a, 1b and 1c, which show the loss of orthogonality of standard methods in tensor format: TT-Householder, TT-MGS, TT-CGS, TT-Gram. Figures 1d, 1e and 1f show the loss of orthogonality of re-orthogonalization methods: TT-MGS2 and TT-CGS2, compared to TT-Householder, TT-MGS and TT-CGS. All six plots in Figure 1 exhibit similar behaviors. The loss of orthogonality of the TT-Householder method in green stagnates around the rounding accuracy δ, as shown in Figure 1a. This is consistent with the matrix theoretical expectation, stated in Equation (12), 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 (11). Finally, both the TT-CGS and and TT-Gram loss of orthogonality curves cross the rounding accuracy dashed line faster than TT-MGS. This curve follows the the



(a) $\delta = 10^{-3}$

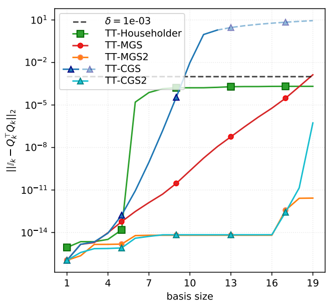


(b) $\delta = 10^{-5}$

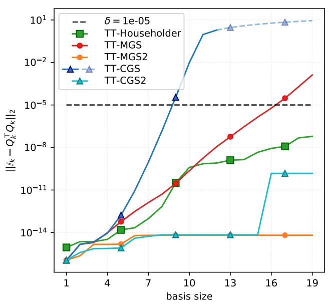


(c) $\delta = 10^{-8}$

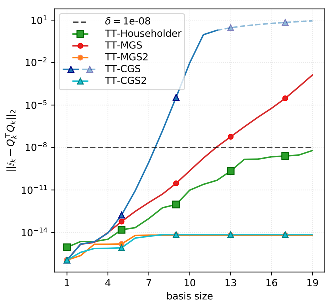
Loss of orthogonality with classical algorithms



(d) $\delta = 10^{-3}$



(e) $\delta = 10^{-5}$



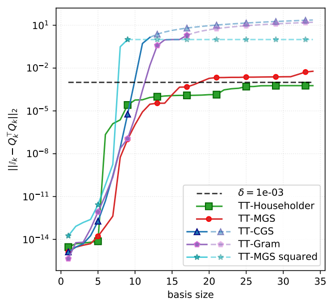
(f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

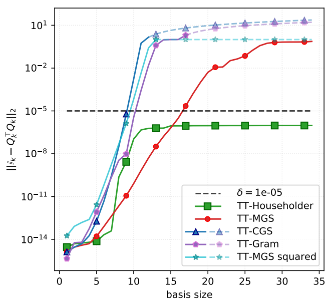
Figure 1: Loss of orthogonality and condition number for $m = 20$ TT-vectors of order $d = 3$ and mode size $n = 15$. The curves get dashed and partially transparent when they get greater than 1.

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-Gram 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 1d, 1e and 1f demonstrate a significant improvement in the 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.

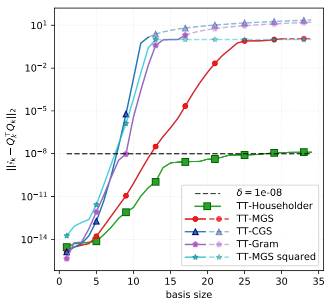
4.3 Order-6 experiments



(a) $\delta = 10^{-3}$

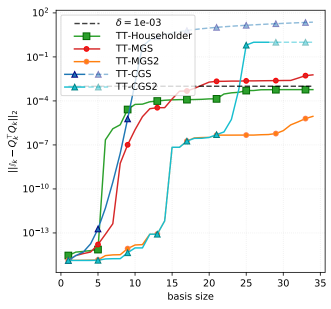


(b) $\delta = 10^{-5}$

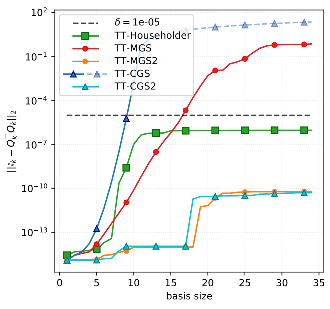


(c) $\delta = 10^{-8}$

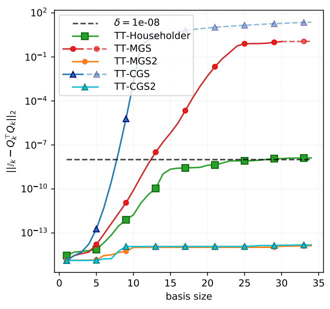
Loss of orthogonality with classical algorithms



(d) $\delta = 10^{-3}$



(e) $\delta = 10^{-5}$



(f) $\delta = 10^{-8}$

Loss of orthogonality with re-orthogonalization comparison

Figure 2: 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.

