HAL open science

On some orthogonalization schemes in Tensor Train format

Olivier Coulaud, Luc Giraud, Martina Iannacito

To cite this version:

Olivier Coulaud, Luc Giraud, Martina Iannacito. On some orthogonalization schemes in Tensor Train format. RR-9491, Inria Bordeaux - Sud-Ouest. 2022. hal-03850387v2

HAL Id: hal-03850387

https://hal.science/hal-03850387v2

Submitted on 17 Nov 2022 (v2), last revised 15 Jan 2024 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

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 — November 2022 — 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 investigate numerically 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 tackle the curse of dimensionality, we represent tensor with low rank approximation using the Tensor Train (TT) formalism, and we introduce recompression steps in the standard algorithm outline through the TT-rounding method at a prescribed accuracy. After describing the algorithm structure and properties, we illustrate numerically that the theoretical bounds for the loss of orthogonality in the classical matrix computation round-off analysis results are maintained, with the unit round-off replaced by the TT-rounding accuracy. The computational analysis for each orthogonalization kernel in terms of the memory requirement and the computational complexity measured as a function of the number of TT-rounding, which happens to be the computational most 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

RESEARCH CENTRE BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

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 **4**

2 Notation and TT-format **5**

3 Orthogonalization schemes **6**
3.1 Classical and Modified Gram-Schmidt . . . 7
3.1.1 Classical schmes without reorthogonalization . . . 7
3.1.2 Classical schemes with reorthogonalization . . . 8
3.2 Gram approach . . . 9
3.3 Householder reflections . . . 11
3.4 Stability comparison . . . 15

4 Numerical tensor experiments **17**
4.1 Numerical loss of orthogonality . . . 17
4.2 Order-3 experiments . . . 17
4.3 Order-6 experiments . . . 19
4.4 Memory usage estimation . . . 20
4.4.1 Householder transformation . . . 21
4.4.2 Orthogonalization kernel comparison . . . 21
4.5 Summary . . . 25

5 Concluding remarks **26**

Distributed under a Creative Commons Attribution 4.0 International License

1 Introduction

Many numerical methods require to generate from a given set of m independent vectors of $\mathbb{R}^n$ an orthogonal basis of the subspace of spanned by these vectors. Different orthogonalization algorithms have been proposed during the years to perform this task. From another viewpoint, they allow also the computation of the matrix QR factorization. Indeed, if the input m vectors of $\mathbb{R}^n$ are organized as the columns of a matrix $A \in \mathbb{R}^{n\times m}$, then the orthogonalization schemes are able to 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) [9, 18], the Modified Gram-Schmidt (MGS) [9, 18], their variants with re-orthogonalization, named CGS2 and MGS2 [1, 5, 17], the Gram approach [20] and the Householder transformations [12]. CGS and MGS are algorithms realizing the Gram-Schmidt method whose fundamental idea is to sequentially remove from an input vector its projection along the previously computed orthonormal vectors and eventually normalizing it. The CGS2 and MGS2 procedures are meant to enhance the quality of the CGS and MGS basis vectors by orthogonalizing once more in the same way the basis computed with CGS and the MGS respectively. The Gram method computes the orthogonal basis taking advantage of the Cholesky factorization of the Gram matrix, defined by the scalar products of input vectors, while the Householder transformation relies on orthogonal reflections constructed from the input vector set and used to reflect the canonical basis.

A key feature in finite precision calculation for these orthogonalization algorithms is the *loss of orthogonality* of the computed basis. It expresses how much the computational rounding errors affect the computed basis orthogonality. This aspect has been deeply studied throughout the years, leading to many useful theoretical results, relating the loss of orthogonality to the linear dependency of the input vectors. The authors of [3, 8] establish some 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. The bounds proved in [8] for CGS2 and MGS2, confirms that this re-orthogonalization effectively improves the orthogonality of the computed basis. Bounds for the loss of orthogonality of the Householder transformation of the Gram method are proved in [22] and [20] respectively.

All the cited algorithms naturally translate in the tensor world, i.e., starting from a set of m tensors of $\mathbb{R}^{n_1\times\cdots\times n_d}$, they produce an orthogonal basis for the relative subspace of dimension m of $\mathbb{R}^{n_1\times\cdots\times n_d}$. Clearly, these orthogonalization schemes can work with dense tensors, but they are affected by the 'curse of dimensionality', i.e., their storage and operation cost grow exponentially with the tensor order. Thus, it is necessary to represent the tensor in compressed format. In this work, we generalize to tensors the six considered orthogonalization kernels relying on the Tensor Train (TT) formalism [14, 15]. However, the operation sequences among tensors in TT-formats reduce the benefit of this compressed representation. So, we introduce some additional compression steps through the `TT-rounding` function [14] in the orthogonalization schemes, knowing that they affect the orthogonality quality of the basis. The aim of this work is describing the orthogonalization kernels generalised to tensor with the TT-format and investigate numerically the loss of orthogonality of the computed basis. These numerical results in TT-format show some resemblance with the theoretical results of classical numerical matrix computation.

The rest of the paper is organized as follows. In Section 2, we present the notation and recall the main properties of the TT-format. Section 3 begins with the description of the six orthogonalization schemes extended to the tensor context through the TT-formalism. We address also the complexity in terms of numbers of `TT-rounding` application, representing the most computationally expensive operation. In Section 3.4 we recall briefly the known theoretical bounds related the loss of orthogonality of these schemes in classical matrix computation. The

theoretical results are linked with the numerical experiments, collected in Section 4, of the same orthogonalization schemes extended with the TT-format. The similarities between the classical orthgonalization kernels and their TT-versions are summarized in Section 5.

2 Notation and TT-format

For ease of reading, we adopt the following notations for the different mathematical objects involved in the description. Small Latin letters stand for scalars and vectors (e.g., a), leaving the context to clarify the object nature. Matrices are denoted by capital Latin letters (e.g., A), tensors by bold small Latin letters (e.g., $\mathbf{a}$), the sets by calligraphic capital letter (e.g., $\mathcal{A}$). We adopt the 'Matlab notation' denoting by " $:$ " all the indices along a mode. For example given a matrix $A \in \mathbb{R}^{m\times n}$, then $A(:,i)$ stands for the i-th column of A. The tensor product is denoted by $\otimes$, while the Euclidean dot product by $\langle\cdot,\cdot\rangle$ both for vectors and tensors, where it is generalized through the tensor contraction. We denote by $||\cdot||$ the Euclidean norm for vectors and the Frobenious norm for matrix and tensors. Given a matrix $A \in \mathbb{R}^{n\times n}$ its condition number 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 $k \in \{1,\ldots,d\}$. Since 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 $n = \max_{i\in\{1,\ldots,d\}}\{n_i\}$, different compression techniques were proposed over the years to reduce the memory consumption [6, 10, 16]. For the purpose of this work the most suitable tensor representation is the *Tensor Train* (TT) format [16]. The key idea of TT is expressing a tensor of order d as the contraction of d tensors of order 3. The contraction is actually the generalization to tensors of the matrix-vector product. Given $\mathbf{a} \in \mathbb{R}^{n_1\times\cdots\times n_h\times\cdots\times n_{d_1}}$ and $\mathbf{b} \in \mathbb{R}^{m_1\times\cdots\times n_h\times\cdots\times m_{d_2}}$, their *tensor contraction* with respect to mode h, denoted $\mathbf{a} \bullet_h \mathbf{b}$, provides a new tensor $\mathbf{c} \in \mathbb{R}^{n_1\times\cdots\times n_{h-1}\times n_{h+1}\times\cdots\times n_{d_1}\times m_1\times\cdots\times m_{h-1}\times m_{h+1}\times\cdots\times m_{d-1}}$ such that its $(i_1,\ldots,i_{h-1},i_{h+1},\ldots,i_{d_1},j_1\ldots,j_{h-1},j_{h+1},\ldots,j_{d_2})$ element is

$$\begin{aligned} c &= (\mathbf{a} \bullet_h \mathbf{b})(i_1,\ldots,i_{h-1},i_{h+1},\ldots,i_{d_1},j_1\ldots,j_{h-1},j_{h+1},\ldots,j_{d_2}) \\ &= \sum_{i_h=1}^{n_h} \mathbf{a}(i_1,\ldots,i_h,\ldots,i_{d_1})\mathbf{b}(j_1,\ldots,i_h,\ldots,j_{d_2}). \end{aligned}$$

The contraction between tensors is linearly extended to more modes. To shorten the notation we omit the bullet symbol and the mode indices when the modes to contract will be clear from the context.

