



HAL
open science

Orthogonalization schemes in tensor train format

Martina Iannacito, Olivier Coulaud, Luc Giraud

► **To cite this version:**

Martina Iannacito, Olivier Coulaud, Luc Giraud. Orthogonalization schemes in tensor train format. AG 2023 - SIAM Conference on Applied Algebraic Geometry, Jul 2023, Eindhoven / Hybrid, Netherlands. 2023. hal-04394130

HAL Id: hal-04394130

<https://inria.hal.science/hal-04394130>

Submitted on 15 Jan 2024

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



Orthogonalization schemes in tensor train format

SIAM Algebraic Geometry conference,
July 10, 2023, Eindhoven

Martina Iannacito*
joint work with Olivier Coulaud[†] and Luc Giraud[†]

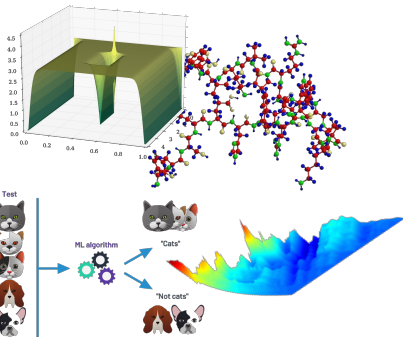
* KU Leuven

[†] Centre Inria at the University of Bordeaux

Registration and travel support for this presentation was provided by SIAM

The context in matrix computation

- Stochastic equations
- Uncertainty quantification
- Quantum chemistry
- Optimization
- Machine learning



reduce their problems to

Least-squares

$$\min_{x \in S} \|b - Ax\|$$

Eigenpairs

$$Ax = \lambda x$$

Linear systems

$$Ax = b$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1,$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1,$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

let $Q_k = [q_1, \dots, q_k]$ be a $n \times k$ matrix

All projections, then subtractions

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

let $Q_k = [q_1, \dots, q_k]$ be a $n \times k$ matrix

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

let $Q_k = [q_1, \dots, q_k]$ be a $n \times k$ matrix

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Projections alternate subtractions

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

let $Q_k = [q_1, \dots, q_k]$ be a $n \times k$ matrix

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

Gram-Schmidt process [Leon et al. 2013]

Given $\{a_h\}_h$ linearly independent vectors, we construct an orthogonal basis $\{q_h\}$ such that

$$q_1 = a_1, \quad q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\|q_1\|^2} q_1, \dots \quad q_{k+1} = a_{k+1} - \sum_{j=1}^k \frac{\langle a_{k+1}, q_j \rangle}{\|q_j\|^2} q_j$$

How to make an algorithm out of it?

let $Q_k = [q_1, \dots, q_k]$ be a $n \times k$ matrix

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

Are these versions equivalent in finite precision arithmetic?

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix} \quad \text{with} \quad \varepsilon = 1e - 10 \quad \text{in fp64}$$

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix} \quad \text{with} \quad \varepsilon = 1e - 10 \quad \text{in fp64}$$

$$Q_{\text{CGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix},$$

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix} \quad \text{with} \quad \varepsilon = 1e - 10 \quad \text{in fp64}$$

$$Q_{\text{CGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}, \quad \langle q_2, q_3 \rangle = \frac{1}{2}$$

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix} \quad \text{with} \quad \varepsilon = 1e - 10 \quad \text{in fp64}$$

$$Q_{\text{CGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}, \quad \langle q_2, q_3 \rangle = \frac{1}{2}$$

$$Q_{\text{MGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon/2 \\ 0 & \varepsilon & -\varepsilon/2 \\ 0 & 0 & \varepsilon \end{bmatrix},$$

From theory to coding: an example II

Not equivalent in general

Classical Gram-Schmidt

All projections, then subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - Q_k Q_k^\top) a_{k+1}$$

Modified Gram-Schmidt

Projections alternate subtractions

$$q_{k+1} \leftarrow (\mathbb{I}_n - q_k \otimes q_k) \dots (\mathbb{I}_n - q_1 \otimes q_1) a_{k+1}$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix} \quad \text{with} \quad \varepsilon = 1e - 10 \quad \text{in fp64}$$

$$Q_{\text{CGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}, \quad \langle q_2, q_3 \rangle = \frac{1}{2}$$

$$Q_{\text{MGS}} = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon/2 \\ 0 & \varepsilon & -\varepsilon/2 \\ 0 & 0 & \varepsilon \end{bmatrix}, \quad \langle q_1, q_3 \rangle = \varepsilon \sqrt{\frac{2}{3}}$$

This example was taken from [Björck 1996].