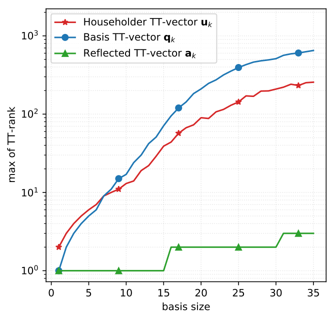
To further validate our results and to study their applicability to large-scale problems, we introduce a second experimental framework. We set the problem order to $d = 6$ with size mode $n_i = 15$ for $i \in \{1, \dots, 6\}$, and generate $m = 35$ TT-vectors, defining the set $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_{35}\}$. 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 2 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 2a, 2b and 2c 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 (12) still holds true in the tensor framework, with the unit round-off u replaced by the `TT-round` accuracy δ. In Figures 2a and 2b, 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-Gram 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 2. 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 2b and 2c demonstrate that the loss of orthogonality of TT-CGS and TT-Gram 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-Gram and TT-CGS increases as the squared condition number grows. Additionally, Figures 2d, 2e and 2f 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 2d- 2f, 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

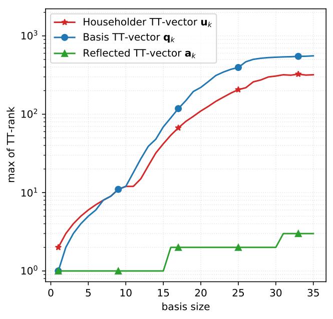
This section aims to analyze the effects of the orthogonalization process on the TT-rank and memory requirement based on experimental results. The growth of the TT-ranks, of the compression ratio (defined in Equation (1)), and the compression gain (cf. Equation (2)) curve of the orthogonal basis are 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

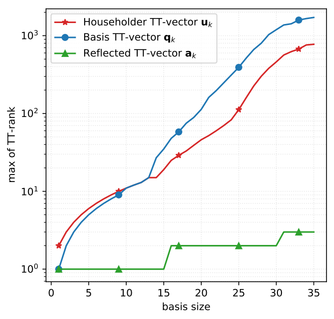
The Householder algorithm 8 applies the `TT-round` to three sets of TT-vectors: the Householder TT-vector $\mathbf{u}_k$, the TT-vector $\mathbf{a}_k$ (to which k Householder transformations are applied) and the orthogonal TT-vector $\mathbf{q}_k$ (obtained from the canonical basis TT-vectors with i 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 3 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 $k \in \{1, \dots, 35\}$ 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



(a) $\delta = 10^{-3}$

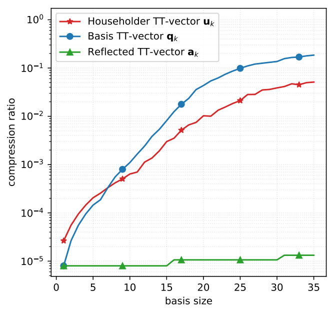


(b) $\delta = 10^{-5}$

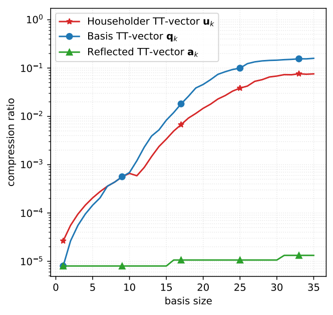


(c) $\delta = 10^{-8}$

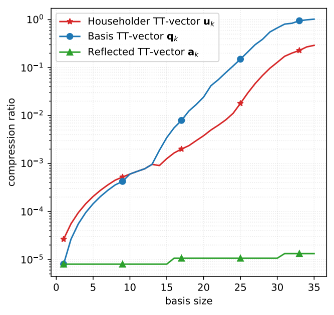
Maximal TT-rank for the TT-vectors in Householder algorithm