The contraction applies also to the tensor representation of tensor linear operators for computing the operator powers. Let $\boldsymbol{\mathcal{A}} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{n_1\times\cdots\times n_d}$ be a linear operator and let the tensor $\mathbf{A} \in \mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ be its representation with respect to the canonical basis of $\mathbb{R}^{n_1\times\cdots\times n_d}$. Then the tensor representation with respect to this canonical basis of $\boldsymbol{\mathcal{A}}^2$ is $\mathbf{B} \in \mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$, whose element $b = \mathbf{B}(i_1,j_1,\ldots,i_d,j_d)$ is

$$\begin{aligned} b &= \big(\mathbf{A}\ {}_{h_\mathrm{L}}\bullet_{h_\mathrm{R}}\ \mathbf{A}\big)(i_1,j_1,\ldots,i_d,j_d) \\ &= \sum_{k_1,\ldots,k_d=1}^{n_1,\ldots,n_d} \mathbf{A}(i_1,k_1,\ldots,i_d,k_d)\mathbf{A}(k_1,j_1,\ldots,k_d,j_d) \end{aligned}$$

with $h_\mathrm{L} = \{2,4,\ldots,2d\}$ and $h_\mathrm{R} = \{1,3,\ldots,2d-1\}$. From this, we recursively obtain the tensor associated with $\boldsymbol{\mathcal{A}}^h$ for $h \in \mathbb{N}$.

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

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

where $\underline{\mathbf{x}}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$ is called k-th *TT-core* for $k \in \{1,\dots,d\}$, with $r_0 = r_d = 1$. Notice 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 the notation consistency we represent them as tensor. The k-th TT-core of a tensor are denoted by the same bold letter underlined with a subscript k. The value r_k is called *k-th TT-rank*. Thanks to the TT-formalism, the $(i_1,\dots,i_d)$-th element of $\mathbf{x}$ writes

$$\mathbf{x}(i_1,\dots,i_d) = \sum_{j_0,\dots,j_d=1}^{r_0,\dots,r_d} \underline{\mathbf{x}}_1(j_0,i_1,j_1)\underline{\mathbf{x}}_2(j_1,i_2,j_2)\dots\underline{\mathbf{x}}_{d-1}(j_{d-2},i_{d-1},j_{d-1})\underline{\mathbf{x}}_d(j_{d-1},i_d,j_d).$$

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., $\underline{X}_k(i_k) = \underline{\mathbf{x}}_k(:,i_k,:)$. Then each element of the TT-tensor $\mathbf{x}$ can be expressed as the product of d matrices, i.e.,

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

with $\underline{X}_k(i_k) \in \mathbb{R}^{r_{k-1}\times r_k}$ for every $i_k \in \{1,\dots,n_k\}$ and $k \in \{2,\dots,d-1\}$, while $\underline{X}_1(i_1) \in \mathbb{R}^{1\times r_1}$ and $\underline{X}_d(i_d) \in \mathbb{R}^{r_{d-1}\times 1}$. Remark that $\underline{X}_1(i_1)$ and $\underline{X}_d(i_d)$ are actually vectors, but as before to have an homogeneous notation they write as matrices with a single row or column.

Storing a tensor in TT-format requires $\mathcal{O}(dnr^2)$ units of memory with $n = \max_{i\in\{1,\dots,d\}}\{n_i\}$ and $r = \max_{i\in\{1,\dots,d\}}\{r_i\}$. In this case the memory footprint growths linearly with the tensor order and quadratically with the maximal TT-rank. Conseuqently, knowing the maximal TT-rank is usually sufficient to get an idea of the TT-compression benefit. However, to be more accurate, we introduce the compression ratio measure. Let $\mathbf{x} \in \mathbb{R}^{n_1\times\dots\times n_d}$ be a tensor in TT-format, then the compression ratio is the ratio between the storage cost of a in TT-format over the dense format storage cost, i.e.,

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

where r_i is the i-th TT-rank of $\mathbf{x}$. As highlighted from the compression ratio, to have a significant benefit in the use of this formalism, the TT-ranks r_i have to stay bounded and small. However, some operations among tensors in TT-format, as for example the algebraic sum, increase the TT-ranks. Indeed 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 [7]. So if $\mathbf{x}$ is a TT-tensor with k-th TT-rank r_k, then tensor $\mathbf{z} = 2\mathbf{x}$ has k-th TT-rank equal to r_k if we simply multiply the first TT-core by 2, but if it is computed as a sum of twice $\mathbf{x}$ then its TT-ranks double. To address the TT-rank growth, a rounding algorithm to reduce it was proposed in [16]. Given a TT-tensor $\mathbf{x}$ and a relative accuracy δ, the `TT-rounding` algorithm provides a TT-tensor $\tilde{\mathbf{x}}$ that is at a relative distance δ from $\mathbf{x}$, i.e., $||\mathbf{x}-\tilde{\mathbf{x}}|| \le \delta||\mathbf{x}||$. To estimate the benefit of the `TT-rounding`, we introduce the *compression gain*, the ratio of the compression ratios, that writes

$$\frac{\sum_{i=1}^{d} r_{i-1}n_ir_i}{\sum_{j=1}^{d} s_{j-1}n_js_j} \tag{2}$$

where r_i and s_i are the i-th TT-rank of $\mathbf{x} \in \mathbb{R}^{n_1\times\dots\times n_d}$ and $\tilde{\mathbf{x}} = \texttt{TT-rounding}(\mathbf{x},\delta)$. The computational cost, in terms of floating point operations, of a `TT-rounding` over $\mathbf{x}$ 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 [16].

3 Orthogonalization schemes

In the following sections, we describe the classical orthogonalization kernels, we propose their extensions to TT-vectors and we discuss their numerical stability comparing it with the theoretical results of classical matrix computation.

3.1 Classical and Modified Gram-Schmidt

In theoretical linear algebra the Gram-Schmidt process [9, 18] is a tool to produce an orthonormal basis from a given set of vectors. Let $\mathcal{A} = \{a_1, \ldots, a_m\}$ be a set of m linearly independent vectors of $\mathbb{R}^n$, then the key idea of the Gram-Schmidt process is to incrementally build an orthonormal basis of the space spanned by the elements of $\mathcal{A}$. At the i-th step, the i-th element a_i is made orthogonal to the previously computed $(i-1)$ orthonormal vectors $\{q_1, \ldots, q_{i-1}\}$, subtracting from a_i its projection along q_j, given by the scalar product of a_i and q_j for $j \in \{1, \ldots, i-1\}$. After normalization the new vector is q_i, the i-th vector of the final orthonormal basis. This mechanism is easily transported in the tensor framework. Consequently instead of stating the theory of Gram-Schmidt procedure with the tensor notation, we illustrate the two different realizations of this theoretical tool only in the TT-format. We carefully underline the differences with the classical matrix implementations.

3.1.1 Classical schmes without reorthogonalization

The direct implementation of the Gram-Schmidt process is known as *Classical Gram-Schmidt* (CGS) and its TT-version is sketched in Algorithm 1. Iteratively TT-CGS initializes $\mathbf{p}_i$ to $\mathbf{a}_i$ for every $i \in \{1, \ldots, m\}$, see line 3. Then 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 5 and 6. As last step, $\mathbf{p}_i$ is normalized and added in the new orthonormal basis $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_i\}$. The projections of $\mathbf{a}_i$ along $\mathbf{q}_j$ for $j \in \{1, \ldots, i-1\}$ define the i-th column of R, which is consequently upper triangular by construction. The i-th diagonal entry of R is the norm of $\mathbf{p}_i$ computed in line 9. These steps appear in both the tensor and in the matrix version of the Classical Gram-Schmidt algorithm. However when dealing with compressed format tensors, we must ensure that different algorithm steps do not reduce significantly the compression quality. Since we are dealing with low-rank TT-vectors, we have to ensure that the TT-ranks stay small. Indeed assuming that $\mathbf{p}_k$ and $\mathbf{q}_j$ have maximum TT-rank r for every $j \in \{1, \ldots, k-1\}$, then after $(k-1)$ subtractions, i.e., after $(k-1)$ repetitions of line 6, the TT-rank of $\mathbf{p}$ will be bounded by kr. To limit the TT-rank growth, we introduce in line 8 the compression of $\mathbf{p}_i$ through the `TT-rounding` algorithm with accuracy δ, which is the most computationally expensive operation in orthogonalization algorithms. Consequently the TT-CGS complexity depends on the number of `TT-rounding` calls and their complexity, which 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 orthogonalization methods studied, the TT-rank is not always known. Indeed, it arrives to round linear combinations of TT-vectors obtained from the `TT-rounding` algorithm with accuracy δ, whose TT-rank is not known a priori. Therefore here and after, we estimate the complexity of the orthogonalization algorithm in terms of rounding operation number. The complexity of TT-CGS is equal to m `TT-rounding` operations.

Algorithm 1 $\mathcal{Q}, R =$ TT-CGS($\mathcal{A}$, δ)

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p} = \mathbf{a}_i$
4: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
5: $\quad\quad R(i,j) = \langle \mathbf{a}_i, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
6: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
7: $\quad$ **end for**
8: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
9: $\quad R(i,i) = ||\mathbf{p}||$
10: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
11: **end for**
12: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

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

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p} = \mathbf{a}_i$
4: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}$ along $\mathbf{q}_j$
5: $\quad\quad R(i,j) = \langle \mathbf{p}, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{p}$ along $\mathbf{q}_j$
6: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
7: $\quad$ **end for**
8: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
9: $\quad R(i,i) = ||\mathbf{p}||$
10: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
11: **end for**
12: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

In the classical matrix framework Classical Gram-Schmidt is known for suffering from loss of orthogonality in the computed basis, as discussed later on, cf. [3]. The *Modified Gram-Schmidt* (MGS) algorithm introduces in the Classical Gram-Schmidt a tiny algorithmic change, which is sufficient to guarantee a generally better numerical orthogonality. Indeed, in TT-MGS we remove the projection along $\mathbf{q}_j$ of $\mathbf{p}_i$ and not of $\mathbf{a}_i$ for every $j \in \{1, \ldots, i-1\}$, as we see comparing line 5 of Algorithm 1 and 2 respectively. This modification reduces the error propagation, improving the general stability of the algorithm both in the classical matrix and in the tensor case, as we discuss in Section 3.4. For the remaining steps, the two algorithms are exactly 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-rounding` calls.