Loss of orthogonality

Let $Q_k = [q_1, \dots, q_k]$ be the computed orthogonal basis, then its **loss of orthogonality** is

$$\|\mathbb{I}_k - Q_k^\top Q_k\|.$$

It measures the quality in terms of orthogonality of the computed basis. It is linked with the computational precision u and the linear dependency of the input vectors $A_k = [a_1, \dots, a_k]$, estimated through $\kappa(A_k)$.

Matrix		
Source	Algorithm	$\ \mathbb{I}_k - Q_k^\top Q_k\ $
[Stathopoulos et al. 2002]	Gram	$\mathcal{O}(u\kappa^2(A_k))$
[Giraud et al. 2005]	CGS	$\mathcal{O}(u\kappa^2(A_k))$
[Björck 1967]	MGS	$\mathcal{O}(u\kappa(A_k))$
[Giraud et al. 2005]	CGS2	$\mathcal{O}(u)$
[Giraud et al. 2005]	MGS2	$\mathcal{O}(u)$
[Wilkinson 1965]	Householder	$\mathcal{O}(u)$

The advent of tensors

matrix

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix}$$

since the last century



tensor

$$\begin{array}{c} \begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 2 & 5 & 2 & 3 & 1 \\ 1 & 2 & 6 & 4 & 9 \\ 6 & 8 & 7 \end{bmatrix} \end{array}$$

The advent of tensors

matrix

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix}$$

since the last century



tensor

$$\begin{array}{c} \begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix} \\ \begin{bmatrix} 2 & 5 & 3 \\ 1 & 2 & 4 \\ 6 & 8 & 7 \end{bmatrix} \end{array}$$

- + Better representation
- Curse of dimensionality

matrix

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix}$$

since the last century



tensor

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 5 & 3 & 1 \\ 7 & 1 & 2 & 6 & 4 & 9 \\ & & & 6 & 8 & 7 \end{bmatrix}$$

- + Better representation
- Curse of dimensionality

Approximation techniques were proposed

- Canonical Polyadic
- Tucker
- Hierarchical Tucker
- Tensor-Train

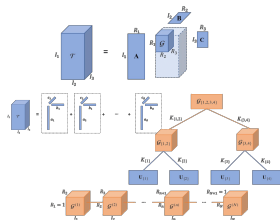


Figure: from [Bi et al. 2022]

matrix

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix}$$

since the last century



tensor

$$\begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 5 & 3 & 1 \\ 7 & 1 & 2 & 6 & 4 & 9 \\ & & & 6 & 8 & 7 \end{bmatrix}$$

- + Better representation
- Curse of dimensionality

Approximation techniques were proposed

- Canonical Polyadic
- Tucker
- Hierarchical Tucker
- Tensor-Train

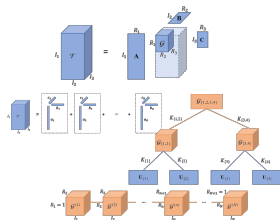


Figure: from [Bi et al. 2022]

Approximation techniques introduce **norm-wise** compression errors

Tensor Train formalism



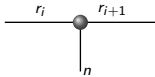
scalar



vector



matrix



order-3 tensor

Tensor Train formalism



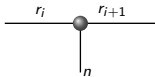
scalar



vector



matrix



order-3 tensor

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be order- d tensor, its TT-representation is

Tensor Train formalism



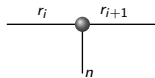
scalar



vector

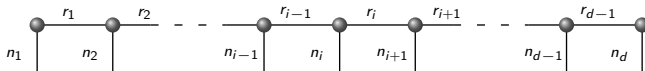


matrix



order-3 tensor

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be order- d tensor, its TT-representation is



+ The storage cost is $\mathcal{O}(dnr^2)$ with $r = \max\{r_i\}$, said **TT-ranks**

Tensor Train formalism



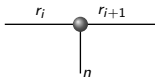
scalar



vector

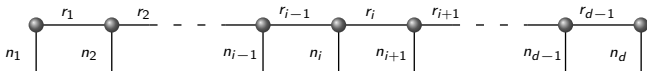


matrix



order-3 tensor

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be order- d tensor, its TT-representation is



- + The storage cost is $\mathcal{O}(dnr^2)$ with $r = \max\{r_i\}$, said **TT-ranks**
- Linear combinations increase the TT-ranks

Tensor Train formalism

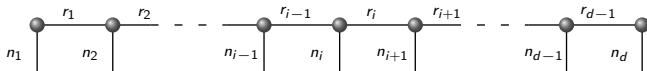
●
scalar

●
|
 n
vector

r_i ●
—
|
 n
matrix

r_i ● r_{i+1}
—
|
 n
order-3 tensor

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be order- d tensor, its TT-representation is



