Randomized Sketched TT-GMRES for Linear Systems with Tensor Structure

RANDOMIZED SKETCHED TT-GMRES FOR LINEAR SYSTEMS WITH TENSOR STRUCTURE

ALBERTO BUCCI*, DAVIDE PALITTA†, AND LEONARDO ROBOL‡

Abstract. In the last decade, tensors have shown their potential as valuable tools for various tasks in numerical linear algebra. While most of the research has been focusing on how to compress a given tensor in order to maintain information as well as reducing the storage demand for its allocation, the solution of linear tensor equations is a less explored venue. Even if many of the routines available in the literature are based on alternating minimization schemes (ALS), we pursue a different path and utilize Krylov methods instead. The use of Krylov methods in the tensor realm is not new. However, these routines often turn out to be rather expensive in terms of computational cost and ALS procedures are preferred in practice. We enhance Krylov methods for linear tensor equations with a panel of diverse randomization-based strategies which remarkably increase the efficiency of these solvers making them competitive with state-of-the-art ALS schemes. The up-to-date randomized approaches we employ range from sketched Krylov methods with incomplete orthogonalization and structured sketching transformations to streaming algorithms for tensor rounding. The promising performance of our new solver for linear tensor equations is demonstrated by many numerical results.

Key words. Tensor equations, randomized numerical linear algebra, Tensor-Train format.

AMS subject classifications. 65F10, 68W20.

1. Introduction.

In the last decade linear tensor equations of the form

$$\mathcal{A}x = b, \tag{1.1}$$

where $\mathcal{A}$ is an operator acting on $\mathbb{R}^{n_1 \times \cdots \times n_d}$ and x, b are tensors of appropriate dimensions, have come up as very useful tools for describing the discrete problems stemming from a large setting of diverse applications. For instance, in, e.g., quantum chemistry [26, 29] and financial mathematics [46, 48] high-order, possibly stochastic and parametric integral and partial differential equations (PDEs) need to be solved. The discretization of these problems often leads to equations of the form (1.1); see, e.g., [2] and the references therein. Similarly, equation (1.1) can be used to model problems in imaging [23] and deep neural networks [21] as well.

In spite of the large range of application settings where equation (1.1) can be met, only a handful of efficient solvers for its solution have been proposed in the literature. Most of them build upon (alternating) optimization schemes [14, 15, 24] with AMEn [12] and DMRG [34] being two of the most prominent representatives in this class of solvers. In [4] a multigrid procedure for (1.1) is proposed whereas in [11] a tensor-based implementation of the Generalized Minimal Residual (GMRES) method [40] is presented and further studied in [10]. The numerical performance of some of these routines on multicore architectures has been recently investigated in [38].

In this paper, we assume that all the quantities in (1.1) are given in the *Tensor-Train* (TT) format [32]. Indeed, this is one of the most suitable formats for representing (very) high-dimensional problems. Many of the procedures we are going to employ are tailored to this tensor format. However, the whole machinery we present here can be probably adapted to other formats as well.

*Department of Mathematics, University of Pisa, Italy, alberto.bucci@phd.unipi.it.

†Dipartimento di Matematica, $(AM)^2$, Alma Mater Studiorum - Università di Bologna, 40126 Bologna, Italy, davide.palitta@unibo.it

‡Department of Mathematics, University of Pisa, Italy, leonardo.robol@unipi.it

The aim of this work is to significantly improve over the TT-GMRES method presented in [11] by enhancing it with several randomization-based techniques developed in the last years in numerical linear algebra. TT-GMRES is a Tensor-Train formulation of the classic GMRES method. In particular, the basis vectors of the constructed Krylov subspace are represented in terms of TT-tensors, and TT-arithmetic is adopted throughout the iterative scheme. The computational cost of any operation involving TT-tensors depends linearly on the number of modes d of the terms at hand but at least quadratically on their tensor rank; see [32, Section 4]. Therefore, maintaining a small TT-rank during all the TT-GMRES iterations is crucial to obtain an affordable numerical scheme. Unfortunately, both the application of the linear operator $\mathcal{A}$ in (1.1) and the orthogonalization step within TT-GMRES remarkably increase the TT-rank of the basis vectors. A low-rank truncation is thus performed after each of these steps to maintain the TT-ranks under control; see [11] and section 2.3 for further details. As most Krylov methods in a low-rank (tensor) setting, the need to deal with repeated truncations can severely affect the performance of the overall Krylov method; see, e.g., [35, 44] for details and analysis on some low-rank Krylov methods.