3.1.2 Classical schemes with reorthogonalization

CGS and MGS are know to have stability issues, described in details in Section 3.4, i.e., the close the input vectors are to linear dependency, the more the algorithms propagate the rounding errors, spoiling the final orthogonality of the new basis. As reported in [8], over the years, different articles, as for example [1, 5, 11], addressed this flaw introducing re-orthogonalization steps, that is, orthogonalizing repeatedly through the same approach the basis produced by the algorithm itself. In [8], the authors showed theoretically that one re-orthogonalization step is sufficient to improve significantly the orthogonality of the new basis generated by CGS and MGS. We present briefly the idea of CGS and MGS with re-orthogonalization, called CGS2 and MGS2, in the tensor case, highlighting those steps performed only in the TT-version.

In CGS2, sketched in Algorithm 3, as in CGS, the input TT-vector $\mathbf{a}_i$ is orthogonalized with respect to the $(i-1)$ previously computed orthogonal TT-vector $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$, 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. With these first $(i-1)$ iterations, given in line 6 of Algorithm 3, we define the TT-vector $\mathbf{p}_1$, which is then rounded in the TT-GCS2, see Algorithm 3 line 10. Up to this point, TT-CGS2 acts exactly as TT-CGS. However in TT-CGS2, after the rounding, the TT-vector $\mathbf{p}_1$ is orthogonalized again against $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$, defining $\mathbf{p}_2$. 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 completely defined, it is rounded and normalized, defining the i-th orthogonal TT-vector $\mathbf{q}_i$, as stated in line 12 and 13 of Algorithm 3. The norm of $\mathbf{p}_2$ at the i-th iteration defines the (i,i)-th diagonal component of R_2. The R factor from the QR decomposition computed by CGS2 is obtained summing R_1 and R_2.

The difference of MGS2 from CGS2 appears in line 7 of Algorithm 4 and 3 respectively. During the first orthogonalization loop, that is for $k = 1$ in line 4 of both methods, in the classical Gram-Schmidt version $\mathbf{a}_i$ is projected along $\mathbf{q}_j$ defining $\mathbf{p}_1$, while in the modified Gram-Schmidt one $\mathbf{p}_1$ is projected along $\mathbf{q}_j$ updating itself. In the second orthogonalization loop, i.e., when $k = 2$ in line 4 of both algorithms, in TT-CGS2 it is removed from $\mathbf{p}_1$ its projection along $\mathbf{q}_j$ defining $\mathbf{p}_2$, while in the TT-MGS2 version, it is $\mathbf{p}_2$ the TT-vector projected along $\mathbf{q}_j$ and updated. All the remaining steps, including the `TT-rounding` and the construction of R_1, R_2 and their sum R, are identical between TT-CGS2 and TT-MGS2. Remark that the rounding steps are performed only in the TT-version of MGS2 and CGS2.

Algorithm 3 $\mathcal{Q}, R =$ TT-CGS2($\mathcal{A}$, δ)

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

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

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

On the basis of the previously used hypothesis, the computational complexity of TT-CGS2 and TT-MGS2 is estimated as $2m$ `TT-rounding` operations, since during the m iterations we have two temporary TT-vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ rounded.

3.2 Gram approach

In [20], the authors propose an algorithm, that we refer to as Gram's algorithm, to generate orthogonal basis starting from a set m linearly independent vectors of $\mathbb{R}^n$ with $m \ll n$. This scheme is based on the Gram matrix, which under the hypothesis $m \ll n$ is significantly small. The key idea is decomposing the small Gram matrix through its Cholesky factorization and benefiting from 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 details the implementation of the Gram algorithm in the TT-format. Indeed the tensor realization of this procedure is extremely close to the matrix one, so that describing the tensor case and its differences with the classical matrix one is sufficient to ensure a good comprehension.

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

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

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

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

Comparing Equation (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 writes 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 strictly speaking is 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. *Notice that by construction the condition number of G is the square of the one of A. Consequently if the condition number of A associated with the set of input vectors $\mathcal{A}$ is larger than the inverse of the squared root of the working precision of the considered arithmetic, e.g., $u_{64} \approx 10^{-16}$ for 64-bit calculation, the associated Gram matrix G is numerically singular and its Cholesky decomposition is no longer defined. This constitutes the main practical drawback of this method.*

This procedure produces 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 $G \in \mathbb{R}^{m\times m}$ through the tensor scalar product and we compute its Cholesky factorization getting 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 are obtained from the elements of $\mathcal{A}$ by back-substitutions involving the R factor. However this approach cannot be easily translated into the tensor framework, where the inverse of R has to be explicitly computed.*

As for the other orthogonalization techniques, in the tensor case, we prevent memory issues, monitoring the TT-ranks. 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 12 has maximum TT-rank bounded by ir. Since this value grows linearly with m, in line 14 we introduce a rounding step at prescribed accuracy δ. As in Sections 3.1 and 3.2, notice that the complexity of the TT-Gram algorithm is given by m `TT-rounding` operations. However, in this particular case, we can even estimate the cost of each single rounding step and consequently 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, that is the number of floating point operations, of each rounding operation, the most expensive step in the entire algorithm, is known and is equal $\mathcal{O}(dni^3r^3)$; summing up over $i \in \{1, \dots, m\}$, we conclude that the TT-Gram algorithm cost is $\mathcal{O}(dnm^4r)$ floating point operations.

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

input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
for $i = 1, \dots, m$ **do**
  for $j = 1, \dots, i$ **do**
  ▷ construct the Gram matrix through the inner product of the input TT-vectors
    $G(i,j) = G(j,i) = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$
  end for
end for
$L = \texttt{cholesky}(G)$ ▷ compute the Cholesky factorization
$R = L^\top$ and $R^{-1} = \texttt{invert}(R)$ ▷ define the R factor of the QR-factorization
for $i = 1, \dots, m$ **do**
  $\mathbf{p} = R^{-1}(i,1)\mathbf{a}_1$
  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
    $\mathbf{p} = \mathbf{p} + R^{-1}(i,j)\mathbf{a}_j$
  end for
  $\mathbf{q}_i = \texttt{TT-round}(\mathbf{p}, \delta)$ ▷ round the TT-vector before adding it to the basis
end for
return: $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

3.3 Householder reflections

In the classical matrix framework, the Householder transformations are widely applied to generate orthogonal basis, thanks to their remarkable stability properties, illustrated in the following sections. We briefly present the theoretical construction of an Householder transformation, underlining how to generate an orthogonal basis. The second half of this section describes in details how we extend the Householder transformation to the tensor context, with a specific attention to the implementation of the Householder orthogonalization scheme in TT-format.

Given a vector $x \in \mathbb{R}^n$, the Householder reflector moves x along a chosen direction, which usually is an element of the canonical basis or a linear combination of them. We illustrate the construction of the Householder reflector in the general case. 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 matrix H representing the Householder reflection is unitary and entirely defined by the Householder vector u. Moreover the action of an Householder reflector is computed by one scalar product with the Householder vector u and one algebraic vector summation. Indeed given a vector $z \in \mathbb{R}^n$ and an Householder reflector $H = \mathbb{I}_n - 2u \otimes u$, the image of w through H is

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

In particular, if u is defined as in Equation (5), then we explicitly verify that $Hx = \|x\| y$. Firstly

remark 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. Thanks to this result and Equation (6), we get

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

The Householder transformations are generally used to compute the QR factorization of a matrix, but they find application also in situations where a set of vectors has to be converted into an orthogonal basis. We examine briefly the two possibilities. Given a matrix $A \in \mathbb{R}^{n\times m}$, we construct m Householder reflections such that the k-th one moves the k-th column of A along a linear combination of the first k canonical basis vectors. Said differently, the k-th Householder transformation sets to zero the last $(n-k)$ entries of the k-th column of A. Consequently, after m Householder reflections the matrix A ends up being upper triangular. We illustrate more in details how the algorithm iteratively proceeds. Let $a_1 \in \mathbb{R}^n$ be the first column of A, the first step consists in reflecting it along the first canonical basis vector e_1, i.e., in constructing the Householder reflector H_1 such that $H_1 a_1 = \|a_1\|e_1$. Consequently the first Householder vector $u_1 \in \mathbb{R}^n$ is

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

and normalized. For stability reasons, cf. [21], the sign of the norm of a_1 is determined by the sign of the first component of a_1, that is a plus if $a_1(1) > 0$, a minus otherwise. The first Householder reflector H_1 is applied to all the columns of A, that is $\tilde{a}_j = H_1 a_j$ for $j \in \{1,\dots,m\}$. Henceforth we denote by $\tilde{a}_j$ the j-th column of A updated by all the previously defined $(j-1)$ Householder transformations for $j \in \{2,\dots,m\}$. Notice that the first Householder transformation moves the first column of A along a multiple of the first canonical basis vector e_1, i.e., it sets the last $(n-1)$ entries of the first column of A to zero. Then we proceed reflecting the second column of A, updated by H_1 along a linear combination of the first two canonical basis vectors e_1 and e_2. Let $u_2 \in \mathbb{R}^n$ be the Householder vector defining the second Householder reflector H_2, which updates a second time $\tilde{a}_j$ the j-th column of A for $j \in \{2,\dots,m\}$. Thus, at this point $\tilde{a}_2$ only has its first two components non null. 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 $\sqrt{\sum_{\ell=1}^k \alpha_\ell^2} = ||\tilde{a}_k||$. Before normalization, the k-th Householder vector is $u_k \in \mathbb{R}^n$ such that

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

where for every $\ell \in \{1, \dots, k-1\}$ we have

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

Again for stability reasons, cf. [21], β_k is positive if $\tilde{a}_k(k)$ is positive, negative otherwise. Then u_k is normalized.