(d) $\delta = 10^{-3}$

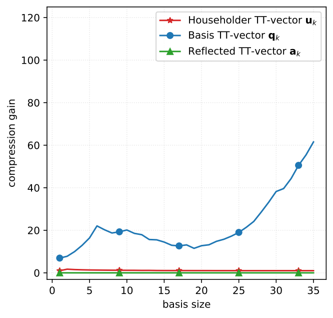


(e) $\delta = 10^{-5}$

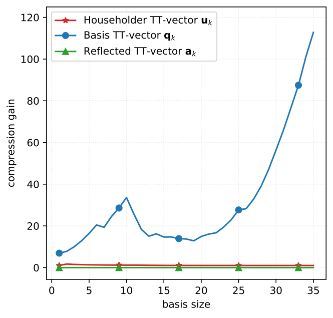


(f) $\delta = 10^{-8}$

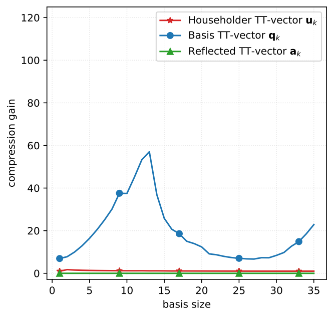
Compression ratio for the TT-vectors in Householder algorithm



(g) $\delta = 10^{-3}$



(h) $\delta = 10^{-5}$



(i) $\delta = 10^{-8}$

Compression gain for the TT-vectors in Householder algorithm

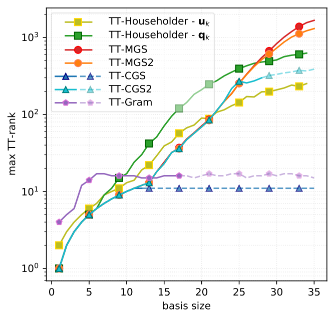
Figure 3: Householder memory requirement for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$.

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 3a, 3b and 3c 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 3a and 3b. This pattern is also observed in Figure 3c 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 3d, 3e and 3f show a compression ratio smaller than 1 for all rounding accuracy δ values, indicating that the TT-format is still effective in reducing the memory usage in all these experiments. Additionally, the compression ratio of $\mathbf{a}_k$ remains around 10^{-5} for all three values of δ. Figures 3d and 3e 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 3f, it is unclear whether the compression ratio of $\mathbf{q}_k$ and $\mathbf{u}_k$ plateaus at 1 and 10^{-1} respectively. In Figures 3g, 3i and 3h, the gain curves of $\mathbf{u}_k$, $\mathbf{q}_k$ and $\mathbf{a}_k$ exhibit same behavior. Specifically, we examine the gain of 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.

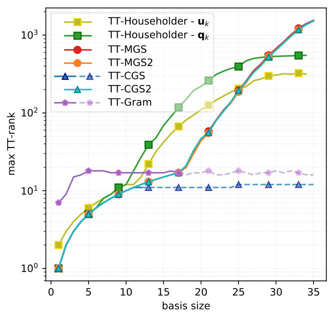
4.4.2 Orthogonalization kernel comparison

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 4 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 4a, 4b, and 4c show that the maximum TT-rank of the orthogonal TT-vectors computed by TT-CGS and TT-Gram 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-Gram 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-Gram 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-Gram 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 4a, 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 4b 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 4c.

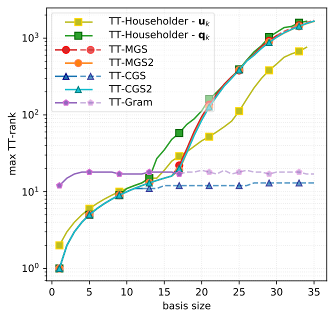
In analyzing the memory requirements of the TT-Householder algorithm, we also examine the compression ratio of the TT-vectors that form the orthogonal basis generated by the four orthogonalization methods. The compression ratio curves in Figures 4d, 4e and 4f have the same slopes as their corresponding maximal TT-rank curves, but they clearly demonstrate the memory needs. The orthogonal TT-vectors obtained from the TT-CGS and TT-Gram schemes require only about 1% of the memory needed to store the full format tensors, as shown in Figure 4d, 4e, and 4f. 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 4d and 4e indicate that storing the basis TT-vectors generated by the TT-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 4f. 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 4d, 4e, and 4f. 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 4g, 4h, and 4i show the compression gain curves for the different rounding accuracy values δ with a similar behavior. The compression gain of TT-Gram 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-Gram bases have almost completely lost the orthogonality already at $k = 10$.