We show that randomization can be a strong ally in this setting. First, we design a TT variant of the so-called sketched GMRES (sGMRES) [31]. This allows us to perform only a partial, incomplete reorthogonalization of the basis TT-vectors, with a consequent reduction in their TT-ranks, but still avoiding a drastic delay in the convergence of the underlying Krylov scheme. In addition to remarkably decreasing the overall computational efforts, the incomplete reorthogonalization step allows us to avoid storing the whole basis at all. While all the basis TT-vectors are clearly not necessary during the partial orthogonalization step, we show that their allocation can be avoided also to retrieve the final solution. In particular, we store and utilize only sketches of the basis vectors thanks to the employment of streaming low-rank approximation schemes [25, 43]. Notice that this is in contrast with different state-of-the-art Krylov-based procedures employing incomplete orthogonalization where the final solution is often retrieved by a so-called *two-pass* strategy, namely a second Arnoldi step is performed at the end of the iterative procedure.

All these different tools and ideas have a non-trivial interplay that we analyze in detail, especially from a computational point of view. We will show that our novel method is competitive and often more efficient than state-of-the-art linear solvers for (1.1). On the other hand, the many, diverse techniques we adopt make the derivation of sharp convergence bounds on the overall routine rather tricky and we thus leave this challenging, yet important, aspect to be studied elsewhere.

Here is a synopsis of the paper. Section 2 sees some background material. In particular, we recall the general framework of sGMRES for (standard) linear systems, the TT-format, and TT-GMRES in section 2.1, 2.2, and 2.3, respectively. The main contribution of this paper is illustrated in section 3 where we derive a sketched version of TT-GMRES (TT-sGMRES). All the randomization-based enhancements we equip TT-sGMRES with are presented in the following subsections. As any Krylov technique applied to poorly conditioned systems, also our novel randomization-enhanced TT-sGMRES needs to be preconditioned to get a fast convergence in terms of number of iterations. This aspect is discussed in section 4. In section 5 a panel of diverse numerical results illustrates the potential of our procedure also when compared with different state-of-the-art techniques. The paper ends with some conclusions in section 6.

2. Background. In this section we provide a concise description of two essential ingredients for the construction of sketched TT-GMRES: the sketched GMRES method, and TT-GMRES, together with the main aspects of the TT-format. We only describe what is necessary for this paper, and we refer the reader to [5,45] for further details on the former, and to [11] for the latter.

2.1. Randomized sketching and GMRES. The Generalized Minimal Residual method (GMRES) [40] is a classic iterative scheme for the numerical solution of large-scale, nonsymmetric systems of linear equations. Given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^n$ the algorithm approximates the solution to the linear system $Ax = b$. In particular, starting from an initial guess x_0, a solution x_k of the form

$$x_k = x_0 + V_k y_k, \tag{2.1}$$

is sought. The columns of the matrix $V_k = [v_1, \dots, v_k] \in \mathbb{R}^{n\times k}$ form an orthonormal basis of the k-th Krylov subspace

$$\mathcal{K}_k(A, r_0) = \text{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}, \tag{2.2}$$

where $r_0 = b - Ax_0$ denotes the initial residual. The vector $y_k \in \mathbb{R}^k$ in (2.1) solves the least squares problem

$$y_k = \text{argmin}_y \|AV_k y - r_0\|_2. \tag{2.3}$$

If the basis V_k is constructed by the *full* Arnoldi method, namely an Arnoldi method where a full orthogonalization of the basis vectors is performed, the celebrated Arnoldi relation holds true, i.e.,

$$AV_k = V_{k+1}\underline{H}_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T, \tag{2.4}$$

where $\underline{H}_k \in \mathbb{R}^{(k+1)\times k}$ collects the orthonormalization coefficients and $H_k \in \mathbb{R}^{k\times k}$ is its principal square submatrix; see, e.g., [39].

Thanks to orthogonality of V_k the computation of y_k in (2.3) simplifies as

$$y_k = \text{argmin}_y \|AV_k y - r_0\|_2 = \text{argmin}_y \|\underline{H}_k y - \beta e_1\|_2, \quad \beta = \|r_0\|_2. \tag{2.5}$$

Moreover, the current residual norm $\|Ax_k - b\|_2$ can be cheaply computed; see, e.g., [39, Proposition 6.9]. GMRES terminates whenever $\|Ax_k - b\|_2$ satisfies a certain threshold condition. Otherwise, the Krylov subspace (2.2) is expanded by computing a new basis vector and the scheme continues iteratively.