Remark 3.3. *By construction the first $(k-1)$ entries of u_k are zeros, the last $(n-k)$ ones are equal to corresponding ones of a_k. The k-th component of u_k is given by the difference of the k-th component of a_k and the quantity β_k, that is $u_k(k) = \tilde{a}_k(k) - \beta_k$. Said differently, only the last $(n-k+1)$ entries of $\tilde{a}_k$ have a determinant role. Thanks to this property, in the matrix context, the Householder QR factorization has a reduced and simplified construction. The k-th Householder vector is $\hat{u}_k \in \mathbb{R}^{n-k+1}$ defined from the norm of $\tilde{a}_k$ with the first $(k-1)$ entries being zeros, i.e.,*

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

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 from $\hat{u}_k$ and is applied only on the last $(n-k+1)$ components of $\tilde{a}_j$ for $j \in \{k, \dots, m\}$ and $k \in \{1, \dots, m\}$. This reduced approach is not reproducible in the tensor framework, where the object is expressed in compressed format. Therefore we present it even in the matrix case in the most general way, that is not the one adopted to make an implementation in matrix computation.

Once the k-th Householder transformation has been applied to $\tilde{a}_j$ for $j \in \{k, \dots, m\}$, the vector $\tilde{a}_k$ has its last $(n-k)$ component equal to zero. The application of m Householder reflections to A leads to the upper triangular matrix R, i.e., the R factor of the QR decomposition. To compute the Q factor, we multiply the m Householder reflection matrices. For the majority of the applications, forming the Householder vectors and knowing the Householder transformations implicitly, as given in Equation (6), is sufficient. However, if we want to produce an orthogonal basis out of a generic set of m vectors $\mathcal{A} = \{a_1, \dots, a_m\}$ with $a_k \in \mathbb{R}^n$, a further step has to be considered. As for computing the QR factorization, H_k the k-th Householder transformation is defined by the k-th Householder vector u_k, 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 H_k the first k Householder transformations to the k-th canonical basis vector e_k in reverse order, i.e.,

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

As in the previous sections, we extend with some modifications this approach to the tensor case. 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, fixed $\mathcal{N}_i = \{1, \dots, n_i\}$ for every $i \in \{1, \dots, d\}$ and $n = \prod_{j=1}^{d} n_j$, let 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 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 we fix for the subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ is $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_m\}$ 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 $k \in \{1, \dots, d\}$. Henceforth we use index i to denote the i-th element of the canonical basis, implying $i = \psi(i_1, \dots, i_d)$.

As stated in Remark 3.3, the first $(k-1)$ components of u_k are zeros, the last $(n-k)$ components are equal to corresponding ones of $\tilde{a}_k$, while the k-th one is given by the difference of the k-th component of $\tilde{a}_k$ and the quantity β_k, defined in Equation (7). We want to transport this structure in the tensor case. However if the element of $\mathcal{A}$ are in TT-format, it is not possible to directly access to the tensor components and we need to recover them, either by multiplying the TT-cores with the correct index or by computing the scalar product with the element of $\mathcal{E}$. Let $\tilde{\mathbf{a}}_k$ be the result of $(k-1)$ Householder reflections applied to $\mathbf{a}_k$, 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 $R(j,k) = \langle \tilde{\mathbf{a}}_k, \mathbf{e}_j \rangle$ 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 8 and 10 of Algorithm 6 respectively. The k-th component of r_k takes positive sign if $\langle \tilde{\mathbf{a}}_k, \mathbf{e}_k \rangle > 0$, otherwise it takes negative sign, extending in the tensor framework the stability preserving idea given in [21]. The j-th component of r_k corresponds to the (j,k) component of the R factor. As previously pointed out, we have 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 $\tilde{\mathbf{a}}_k$, i.e., after line 8, the TT-rank of $\mathbf{u}_k$ is bounded by $(r_a + k - 1)$. We introduce at this step the first `TT-rounding` call on the Householder TT-vector $\mathbf{u}_k$, whose TT-rank is decreased depending on the accuracy value δ. Then the k-th component of r_k is subtracted, determining a further growth in the TT-rank of $\mathbf{u}_k$. Since $\mathbf{u}_k$ plays a key role in the Householder transformation and consequently its TT-rank has a strong impact on the entire process, we decide to perform a further `TT-rounding` step over $\mathbf{u}_k$ at accuracy δ.

After describing the construction of an Householder TT-vector, which is summarized in Algorithm 6, the focus is on the process that generates an 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 with Algorithm 6 the k-th Householder TT-vector $\mathbf{u}_k$, which defines implicitly the Householder transformation $\mathbf{H}_k$. We store the first k components of r_k in the k-th column of the upper triangular matrix $R \in \mathbb{R}^{m \times m}$. Then $\mathbf{H}_k$ is implicitly applied using $\mathbf{u}_k$ to form $\tilde{\mathbf{a}}_j$ for every $j \in \{k, \dots, m\}$, as expressed in line 7 of Algorithm 8, following the same approach as in matrix case, i.e.,

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

Algorithm 7 applies a given Householder reflection to a specific input vector. Remark that $(k-1)$ transformations are performed on $\mathbf{a}_k$, computing $\tilde{\mathbf{a}}_k$, before generating the k-th reflector, leading the maximum TT-rank of $\mathbf{a}_k$ to be potentially much larger that its initial value. Therefore to keep the TT-rank of $\tilde{\mathbf{a}}_k$ reasonably small, we perform a `TT-rounding` before generating the associated Householder TT-vector, see line 10.

The conclusive part of the TT-Householder transformation algorithm 8 generates the new set $\mathcal{Q}$ of orthonormal TT-vectors. Let $\mathbf{q}_i$ be the i-th element of $\mathcal{Q}$ obtained applying the first

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

input: $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $\mathcal{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_i\}$, $\delta \in \mathbb{R}_+$
$s = 0$
$\mathbf{w} = \mathbf{a}$
for $j = 1, \ldots, i-1$ **do**
▷ compute the component of $\mathbf{a}$ along the j-th canonical basis TT-vector
  $r(j) = \langle \mathbf{a}, \mathbf{f}_j \rangle$
  $s = s + \big(r(j)\big)^2$
▷ set to zero the component of $\mathbf{a}$ along the j-th canonical basis TT-vector $\mathbf{f}_j$
  $\mathbf{w} = \mathbf{w} - r(j)\mathbf{f}_j$
end for
$\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
▷ subtract from the norm of $\mathbf{a}$ the contribution of the components set to zero
$r(i) = \text{sign}(\langle \mathbf{a}, \mathbf{f}_i \rangle)\sqrt{||\mathbf{a}||^2 - s}$
$\mathbf{w} = \mathbf{w} - r(i)\mathbf{f}_i$
$\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
$\mathbf{u} = (1/||\mathbf{z}||)\mathbf{w}$
return: $\mathbf{u}$, r

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

input: $\mathbf{a}, \mathbf{u} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
$\mathbf{b} = \mathbf{a} - 2\langle \mathbf{a}, \mathbf{u} \rangle \mathbf{u}$ ▷ apply the Householder reflection defined by $\mathbf{u}$ to $\mathbf{a}$
return: $\mathbf{b}$

i Householder reflections in reverse order to $\mathbf{e}_i$ from the canonical basis $\mathcal{E}$, see line 16 of Algorithm 8. To keep the maximum TT-rank of $\mathbf{q}_i$ limited and to avoid running out of memory, each $\mathbf{q}_i$ is rounded at accuracy δ, as stated in line 18 of Algorithm 8.