(a) $\delta = 10^{-3}$

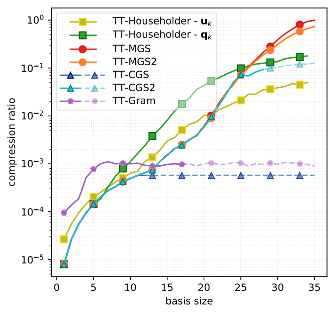


(b) $\delta = 10^{-5}$

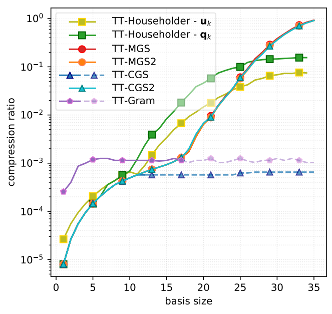


(c) $\delta = 10^{-8}$

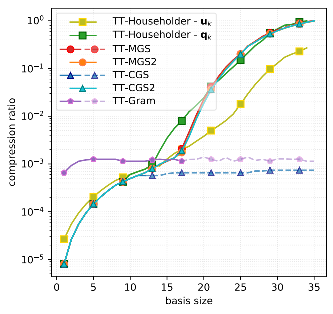
Maximal TT-rank for the orthogonal basis



(d) $\delta = 10^{-3}$

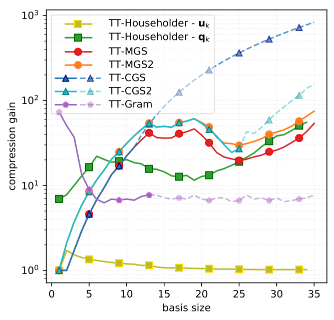


(e) $\delta = 10^{-5}$

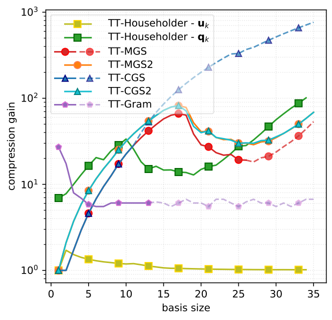


(f) $\delta = 10^{-8}$

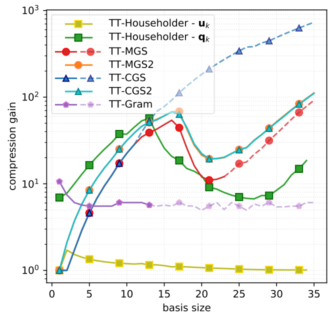
Compression ratio for the orthogonal basis



(g) $\delta = 10^{-3}$



(h) $\delta = 10^{-5}$



(i) $\delta = 10^{-8}$

Compression gain for the orthogonal basis

Figure 4: 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 then the prescribed rounding accuracy δ.

4.5 Summary

Table 1 summarizes the computational costs for the tensor and matrix cases. In terms of memory

	Matrix		TT-vectors	
Algorithm	*Computational cost in fp operations*	$\|\mathbb{I}_k - Q_k^\top Q_k\|$	*Computational cost in* `TT-round`	$\|\mathbb{I}_k - Q_k^\top Q_k\|$
Gram	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
CGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
MGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
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	$\mathcal{O}(2nm^2 - 2m^3/3)$	$\mathcal{O}(u)$	$4m$	$\mathcal{O}(\delta)$

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.

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 4d, 4e, and 4f 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 the data representation accuracy is decoupled from computational accuracy, as previously proposed in [27, 28, 29], we investigate the loss of orthogonality of six orthogonalization kernels in the tensor format. The Tensor Train [14] 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 Gram approach and the Householder transformation. Section 3 describes the generalization of these kernels to the tensor space in TT-format, relying on the compression function called `TT-round`. Section 4 presents the numerical experiments related to the loss of orthogonality and memory requirement of these kernels in TT-format.

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 theoretical validation of the experimental results remains an open question and will be the focus of future research.

References

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

[2] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011. DOI: 10.1137/1.9781611970739.

[3] J. P. Gram. "Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate". In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1883.94 (1883), pp. 41 –73.

[4] E. Schmidt. "Zur Theorie der linearen und nichtlinearen Integralgleichungen". In: *Mathematische Annalen* 63.4 (Dec. 1907), pp. 433–476. DOI: 10.1007/BF01449770.

[5] N. N. Abdelmalek. "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. DOI: 10.1007/BF01939404.

[6] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization". In: *Mathematics of Computation* 30.136 (1976), pp. 772–795.

[7] B. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1980.