- + The storage cost is $\mathcal{O}(dnr^2)$ with $r = \max\{r_i\}$, said **TT-ranks**
- Linear combinations increase the TT-ranks

TT-rounding [Oseledets 2011]

If \mathbf{z} is an order- d tensor in TT-format and $\delta \in (0, 1)$, then $\bar{\mathbf{z}} = \text{TT-rounding}(\mathbf{z}, \delta)$ such that

$$\|\mathbf{z} - \bar{\mathbf{z}}\| \leq \delta \|\mathbf{z}\|$$

In the Tensor-Train framework

compression precision δ

- norm-wise perturbation
- TT-model [Oseledets 2011]

computational precision u

- component-wise perturbation
- IEEE model [Higham 2002]

In the Tensor-Train framework

compression precision δ

- norm-wise perturbation
- TT-model [Oseledets 2011]

computational precision u

- component-wise perturbation
- IEEE model [Higham 2002]



- How to design orthogonalization kernels for tensor subspace?
- Does compression affect the loss of orthogonality?
- Are tensor results related with the known linear algebra ones?

 $Q, R = \text{CGS}(\mathcal{A}, \delta)$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```
1 for  $i = 1, \dots, m$  do
2    $\mathbf{p} = \mathbf{a}_i$ 
3   for  $j = 1, \dots, i - 1$  do
4      $R(i, j) = \langle \mathbf{a}_i, \mathbf{q}_j \rangle$ 
5      $\mathbf{p} = \mathbf{p} - R(i, j)\mathbf{q}_j$ 
6   end
7    $\mathbf{p} = \text{TT-rounding}(\mathbf{p}, \delta)$ 
8    $R(i, i) = \|\mathbf{p}\|$ 
9    $\mathbf{q}_i = 1/R(i, i)\mathbf{p}$ 
10 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 
```

 $Q, R = \text{MGS}(\mathcal{A}, \delta)$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```
1 for  $i = 1, \dots, m$  do
2    $\mathbf{p} = \mathbf{a}_i$ 
3   for  $j = 1, \dots, i - 1$  do
4      $R(i, j) = \langle \mathbf{p}, \mathbf{q}_j \rangle$ 
5      $\mathbf{p} = \mathbf{p} - R(i, j)\mathbf{q}_j$ 
6   end
7    $\mathbf{p} = \text{TT-rounding}(\mathbf{p}, \delta)$ 
8    $R(i, i) = \|\mathbf{p}\|$ 
9    $\mathbf{q}_i = 1/R(i, i)\mathbf{p}$ 
10 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 
```

They readily write in TT-format.

CGS and MGS with reorthogonalization

 $Q, R = \text{CGS2}(\mathcal{A}, \delta)$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```
1 for  $i = 1, \dots, m$  do
2    $\mathbf{p}_k = \mathbf{a}_i$ 
3   for  $k = 1, 2$  do
4      $\mathbf{p}_k = \mathbf{p}_{k-1}$ 
5     for  $j = 1, \dots, i - 1$  do
6        $R(i, j) = \langle \mathbf{p}_{k-1}, \mathbf{q}_j \rangle$ 
7        $\mathbf{p}_k = \mathbf{p}_k - R(i, j)\mathbf{q}_j$ 
8     end
9      $\mathbf{p}_k = \text{TT-rounding}(\mathbf{p}_k, \delta)$ 
10  end
11   $R(i, i) = \|\mathbf{p}_2\|$ 
12   $\mathbf{q}_i = 1/R(i, i) \mathbf{p}_2$ 
13 end
```

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

 $Q, R = \text{MGS2}(\mathcal{A}, \delta)$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```
1 for  $i = 1, \dots, m$  do
2    $\mathbf{p}_k = \mathbf{a}_i$ 
3   for  $k = 1, 2$  do
4      $\mathbf{p}_k = \mathbf{p}_{k-1}$ 
5     for  $j = 1, \dots, i - 1$  do
6        $R(i, j) = \langle \mathbf{p}_k, \mathbf{q}_j \rangle$ 
7        $\mathbf{p}_k = \mathbf{p}_k - R(i, j)\mathbf{q}_j$ 
8     end
9      $\mathbf{p}_k = \text{TT-rounding}(\mathbf{p}_k, \delta)$ 
10  end
11   $R(i, i) = \|\mathbf{p}_2\|$ 
12   $\mathbf{q}_i = 1/R(i, i) \mathbf{p}_2$ 
13 end
```

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

Let $A = [a_1, \dots, a_m]$, then we look for $A = QR$ with $Q^\top Q = \mathbb{I}_m$
compute the Gram matrix

$$A^\top A = (R^\top Q^\top)QR = R^\top R$$

this is (almost) the **Cholesky** factorization of $A^\top A$ that can be written as

$$A^\top A = R^\top R = LL^\top$$

with the Cholesky factor $L = R^\top$. Thus, it follows

$$A = QR = QL^\top$$

from which we obtain Q by solving a linear system.