From the complexity viewpoint, the TT-Householder algorithm costs $4m$ `TT-rounding` operations: 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. Consequently, the TT-Householder algorithm is more expensive than all the other orthogonalization methods, that is 4 times more expensive than CGS and MGS, twice more than 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 $\mathcal{A}_m = \{a_1, \ldots, a_m\}$ is defined by the L2-norm of the difference among the identity matrix of size m and the Gram matrix defined from the m vectors generated by the orthogonalisation algorithm. We state more formally the definition. Let $\mathcal{Q}_m = \{q_m, \ldots, q_m\}$ be a set of m vectors, obtained from an orthogonalisation 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 $Q_m^\top Q_m$. Remark that the (i,j) element of $Q_m^\top Q_m$ is the scalar product of q_i and q_j, that is $Q_m^\top Q_m(i,j) = \langle q_i, q_j \rangle$ 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

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

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

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, δ
let $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ be the canonical basis of a subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$
$\mathbf{w} = \mathbf{a}_1$
for $i = 1, \ldots, m$ **do**
▷ construct the i-th Householder TT-vector
  $\mathbf{u}_i$, $R(: i, i) = \texttt{TTH-vec}(\mathbf{w}, \mathcal{F}_i, \delta)$ with $\mathcal{F}_i = \{\mathbf{e}_1, \ldots, \mathbf{e}_i\}$
  for $j = i, \ldots, m$ **do**
    $\mathbf{a}_j = \texttt{apply-H-vec}(\mathbf{a}_j, \mathbf{u}_i)$ ▷ update the j-th element of $\mathcal{A}$
  end for
  if i $<$ m **then**
▷ round the TT-vector that will define the successive Householder reflection
    $\mathbf{w} = \texttt{TT-round}(\mathbf{a}_{i+1}, \delta)$
  end if
end for
for $i = 1, \ldots, m$ **do** ▷ compute the new orthogonal basis
  $\mathbf{q}_i = \mathbf{e}_i$
  for $j = i, \ldots, 1$ **do**
    $\mathbf{q}_i = \texttt{apply-H-vec}(\mathbf{q}_i, \mathbf{u}_j)$ for ▷ reflect the i-th element of $\mathcal{E}$
  end for
  $\mathbf{q}_i = \texttt{TT-round}(\mathbf{q}_i, \delta)$
end for
return: $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

In the classical matrix framework, an orthogonalization scheme is said *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 holding true for the six orthogonalization schemes in classical linear algebra constitute a base line for the comparison with the numerical results obtained in the tensor framework, discussed in Section 4.

In [8, Theorem 1] the authors prove that given $\mathcal{A}_m = \{a_1, \ldots, a_m\}$ a set of m vectors, the loss of orthogonality for a basis obtained by CGS 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 [3], it is provided an upper bound for the loss of orthogonality of MGS. Indeed 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 [20, Theorem 4.1] who proposed the Gram orthogonalization scheme, estimated also an upper bound for the loss of orthogonality of their orthogonalization technique. More precisely, 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). Lastly the Householder orthogonalization algorithm is known for being the most stable one. Indeed 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 proved in [23]. When a further orthogonalization step is introduced in the classical and modified Gram-Schmidt, defining CGS2 and MGS2, their loss of orthogonality improves considerably, reaching Householder quality. Indeed, as proved in [8, 19], 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 $\kappa(A_m)u \ll 1$. A summary of all the loss of orthogonality bounds is presented in Table 1.

4 Numerical tensor experiments

In Sections 3.1 - 3.3, we describe four orthogonalization methods which produce an orthonormal basis of TT-vectors, given a set of TT-vectors and a rounding accuracy δ. This section analyses 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 with a Krylov process. Let $\mathbf{x}_1 \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be the TT-vector of ones, iteratively we 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 [13], and $\mathbf{a}_j$ is the normalized output of the `TT-round` algorithm applied to $\mathbf{x}_j$. Consequently, $\mathbf{a}_j$ the j-th element of $\mathcal{A}_m$ has TT-rank 1 for every $j \in \{1, \ldots, m\}$. This TT-rank constraint facilitates the analysis of the memory requirement. By construction, the elements of $\mathcal{A}$ are generated as a sequence of m normalized (and rounded) Krylov TT-vectors, so that when we vectorize and arrange only the first k as columns of the matrix A_k, its condition number $\kappa(A_k)$ grows for $k \in \{1, \ldots, m\}$. Henceforth $\mathcal{A}_k$ denotes the subset of $\mathcal{A}_m$ defined by its first k TT-vectors. The two experiments differ in the dimension of the problem, 3 for the first one and 6 for the second one, but they have the same mode size $n = 15$. We provide further details in the following sections.

4.1 Numerical loss of orthogonality

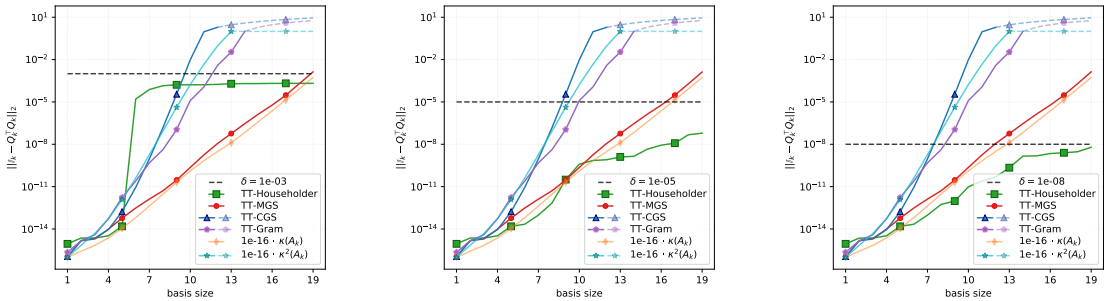
In this section, we examine the numerical results from two set of experiments from the viewpoint of the loss of orthogonality. The purpose is highlighting the similarities with the classical matrix orthogonalization methods. In particular, we investigate the loss of orthogonality $\|\mathbb{I}_k - Q_k^\top Q_k\|_2$, where the (i, j) element of $Q_k^\top Q_k$ is computed by the scalar 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 $\mathcal{Q}$is the set of TT-vectors produced by the considered orthogonalization kernel.

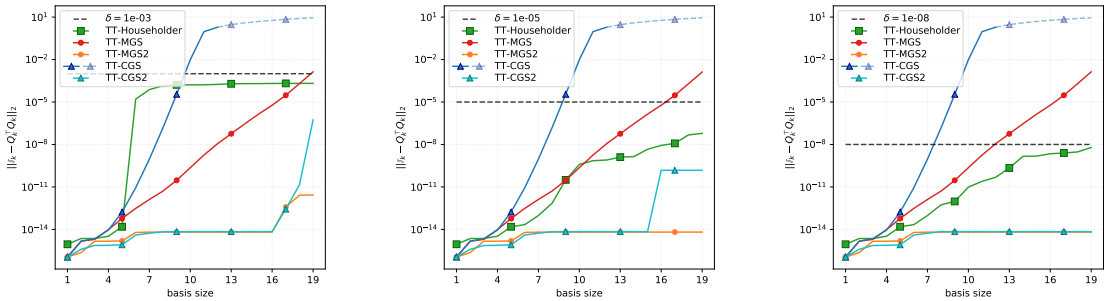
4.2 Order-3 experiments

For 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 $m = 20$. The results of this first group of experiments for the six different schemes and three different rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$ are reported in Figure 1. The choice of a low dimensional problem enables us to convert in dense format and vectorize each TT-vector $\mathbf{a}_j$, storing it as the j-th column of $A_m \in \mathbb{R}^{n^3 \times m}$. As consequence, we can estimate the condition number of $A_k \in \mathbb{R}^{n^3 \times k}$, i.e., the submatrix of A_m formed by its first k columns. In all the plots of Figure 1, together with the loss of orthogonality with continuous coloured curves, we display the constant rounding accuracy δ with a dashed black line, with coloured continuous lines the condition number $\kappa(A_k)$ and its squared value



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

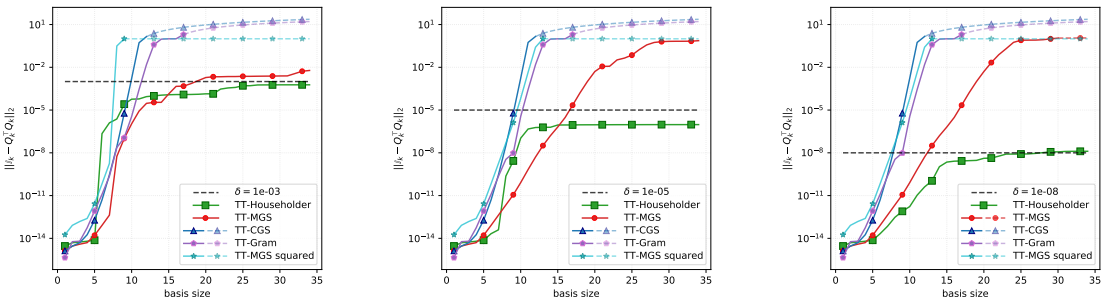
Loss of orthogonality with reorthogonalization comparison

Figure 1: LOO 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.