[8] A. Stathopoulos and K. Wu. "A Block Orthogonalization Procedure with Constant Synchronization Requirements". In: *SIAM Journal on Scientific Computing* 23.6 (2002), pp. 2165–2182. DOI: 10.1137/S1064827500370883.

[9] A. S. Householder. "Unitary Triangularization of a Nonsymmetric Matrix". In: *J. ACM* 5.4 (Oct. 1958), 339–342. DOI: 10.1145/320941.320947.

[10] Å. Björck. "Solving linear least squares problems by Gram-Schmidt orthogonalization". In: *BIT Numerical Mathematics* 7.1 (Mar. 1967), pp. 1–21. DOI: 10.1007/BF01934122.

[11] L. Giraud, J. Langou, M. Rozložník, and J. v. d. Eshof. "Rounding error analysis of the classical Gram-Schmidt orthogonalization process". In: *Numerische Mathematik* 101.1 (July 2005), pp. 87–100. DOI: 10.1007/s00211-005-0615-4.

[12] J. H. Wilkinson. "Modern Error Analysis". In: *SIAM Review* 13.4 (Oct. 1971), pp. 548–568. DOI: 10.1137/1013095.

[13] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317. DOI: 10.1137/090752286.

[14] I. V. Oseledets and E. E. Tyrtyshnikov. "Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions". In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3744–3759. DOI: 10.1137/090748330.

[15] S. V. Dolgov. "TT-GMRES: solution to a linear system in the structured tensor format". In: *Russian Journal of Numerical Analysis and Mathematical Modelling* 28.2 (2013), pp. 149–172. DOI: 10.1515/rnam-2013-0009.

[16] L. De Lathauwer, B. De Moor, and J. Vandewalle. “A Multilinear Singular Value Decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278. DOI: 10.1137/S0895479896305696.

[17] L. Grasedyck. “Hierarchical Singular Value Decomposition of Tensors”. In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2029–2054. DOI: 10.1137/090764189.

[18] I. V. Oseledets. “DMRG Approach to Fast Linear Algebra in the TT-Format”. In: *Computational Methods in Applied Mathematics* 11.3 (2011), pp. 382–393. DOI: 10.2478/cmam-2011-0021.

[19] P. Gelß. “The Tensor-Train Format and Its Applications”. PhD thesis. Freien Universitat Berlin, 2017.

[20] H. Al Daas, G. Ballard, P. Cazeaux, E. Hallman, A. Międlar, M. Pasha, T. W. Reid, and A. K. Saibaba. “Randomized Algorithms for Rounding in the Tensor-Train Format”. In: *SIAM Journal on Scientific Computing* 45.1 (2023), A74–A95. DOI: 10.1137/21M1451191.

[21] M. Che and Y. Wei. “Randomized algorithms for the approximations of Tucker and the tensor train decompositions”. In: *Advances in Computational Mathematics* 45.1 (2019), pp. 395–428. DOI: 10.1007/s10444-018-9622-8.

[22] W. Hoffmann. “Iterative algorithms for Gram-Schmidt orthogonalization”. In: *Computing* 41.4 (Dec. 1989), pp. 335–348. DOI: 10.1007/BF02241222.

[23] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, 1997.

[24] J. H. Wilkinson. *The algebraic eigenvalue problem.* en. Numerical Mathematics and Scientific Computation. Oxford, England: Clarendon Press, Jan. 1965.

[25] A. Smoktunowicz, J. L. Barlow, and J. Langou. “A note on the error analysis of classical Gram–Schmidt”. In: *Numerische Mathematik* 105.2 (Dec. 2006), pp. 299–313. DOI: 10.1007/s00211-006-0042-1.

[26] V. A. Kazeev and B. N. Khoromskij. “Low-Rank Explicit QTT Representation of the Laplace Operator and Its Inverse”. In: *SIAM Journal on Matrix Analysis and Applications* 33.3 (2012), pp. 742–758. DOI: 10.1137/100820479.

[27] E. Agullo, O. Coulaud, L. Giraud, M. Iannacito, G. Marait, and N. Schenkels. *The backward stable variants of GMRES in variable accuracy.* Research Report RR-9483. Inria, Sept. 2022, pp. 1–77.

[28] O. Coulaud, L. Giraud, and M. Iannacito. *A note on GMRES in TT-format.* Research Report RR-9484. Inria Bordeaux Sud-Ouest, 2022.

[29] M. Iannacito. “Numerical linear algebra and data analysis in large dimensions using tensor format”. PhD thesis. Inria Center at the University of Bordeaux, France, Dec. 2022.

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399