$$Q, R = \text{Gram}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

- 1 G is $(m \times m)$ Gram matrix from \mathcal{A}
- 2 $L = \text{cholesky}(G)$
- 3 $\{\mathbf{p}_1, \dots, \mathbf{p}_m\}$ from \mathcal{A} and $(L^T)^{-1}$
- 4 **for** $i = 1, \dots, m$ **do**
- 5 | $\mathbf{q}_i = \text{TT-rounding}(\mathbf{p}_i, \delta)$
- 6 **end**

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

In TT-format the following modifications occur

- $G(i, j)$ is the scalar product of \mathbf{a}_i and \mathbf{a}_j
- The inverse of L^T is explicitly computed
- \mathbf{p}_i is constructed as a linear combination of \mathcal{A} elements

Householder transformation - matrix format

Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.

Householder transformation - matrix format

Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.

$$\begin{bmatrix} | & | & | & | & | \\ a_1 & a_2 & a_3 & a_4 & \\ | & | & | & | & | \end{bmatrix}$$

Householder transformation - matrix format

Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Householder transformation - matrix format

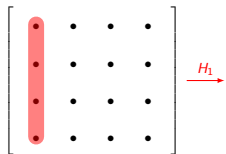
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.


$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} \xrightarrow{H_1}$$

Householder transformation - matrix format

Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.

$$\left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] \xrightarrow{H_1} \left[\begin{array}{cccc} \times & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet \end{array} \right]$$

Householder transformation - matrix format

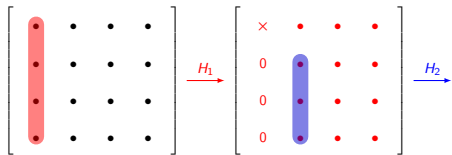
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.



Householder transformation - matrix format

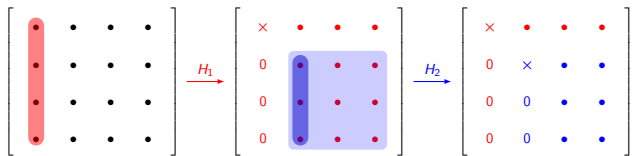
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.



Householder transformation - matrix format

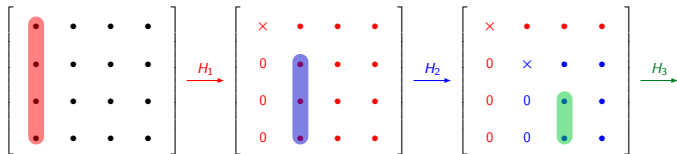
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.



Householder transformation - matrix format

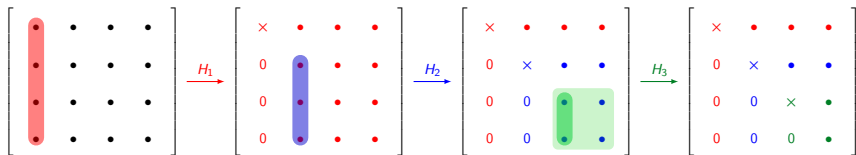
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.



Householder transformation - matrix format

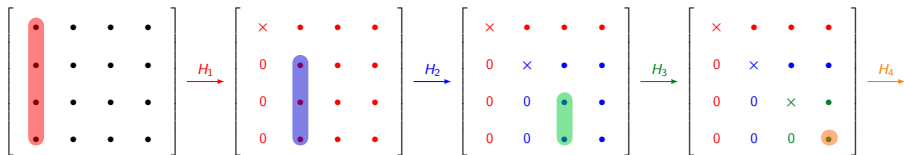
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.



Householder transformation - matrix format

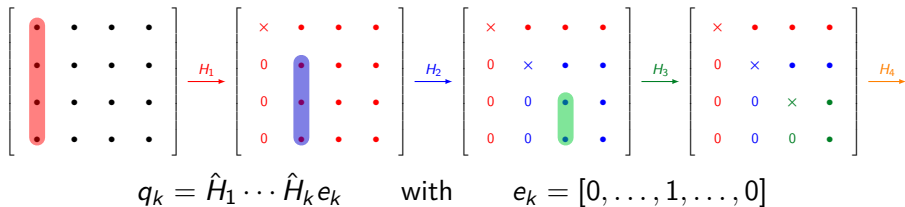
Given a vector $x \in \mathbb{R}^n$ and a direction $y \in \mathbb{R}^n$, the Householder reflector H reflects x along y , i.e.,

$$Hx = \|x\|y \quad \text{with} \quad \|y\| = 1.$$

Thanks to its properties, H writes as

$$H = \mathbb{I}_n - \frac{2}{\|z\|^2} z \otimes z \quad \text{with} \quad z = (x - \|x\|y).$$

Householder kernel uses the input vectors components.