scaled by $u \approx 10^{-16}$, as long as they are smaller than 1. We scale the condition number curves for the ease of comparison with the slope of the loss of orthogonality of TT-MGS and TT-CGS. All the curves are in function of the basis size k. Indeed, as previously stated, since the TT-vectors are generated as a sequence of normalized Krylov TT-vectors, for increasing values of k, the elements of $\mathcal{A}_k$ get more linearly dependent, leading the associated condition number $\kappa(A_k)$ to increase. For the interpretation ease, we present in the first line three plots, that are Figure 1a, 1b and 1c, with the loss of orthogonality of standard methods in tensor format, i.e., TT-Householder, TT-MGS, TT-CGS, TT-Gram; on the second line, Figure 1d, 1e and 1f display the loss of orthogonality of the methods with re-orthogonalization, that are TT-MGS2 and TT-CGS2, compared with the result of TT-Householder, TT-MGS and TT-CGS. In all the six plots of Figure 1, we observe similar behaviours. The TT-Householder loss of orthogonality in green stagnates around the rounding accuracy δ; this phenomenon is extremely clear 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-rounding` accuracy δ. The TT-MGS loss of orthogonality in red grows with the same slope of 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, following the behaviour

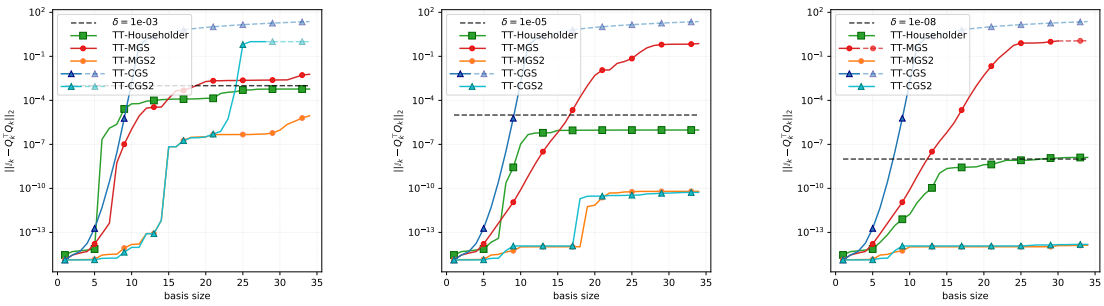
of the squared condition number $\kappa^2(A_k)$, as long as $\kappa^2(A_k) < 10^2$. Indeed TT-CGS and TT-Gram loss of orthogonality curves stagnates below 10^2 for $k > 10$ approximately, while $\kappa^2(A_k)$ continues to grow. Looking at the second line plots, Figure 1d, 1e and 1f show that introducing a second re-orthogonalization loop improves considerably the loss of orthogonality. Indeed as long as $k < 15$ the loss of orthogonality of TT-CGS2 is close to the machine precision, around 10^{-14} for $\delta \in \{10^{-3}, 10^{-5}\}$, increasing after; for $\delta = 10^{-8}$ it remains around 10^{-14}. Consequently as long as the element of $\mathcal{A}_k$ are not too much collinear, TT-CGS2 outcompetes TT-CGS, TT-MGS and TT-Householder, but not TT-MGS2. For the rounding accuracy $\delta \in \{10^{-5}, 10^{-8}\}$, the loss of orthogonalization of TT-MGS2 stagnates around 10^{-14}, while for $\delta = 10^{-3}$ the loss of orthogonality jumps from 10^{-14} to 10^{-11}, where it seems to stagnates, when $k > 16$. Overall TT-MGS2 outperforms all the other algorithms. These results found for TT-CGS2 and TT-MGS2 are consistent with the matrix theory presented in Section 3.4. Moreover, we may hypothesise that the jumps arrive when the condition number $\kappa(A_k)$, or its square, times the rounding accuracy is not any more 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 reorthogonalization comparison

Figure 2: LOO 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 the applicability to large-scale problems, we consider a second experimental framework. Set the problem order to $d = 6$ with size mode $n_i = 15$ for $i \in \{1, \dots, 6\}$, we 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. The loss of orthogonality of these experiments is displayed in Figure 2. Because of the problem order $d = 6$ and size $n = 15$, in this case we do not add the curve of the condition number of the $A_k \in \mathbb{R}^{n^6 \times k}$, whose k-th column would be the TT-vector $\mathbf{a}_k \in \mathcal{A}_m$ uncompressed and vectorized. To compensate the absence of the condition number curve, we display the square of the TT-MGS loss of orthogonality values with a dashed line, that should exhibit the same slop 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. As in the previous case, the first line plots display standard orthogonalization algorithms in TT-format, while on the second line we display results from methods with re-orthogonalization. Figures 2a, 2b and 2c clearly show that the TT-Householder orthogonalization algorithm produces a basis whose loss of orthogonality stagnates around the rounding accuracy δ for every value in $\{10^{-3}, 10^{-5}, 10^{-8}\}$ after the basis size gets greater than approximately 10. This seems to support further our intuition that even in the tensor framework the bound expressed in Equation (12) is still holding true, with the unit round-off u replaced by the `TT-rounding` accuracy δ. In Figure 2a and 2b, when the basis size is smaller than about 15, the TT-MGS loss of orthogonality is smaller than the Householder one, but when the basis includes more then 15 TT-vectors, the relation reverses. For more accurate computation, that is for $\delta = 10^{-8}$, already when the basis size is around 5, the Householder loss of orthogonality outcompetes the TT-MGS one. Notice that for all the rounding accuracy values, the TT-MGS loss of orthogonality firstly grows linearly and stagnates after a while at different level, at 10^{-2} for $\delta = 10^{-3}$, at 1 for $\delta \in \{10^{-5}, 10^{-8}\}$. The stagnation of the TT-CGS and TT-Gram loss of orthogonality curves arrives almost immediately, when the basis size is greater than 10 for all the rounding accuracy values, i.e., in all the plots of Figure 2. This is probably due to the very bad condition number of the basis, so that the assumptions of the matrix theory are not longer valid, that is, condition number times rounding smaller than 1. Moreover mainly Figures 2b and 2c show that the TT-CGS and TT-Gram loss of orthogonality follow the square of the TT-MGS loss of orthogonality, supporting the idea that even in the TT-format, TT-MGS loss of orthogonality grows as the condition number, while TT-Gram and TT-CGS loss of orthogonality as the squared condition number. Figures 2d, 2e and 2f display the loss of orthogonality of TT-MGS2 and TT-CGS2, compared with the previously analysed results of TT-Householder, TT-MGS and TT-CGS. For all the considered rounding accuracies, TT-MGS2 outperforms all the other methods, with a loss of orthogonality which stagnates around 10^{-5} for $\delta = 10^{-3}$, around 10^{-10} for $\delta = 10^{-5}$ and around 10^{-13} for $\delta = 10^{-8}$. In all the three Figures 2d- 2f, the TT-MGS2 curve presents a jump larger for greater values of δ, for $15 \leq k \leq 20$ with $\delta \in \{10^{-3}, 10^{-5}\}$ and for $k \sim 10$ with $\delta = 10^{-8}$. Also TT-CGS2 outcompetes the TT-Householder, TT-CGS and TT-MGS, following the behaviour of TT-MGS2, always for $\delta \in \{10^{-5}, 10^{-8}\}$ and as long as the TT-vectors are not too much collinear, i.e., for $k < 20$ when $\delta = 10^{-3}$. These results support the conclusion stated for the $d = 3$ experiments, that the bounds for the loss of orthogonality of TT-CGS2 and TT-MGS2 proved in the classical matrix framework still apply, probably under revised hypothesis.

4.4 Memory usage estimation

This section aims to study the effect on the TT-rank and consequently on the memory requirement of the orthogonalization process from the numerical results. We investigate the growth of

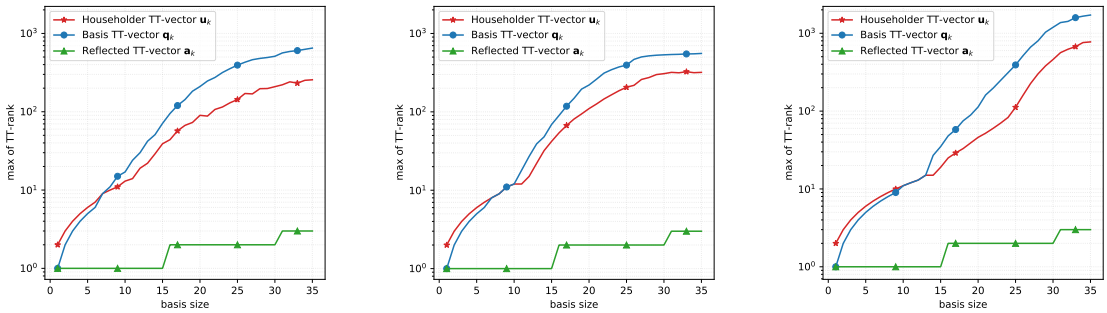
the TT-ranks, of the compression ratio, defined in Equation (1), and the compression gain curve, stated in Equation (2). We examine the TT-rank, the compression ratio and the compression gain of the new basis produced by the orthogonalization algorithms in the second set of experiments with $d = 6$, $n_i = 15$, $m = 35$ and $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, since this problem can already be considered a large scale one.