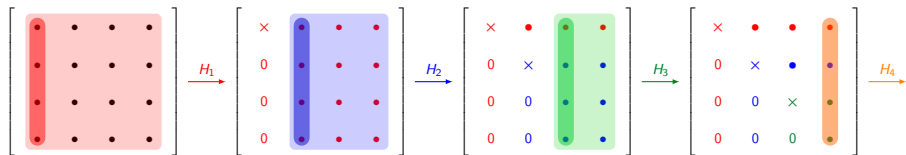


Remark

In TT-formats the components are **not** directly accessible since the tensor is compressed

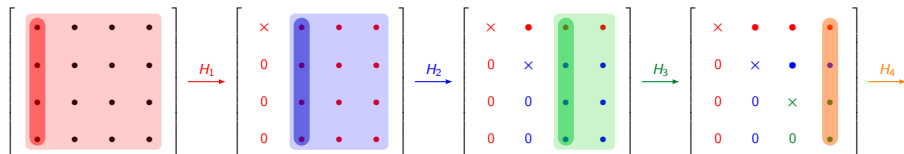
Remark

In TT-formats the components are **not** directly accessible since the tensor is compressed



Remark

In TT-formats the components are **not** directly accessible since the tensor is compressed



Given a TT-vector \mathbf{a} , how to define the Householder TT-vector \mathbf{z} s.t. the reflected TT-vector $\mathbf{b} = (\mathbb{I} - \mathbf{z} \otimes \mathbf{z})\mathbf{a}$ has

- norm equal to \mathbf{a}
- the **same** first $(i - 1)$ entries of \mathbf{a}
- the last $(n - i)$ entries **null**?

Let $\text{vec}(\mathbf{a}) = [\alpha_1, \dots, \alpha_n]$ and $\mathbf{b} = (\mathbb{I} - \mathbf{z} \otimes \mathbf{z})\mathbf{a}_i$ such that
 $\text{vec}(\mathbf{b}) = [\alpha_1, \dots, \alpha_{i-1}, \beta, 0, \dots, 0]$ with $\|\mathbf{a}\| = \|\mathbf{b}\|$

$\mathbf{z}, r = \text{HH-TT-vec}(\mathbf{a}, \mathcal{E}, \delta)$

Input: $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$, $\delta \in \mathbb{R}_+$

```
1  $\mathbf{w} = \mathbf{a}$ ,  $r \in \mathbb{R}^i$  s.t.  $r = 0$ 
2 for  $j = 1, \dots, i - 1$  do
3    $r(j) = \langle \mathbf{a}, \mathbf{e}_j \rangle$ 
4    $\mathbf{w} = \mathbf{w} - r(j)\mathbf{e}_j$ 
5 end
6  $r(i) = \text{sign}(\langle \mathbf{a}, \mathbf{e}_i \rangle) \sqrt{\|\mathbf{a}\|^2 - \|r\|^2}$ 
7  $\mathbf{w} = \mathbf{w} - r(i)\mathbf{e}_i$ 
8  $\mathbf{w} = \text{TT-rounding}(\mathbf{w}, \delta)$ 
9  $\mathbf{z} = \mathbf{w} / \|\mathbf{w}\|$ 
```

Output: \mathbf{z}, r

From the given constraints and Householder reflection properties, we get

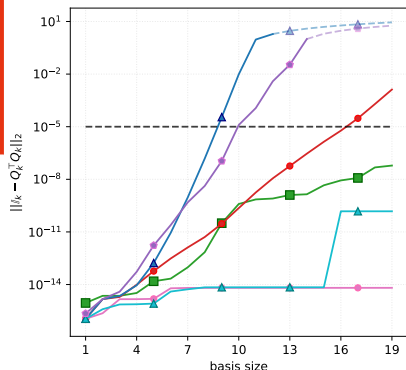
- $\beta^2 = \|\mathbf{a}\|^2 - \sum_{j=1}^{i-1} \alpha_j^2$
 - $\mathbf{z} = \mathbf{w} / \|\mathbf{w}\|$ with
- > $\text{vec}(\mathbf{w}) = [0, \dots, \alpha_i \pm \beta, \alpha_{i+1}, \dots, \alpha_n]$

Numerical experiments in TT-format

Let Δ_d be the TT-Laplacian, then $\{\mathbf{a}_k\}$ are 'Krylov tensors', i.e.,
 $\mathbf{a}_{k+1} = \text{TT-rounding}(\Delta_d \mathbf{x}_k, \text{max_rank} = 1)$ with $\mathbf{x}_{k+1} = \frac{1}{\|\mathbf{a}_{k+1}\|} \mathbf{a}_{k+1}$

Numerical experiments in TT-format

Let Δ_d be the TT-Laplacian, then $\{\mathbf{a}_k\}$ are 'Krylov tensors', i.e.,
$$\mathbf{a}_{k+1} = \text{TT-rounding}(\Delta_d \mathbf{x}_k, \text{max_rank} = 1) \quad \text{with} \quad \mathbf{x}_{k+1} = \frac{1}{\|\mathbf{a}_{k+1}\|} \mathbf{a}_{k+1}$$

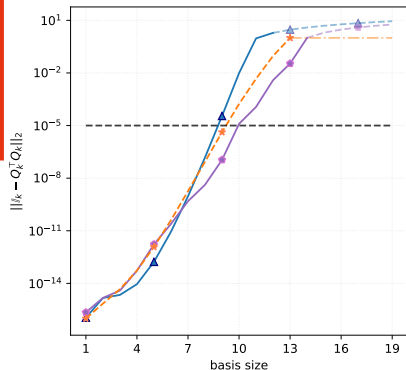


- Gram approach
- CGS
- MGS
- CGS2
- MGS2
- Householder transformation

Figure: 20 tensors of order $d = 3$ and mode size $n = 15$, compression precision $\delta = 10^{-5}$, computational precision $u = \mathcal{O}(10^{-16})$

Numerical experiments in TT-format

Loss of orthogonality for $\{\mathbf{a}_k\}$ 'Krylov tensors'



- Gram approach
- CGS
- $\delta\kappa^2(A_k)$ with

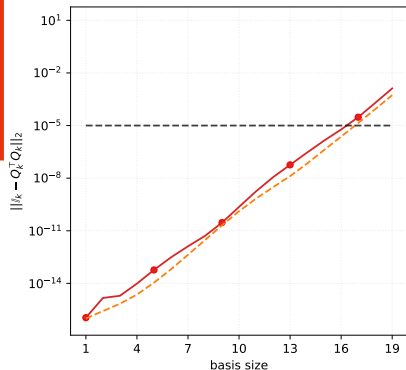
$$A_k = [\text{vec}(\mathbf{a}_1), \dots, \text{vec}(\mathbf{a}_k)]$$

then we conjecture

$$\mathcal{O}(\delta\kappa^2(A_k))$$

Figure: 20 tensors of order $d = 3$ and mode size $n = 15$, compression precision $\delta = 10^{-5}$, computational precision $u = \mathcal{O}(10^{-16})$

Loss of orthogonality for $\{\mathbf{a}_k\}$ 'Krylov tensors'



- **MGS**
- $\delta\kappa(A_k)$ with

$$A_k = [\text{vec}(\mathbf{a}_1), \dots, \text{vec}(\mathbf{a}_k)]$$

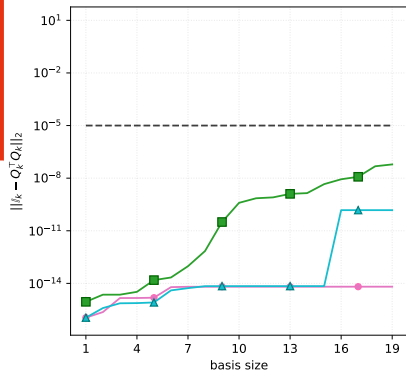
then we conjecture

$$\mathcal{O}(\delta\kappa(A_k))$$

Figure: 20 tensors of order $d = 3$ and mode size $n = 15$, compression precision $\delta = 10^{-5}$, computational precision $u = \mathcal{O}(10^{-16})$

Numerical experiments in TT-format

Loss of orthogonality for $\{\mathbf{a}_k\}$ 'Krylov tensors'



- CGS2
 - MGS2
 - Householder transformation
- then we conjecture

$$\mathcal{O}(\delta)$$

Figure: 20 tensors of order $d = 3$ and mode size $n = 15$, compression precision $\delta = 10^{-5}$, computational precision $u = \mathcal{O}(10^{-16})$