4.4.1 Householder transformation

In the Householder algorithm 8 there are three sets of TT-vectors on which the `TT-rounding` is applied: the Householder TT-vector $\mathbf{u}_k$, the TT-vector $\mathbf{a}_k$ to which we apply k Householder transformations and the orthogonal TT-vector $\mathbf{q}_k$ obtained from the canonical basis TT-vectors with i successive Householder transformations. It is worthwhile studying the evolution of the maximum TT-rank, of the compression ratio and gain for each of these three TT-vector groups. In Figure 3 we display the maximum TT-rank, the compression ratio and the compression gain of $\mathbf{u}_k$, $\mathbf{a}_k$ after the k-th reflection and $\mathbf{q}_k$ for every $k \in \{1, \ldots, 35\}$ for all the values of the rounding accuracy δ. As expected, the maximum TT-rank and the compression ratio of $\mathbf{u}_k$, $\mathbf{a}_k$ and $\mathbf{q}_k$ grow for increasing basis sizes, because the number of terms in their computation grows with k. Remark that the maximum TT-rank of $\mathbf{q}_k$ becomes larger than one of $\mathbf{u}_k$ for a basis size greater than 10. This property is significant since in the majority of the 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.
In Figures 3a, 3b and 3c we observe that the maximum TT-rank of $\mathbf{a}_k$ after the k-th Householder reflection is extremely low, mainly if it is compared with those of $\mathbf{q}_k$ and $\mathbf{u}_k$. Moreover, the rise of 1 of the maximum TT-rank of $\mathbf{a}_k$ arrives every time the index k is equal to a multiple of the mode size $n = 15$, as we can see in Figure 3a and 3b. This behaviour is present also for $\delta = 10^{-8}$ for Figure 3c, but it is less regular. We tried to investigate further this phenomenon from a theoretical viewpoint, but we did not arrive to a clear and convincing explanation.
The compression ratio has closely the same shape as maximum TT-rank, but it enables us to observe the memory growth in percentage. Figures 3d, 3e and 3f present for all the values of the rounding accuracy δ a compression ratio smaller than 1, implying that for all these experiments the TT-format is still relevant to reduce the memory footprint. Moreover the compression ratio of $\mathbf{a}_k$ stagnates around 10^{-5} for all the three values of δ. Figures 3d and 3e show that the compression ratio of $\mathbf{q}_k$ stagnates around 10^{-1}, while the one of $\mathbf{u}_k$ around 10^{-2}. In Figure 3f, it is not clear if the compression ratio of $\mathbf{q}_k$ and $\mathbf{u}_k$ stagnate around 1 and 10^{-1} respectively. All the Figures 3g, 3i and 3h, the gain curves of $\mathbf{u}_k$, $\mathbf{q}_k$ and $\mathbf{a}_k$ present approximately the same behaviour. In particular 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 is almost constant through the iterations and minimal, but both the TT-vectors are compressed several times during the previous iterations. On the other side, the compression gain for storing the Householder basis TT-vector $\mathbf{q}_k$ is significantly larger. For all the rounding accuracies, the compression gain curve of $\mathbf{q}_k$ increases during the first 10 iterations, decreases slightly after and seems to grow again during the last 10 iterations. Remark that for the compression gain curve of the Householder basis vector the highest value is around 110 if $\delta = 10^{-5}$. This means that after the compression, it is actually requested slightly less than 1% of the memory used to store the same tensor in TT-format before the compression.

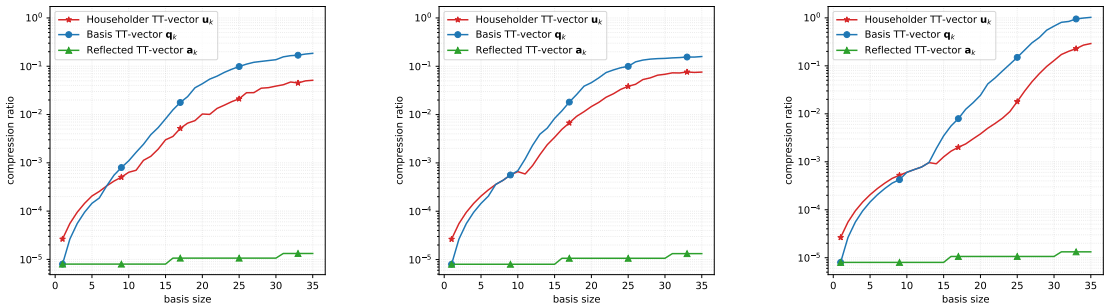
4.4.2 Orthogonalization kernel comparison

After describing the memory need of the TT-Householder algorithm, we compare the four orthogonalization schemes from the memory requirement viewpoint. Indeed, to fairly compare the orthogonalization schemes, we have to consider the memory consumptions together with the



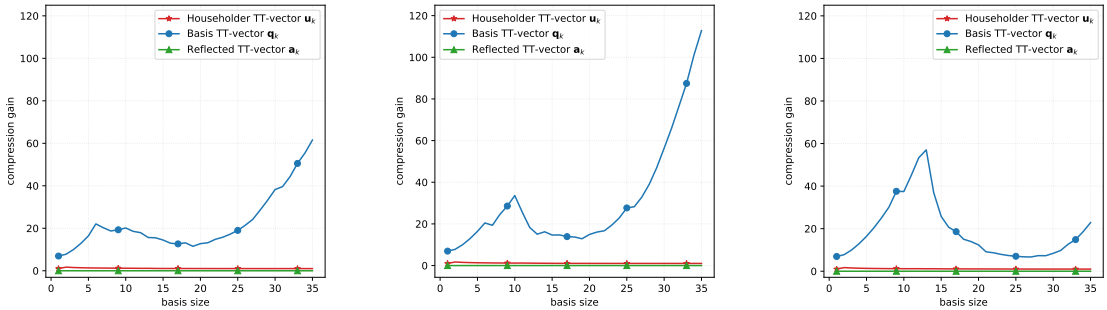
(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

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



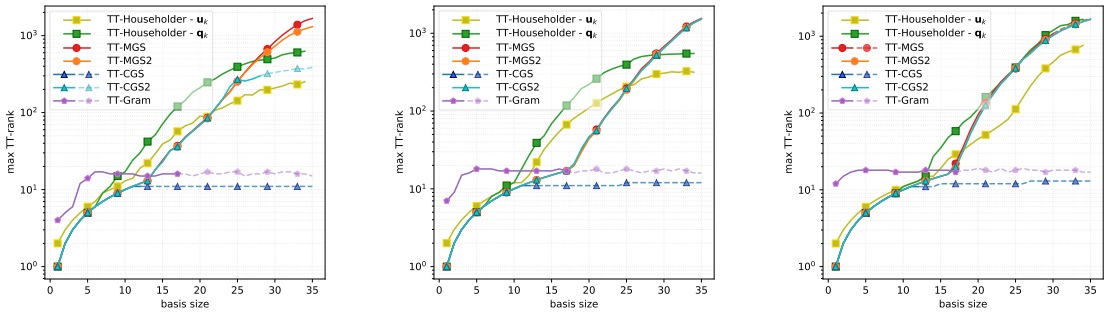
(g) $\delta = 10^{-3}$ (h) $\delta = 10^{-5}$ (i) $\delta = 10^{-8}$

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 $n = 15$.

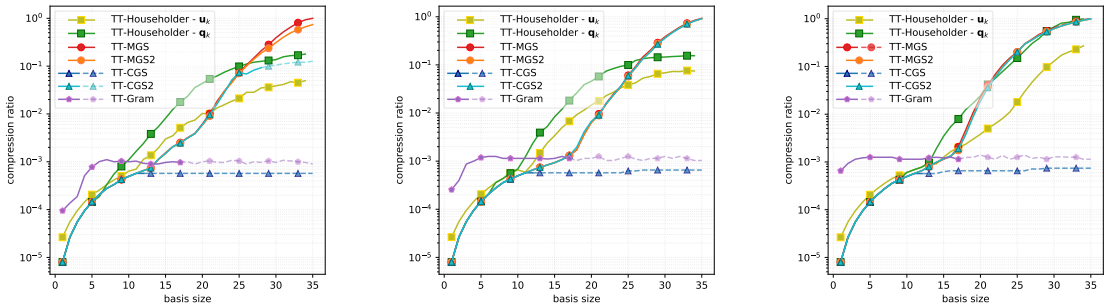
loss of orthogonality, studied in Section 4.1. In Figure 4 we display, for different accuracies δ, the maximum TT-rank, the compression ratio and the gain for the orthogonal TT-vectors $\mathbf{q}_k$ generated by the orthogonalization schemes plus the Householder TT-vector $\mathbf{u}_k$ ones, since, as already mentioned, in many applications $\mathbf{q}_k$ are not computed. Remark that for every rounding accuracy δ, in the corresponding figure the curves get dashed and partially transparent when the corresponding loss of orthogonality become larger than 10^{-1}. In all Figures 4a, 4b and 4c we see that the maximum TT-rank of the orthogonal TT-vectors computed by TT-CGS and TT-Gram schemes stagnates around 10, but we do not have a clear theoretical justification for this phenomenon. The maximal TT-rank of $\mathbf{q}_k$ from TT-Gram algorithm is theoretically bounded by k times the maximal TT-rank of $\mathbf{a}_j$ for $j \in \{1, \ldots, k\}$. When $\mathbf{a}_j$ are generated, they are rounded by maximal TT-rank equal to 1, then the maximal TT-rank of $\mathbf{q}_k$ is bounded by k in our experimental framework. On the other side, 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. From the viewpoint of the maximal TT-rank, TT-Gram outcompetes TT-MGS and TT-Householder for basis size greater than 10, while TT-CGS establishes a lower bound in terms of maximal TT-rank. However the loss of orthogonality of TT-CGS and TT-Gram gets greater than 10^{-1} around $k = 10$. On the other side, the TT-MGS maximal TT-rank curve gets dashed around $k = 20$, while the Householder one never, that is the loss of orthogonality arrives much after for TT-MGS or even does not arrive for TT-Householder. In Figure 4a, the maximal TT-rank of $\mathbf{q}_k$ from TT-MGS overcomes the maximal TT-rank of the Householder TT-vector $\mathbf{u}_k$ and of the Householder orthogonal TT-vector $\mathbf{q}_k$ when the basis size k reaches 20 and 25 respectively. A similar relation among the maximal TT-rank for Householder and for MGS generated orthogonal TT-vectors appears in Figure 4b, but the turning point is set at a different basis size, that is 25 and 30 for the Householder TT-vector $\mathbf{u}_k$ and $\mathbf{q}_k$ respectively. For the last rounding accuracy value $\delta = 10^{-8}$, whose related results are displayed in Figure 4c, the maximal TT-rank of the MGS orthogonal TT-vector reaches the Householder $\mathbf{q}_k$ when the basis size gets larger than 15, overcoming the maximal TT-rank of the Householder $\mathbf{u}_k$ approximately at the same basis size.