<i>Algorithm</i>	Matrix, theoretical	TT-format, conjecture
	$\ \mathbb{I}_k - Q_k^\top Q_k\ $	$\ \mathbb{I}_k - \mathcal{Q}_k^\top \mathcal{Q}_k\ $
Gram	$\mathcal{O}(u\kappa^2(A_k))$	$\mathcal{O}(\delta\kappa^2(\mathcal{A}_k))$
CGS	$\mathcal{O}(u\kappa^2(A_k))$	$\mathcal{O}(\delta\kappa^2(\mathcal{A}_k))$
MGS	$\mathcal{O}(u\kappa(A_k))$	$\mathcal{O}(\delta\kappa(\mathcal{A}_k))$
CGS2	$\mathcal{O}(u)$	$\mathcal{O}(\delta)$
MGS2	$\mathcal{O}(u)$	$\mathcal{O}(\delta)$
Householder	$\mathcal{O}(u)$	$\mathcal{O}(\delta)$

with u the computational precision, δ the compression precision

Given m input vectors of size n or m TT-vectors of order d

	cost in fp operations	cost in TT-rounding
Gram	$\mathcal{O}(2nm^2)$	m
CGS	$\mathcal{O}(2nm^2)$	m
MGS	$\mathcal{O}(2nm^2)$	m
CGS2	$\mathcal{O}(4nm^2)$	$2m$
MGS2	$\mathcal{O}(4nm^2)$	$2m$
Householder	$\mathcal{O}(2nm^2 - 2m^3/3)$	$4m$

since the TT-rounding is the most expensive step in the TT-kernels

Conclusions and perspectives

- All the 6 kernels can be generalized to the TT-framework
- Loss of orthogonality bounds appears to hold true with
 - > the compression precision δ independent from the machine architecture
 - > the compression precision δ replacing the computational one u
 - > the compression acting norm-wise rather than component-wise

More detailed results can be found at [Coulaud et al. 2022]

Conclusions and perspectives

- All the 6 kernels can be generalized to the TT-framework
- Loss of orthogonality bounds appears to hold true with
 - > the compression precision δ independent from the machine architecture
 - > the compression precision δ replacing the computational one u
 - > the compression acting norm-wise rather than component-wise

More detailed results can be found at [Coulaud et al. 2022]

What is left out?

- Theoretical proof of the loss of orthogonality bounds
- Investigating the quality of the tensor subspace spanned by the orthogonal basis

- [1] Y. Bi et al. "Chapter 1 - Tensor decompositions: computations, applications, and challenges". In: *Tensors for Data Processing*. Ed. by Y. Liu. Academic Press, 2022, pp. 1–30. ISBN: 978-0-12-824447-0. DOI: 10.1016/B978-0-12-824447-0.00007-8.
- [2] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996. DOI: 10.1137/1.9781611971484.
- [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. *On some orthogonalization schemes in Tensor Train format*. Tech. rep. RR-9491. Inria Bordeaux - Sud-Ouest, Nov. 2022.
- [5] L. Giraud, J. Langou, and M. Rozložník. "The loss of orthogonality in the Gram-Schmidt orthogonalization process". In: *Computers & Mathematics with Applications* 50.7 (2005). Numerical Methods and Computational Mechanics, pp. 1069–1075. DOI: 10.1016/j.camwa.2005.08.009.
- [6] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898718027.
- [7] M. Iannacito. "Numerical linear algebra and data analysis in large dimensions using tensor format". Theses. Université de Bordeaux, Dec. 2022.
- [8] S. J. Leon, Å. Björck, and W. Gander. "Gram-Schmidt orthogonalization: 100 years and more". In: *Numerical Linear Algebra with Applications* 20.3 (2013), pp. 492–532. DOI: 10.1002/nla.1839.
- [9] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317. DOI: 10.1137/090752286.
- [10] 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.
- [11] J. H. Wilkinson. *The algebraic eigenvalue problem*. en. Numerical Mathematics and Scientific Computation. Oxford, England: Clarendon Press, 1965.

Thanks for the attention.

Questions?

Inria

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   | construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i)$  from
   |  $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   | for  $j = i, \dots, m$  do
6   |   |  $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   |   | end
8   |  $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9   | end
10  | for  $i = 1, \dots, m$  do
11  |   |  $\mathbf{q}_i = \mathbf{e}_i$ 
12  |   | for  $j = i, \dots, 1$  do
13  |   |   |  $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14  |   |   | end
15  |   |  $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16  | end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

1 $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ canonical basis

2 $\mathbf{w} = \mathbf{a}_1$

3 **for** $i = 1, \dots, m$ **do**

4 construct the Householder TT-vector \mathbf{z}_i and $R(:, i, i)$ from
 \mathbf{w} with $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$ and precision δ

5 **for** $j = i, \dots, m$ **do**

6 $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$

7 **end**

8 $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$ if $i < m$

9 **end**

10 **for** $i = 1, \dots, m$ **do**

11 $\mathbf{q}_i = \mathbf{e}_i$

12 **for** $j = i, \dots, 1$ **do**

13 $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$

14 **end**

15 $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$

16 **end**

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

1 $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ canonical basis

2 $\mathbf{w} = \mathbf{a}_1$

3 **for** $i = 1, \dots, m$ **do**

4 construct the Householder TT-vector \mathbf{z}_i and $R(:, i, i)$ from
 \mathbf{w} with $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$ and precision δ

5 **for** $j = i, \dots, m$ **do**

6 $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$

7 **end**

8 $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$ if $i < m$

9 **end**

10 **for** $i = 1, \dots, m$ **do**

11 $\mathbf{q}_i = \mathbf{e}_i$

12 **for** $j = i, \dots, 1$ **do**

13 $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$

14 **end**

15 $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$

16 **end**

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   |   construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i)$  from
      |    $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   |   for  $j = i, \dots, m$  do
6   |   |    $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   |   end
8   |    $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9 end
10 for  $i = 1, \dots, m$  do
11 |    $\mathbf{q}_i = \mathbf{e}_i$ 
12 |   for  $j = i, \dots, 1$  do
13 |   |    $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14 |   end
15 |    $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```


$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

1 $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ canonical basis

2 $\mathbf{w} = \mathbf{a}_1$

3 **for** $i = 1, \dots, m$ **do**

4 construct the Householder TT-vector \mathbf{z}_i and $R(:, i, i)$ from
 \mathbf{w} with $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$ and precision δ

5 **for** $j = i, \dots, m$ **do**

6 | $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$

7 **end**

8 $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$ if $i < m$

9 **end**

10 **for** $i = 1, \dots, m$ **do**

11 $\mathbf{q}_i = \mathbf{e}_i$

12 **for** $j = i, \dots, 1$ **do**

13 | $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$

14 **end**

15 $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$

16 **end**

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   | construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i, i)$  from
   |  $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   | for  $j = i, \dots, m$  do
6   | |  $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   | end
8   |  $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9 end
10 for  $i = 1, \dots, m$  do
11 |  $\mathbf{q}_i = \mathbf{e}_i$ 
12 | for  $j = i, \dots, 1$  do
13 | |  $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14 | end
15 |  $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

1 $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ canonical basis

2 $\mathbf{w} = \mathbf{a}_1$

3 **for** $i = 1, \dots, m$ **do**

4 construct the Householder TT-vector \mathbf{z}_i and $R(:, i, i)$ from
 \mathbf{w} with $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$ and precision δ

5 **for** $j = i, \dots, m$ **do**

6 $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$

7 **end**

8 $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$ if $i < m$

9 **end**

10 **for** $i = 1, \dots, m$ **do**

11 $\mathbf{q}_i = \mathbf{e}_i$

12 **for** $j = i, \dots, 1$ **do**

13 $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$

14 **end**

15 $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$

16 **end**

Output: $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   | construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i)$  from
   |  $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   | for  $j = i, \dots, m$  do
6   |   |  $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   |   end
8   |  $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9 end
10 for  $i = 1, \dots, m$  do
11   |  $\mathbf{q}_i = \mathbf{e}_i$ 
12   | for  $j = i, \dots, 1$  do
13   |   |  $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14   |   end
15   |  $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   | construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i, i)$  from
   |  $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   for  $j = i, \dots, m$  do
6   |    $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   end
8    $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9 end
10 for  $i = 1, \dots, m$  do
11 |    $\mathbf{q}_i = \mathbf{e}_i$ 
12 |   for  $j = i, \dots, 1$  do
13 | |    $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14 | end
15 |    $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```

$$Q, R = \text{HH}(\mathcal{A}, \delta)$$

Input: $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$

```

1  $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$  canonical basis
2  $\mathbf{w} = \mathbf{a}_1$ 
3 for  $i = 1, \dots, m$  do
4   | construct the Householder TT-vector  $\mathbf{z}_i$  and  $R(:, i, i)$  from
   |  $\mathbf{w}$  with  $\mathcal{F}_i = \{\mathbf{e}_1, \dots, \mathbf{e}_i\}$  and precision  $\delta$ 
5   | for  $j = i, \dots, m$  do
6   | |  $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{z}_i \rangle$ 
7   | end
8   |  $\mathbf{w} = \text{TT-rounding}(\mathbf{a}_{i+1}, \delta)$  if  $i < m$ 
9 end
10 for  $i = 1, \dots, m$  do
11 |  $\mathbf{q}_i = \mathbf{e}_i$ 
12 | for  $j = i, \dots, 1$  do
13 | |  $\mathbf{q}_j = \mathbf{q}_j - \langle \mathbf{q}_j, \mathbf{z}_i \rangle$ 
14 | end
15 |  $\mathbf{q}_i = \text{TT-rounding}(\mathbf{q}_i, \delta)$ 
16 end
Output:  $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ ,  $R$ 

```