As in analysis of TT-Householder algorithm memory requirement, we investigate also the compression ratio for the TT-vectors forming the orthogonal basis produced by the four orthogonalization methods. The compression ratio curves of Figures 4d, 4e and 4f have the same slopes of their corresponding maximal TT-rank curves, but they display clearly the memory needs. The orthogonal TT-vectors from TT-CGS and TT-Gram schemes demand approximately 1% of the total memory necessary to store the full format tensors, as illustrated in Figure 4d, 4e and 4f. The same figures highlight that when the basis size gets greater than 10 and the TT-vectors start to be more collinear, the curves gets dashed and these scheme resulting basis are very poor in terms of orthogonality. From both Figures 4d and 4e, we infer that storing the basis TT-vector generated by TT-Householder scheme request approximately 20% of the memory that we would need to store those tensors in full format. Similarly only 10% of the entire memory requested for full format storage is necessary to store the Householder TT-vector $\mathbf{u}_k$. Finally, for $\delta = 10^{-8}$ storing cost for the Householder basis TT-vector $\mathbf{q}_k$ reaches the same amount of memory demanded for storing them in full format, as depicted in Figure 4f. However from the same figure, we notice that storing the Householder TT-vectors even for $\delta = 10^{-8}$ needs approximately 30% of the memory necessary to store the same tensor in full format. This property makes the TT-Householder algorithm extremely interesting, since usually it is sufficient to store the Householder TT-vectors. TT-Householder algorithm gets even more convenient when we compare its compression ratio curves with the TT-MGS, TT-MGS2 and TT-CGS2 ones. Indeed, for all the rounding accuracy values, the compression ratio curve of TT-MGS and TT-MGS2 always reaches 1, see Figures 4d, 4e and 4f, implying that the memory requested by the TT-vectors from these



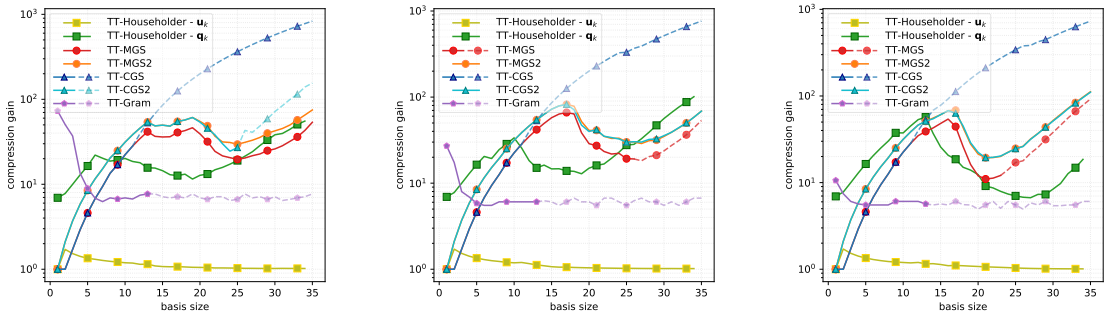
(a) $\delta = 10^{-3}$ (b) $\delta = 10^{-5}$ (c) $\delta = 10^{-8}$

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.

schemes reaches the same amount of memory we would use to store in full format the orthogonal basis tensors. This same consideration holds true for the compression ratio of the orthogonal basis produced by TT-CGS2, for $\delta \in \{10^{-5}, 10^{-8}\}$, while for $\delta = 10^{-3}$, the TT-vectors from TT-CGS2 consume just 20% of the memory we would need to store the same tensors in dense format. The compression gain curves in Figures 4g, 4h and 4i present approximately the same behaviour for the different rounding accuracy values δ. The compression gain of TT-Gram has a pick at the very beginning, then it stagnates around 10 starting from $k = 10$, meaning at the compressing the j-th basis TT-vector reduce of 10 times the memory requirement for $j \geq k$. The gain curves of TT-MGS, TT-MGS2 and TT-CGS2 have approximately the same shape, they increase up to $k = 15$, for the successive 5 iterations the gain curves drop down and around $k = 22$, they rise again. Also the TT-Householder basis gain curve has a similar patter, it first grows, reaching a pick, then it decreases and during the last iterations it rises again. The Householder TT-vector gain curve, as previously observed, has a tiny growth at the beginning, then it drops down and stagnates at a very low value from $k > 10$. Finally the TT-CGS gain curve increases for increasing dimension k of the basis, so when the last basis TT-vector is rounded, it leads to use 0.001 of the memory used to store the TT-vector before rounding it. However as highlighted by the dash style, the TT-CGS and TT-Gram basis have almost entirely lost the orthogonality already at $k = 10$.

4.5 Summary

The computational costs in tensor and matrix case are summarized in Table 1. From the view

	Matrix		**TT-vectors**	
Algorithm	*Computational cost in fp operations*	$\|\mathbb{I}_k - Q_k^\top Q_k\|$	*Computational cost in* `TT-rounding`	$\|\mathbb{I}_k - Q_k^\top Q_k\|$
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-rounding` 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.

point of the memory footprint, thanks to its stability property and the possibility of storing only the TT-vectors $\mathbf{u}_i$, the TT-Householder orthogonalization scheme appears to be the best in many cases, together with TT-CGS2 and TT-MGS2. Figures 4d, 4e and 4f display that when the input TT-vectors are not too much collinear, $k < 15$, TT-MGS2 and TT-CGS2 have a compression ratio approximately equal to TT-Householder. For a basis size between 15 and 25 (or 20 for $\delta = 10^{-8}$) TT-Householder is more expensive than TT-MGS, TT-MGS2 and TT-CGS2 in memory terms. Finally when the input TT-vectors get more and more linearly dependent, the memory need of TT-MGS2 and TT-CGS2 basis is greater or equal than both the TT-Householder basis and Householder TT-vector. However from the viewpoint of the loss of orthogonality, TT-MGS2 performs better than TT-Householder for every rounding accuracy,

while TT-CGS2 for $\delta \in \{10^{-5}, 10^{-8}\}$. Finally from the computational cost side, TT-CGS2 and TT-MGS2 are cheaper than TT-Householder, that is $2m$ `TT-rounding` versus $4m$.

5 Concluding remarks

In the framework where the data representation accuracy is decoupled from the computational one, previously proposed in [2, 4], we investigate the loss of orthogonality of six orthogonalization kernels generalized to the tensor space. The compressed format to represent tensor is the Tensor Train [15]; the considered orthogonalization methods are Classical and Modified Gram-Schmidt (CGS, MGS), their version with re-orthogonalization (CGS2, MGS2), the Gram approach and the Householder transformation. In Section 3 we describe their generalization to the tensor space in TT-format, replying on the compression function called `TT-rounding`, while Section 4 collects the numerical experiments related to the loss of orthogonality and memory requirement of these kernels in TT-format.

Similarly to 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. Indeed TT-Householder requires less memory, but it is computationally more expensive and its orthogonality stagnates around the rounding accuracy. On the other side, 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, but it is more expensive from the viewpoint of the memory consumption. The same considerations hold also for TT-CGS2, under the same hypothesis. The theoretical proof of the numerical results remains an open question and will be the challenge of future works.

References

[1] 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.

[2] 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.

[3] 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.

[4] O. Coulaud, L. Giraud, and M. Iannacito. *A note on GMRES in TT-format.* Research Report RR-9384. Inria Bordeaux Sud-Ouest, 2022.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

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

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

[23] J. H. Wilkinson. *The algebraic eigenvalue problem.* en. Numerical Mathematics and Scientific Computation. Oxford, England: Clarendon Press, Jan. 1988.

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

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

ISSN 0249-6399