Many of the practical features and theoretical properties of GMRES depend on the orthogonality of the Krylov basis V_k. However, maintaining the orthogonality of V_k often becomes the bottleneck in practical computations, unless convergence is fast.

Several strategies have been proposed over the years to mitigate this issue. A standard approach is to *restart* either explicitly [39, Section 6.5.6] or implicitly by deflated restarting [30]. Another option to lower the computational cost of the orthogonalization step is to perform an *incomplete* orthogonalization, namely the new basis vector v_k is explicitly orthogonalized only with respect to a certain number ℓ of previously computed v_is; see, e.g., [39, Section 6.5.7]. A strategy with a different flavor is *preconditioning*, where the original problem is implicitly transformed into a problem for which GMRES converges in fewer iterations. Reducing the number of iterations clearly lowers the cost of the orthogonalization as well. However, selecting

the right preconditioner may be tricky, problem-dependent, and its application time consuming. While these approaches all share similar goals, they are often applied independently of each other. In the following, we will show that for tensor equations of the form (1.1) it is often sensible to integrate the aforementioned techniques to attain a very efficient solution scheme.

At this point we focus on the incomplete orthogonalization GMRES scheme. For this GMRES variant, the basis V_k is no longer orthogonal. However, the Arnoldi relation (2.4) still holds and the vector y_k may still be computed as

$$y_k = \text{argmin}_y \|\underline{H}_k y - \beta e_1\|_2. \tag{2.6}$$

Nevertheless, due the nonorthogonality of the basis, $y_k \neq \text{argmin}_y \|AV_k y - r_0\|_2$. It is well-known that this drawback often leads to a delay in the convergence of the solution scheme, in general. However, in the recent literature, it has been shown that when combined with sketching techniques, GMRES with incomplete orthogonalization is often able to retrieve the rate of convergence of the fully orthogonal procedure; see [31].

The integration of sketching and GMRES with incomplete orthogonalization, called *sketched* GMRES (sGMRES), makes use of oblivious subspace embeddings as sketching matrices. In particular, given a k-dimensional subspace $\mathcal{V}_k$, a linear transformation $S \in \mathbb{R}^{s\times n}$, with $s > k$, is a subspace embedding with distortion $\varepsilon \in [0, 1)$ for $\mathcal{V}_k$ if, for any $v \in \mathcal{V}_k$, we have

$$(1-\varepsilon)\|v\|_2^2 \leq \|Sv\|_2^2 \leq (1+\varepsilon)\|v\|_2^2; \tag{2.7}$$

see, e.g., [13, 42, 49]. Notice that the sketching matrix induces the semidefinite inner product $x^T S^T S y$. It can be shown that this is indeed an actual inner product on the space $\mathcal{V}_k$ for which S is an ε-subspace embedding; see, e.g., [3, Section 3.1].

In our case, the space $\mathcal{V}_k$ corresponds to the Krylov subspace (2.2) which is clearly not known a priori. Therefore, we will need to employ *oblivious* subspace embeddings (OSEs) in our work. These are particular transformations S that can be constructed by solely knowing the dimension of the subspace to be embedded and such that (2.7) holds with high probability. Common choices for oblivious subspace embeddings are, e.g., Gaussians, for their theoretical guarantees, or subsampled trigonometric transforms since they allow for fast application; see, e.g., [20].

In [31], the authors integrate sketching and GMRES by replacing the selection of y_k in (2.3) by the following condition

$$y_k = \text{argmin}_y \|SAV_k y - Sr_0\|_2, \tag{2.8}$$

where the basis V_k of the Krylov subspace is computed by an Arnoldi scheme with incomplete orthogonalization. Due to the lack of an Arnoldi-like relation for the sketched quantities in (2.8), in [31] the authors compute y_k by performing

$$y_k = (SAV_k)^\dagger Sr_0. \tag{2.9}$$

In Algorithm 2.1 we report the overall sGMRES algorithm.

2.2. Tensor-Train decomposition. A tensor $\mathcal{T}$ of size $n_1 \times n_2 \times \cdots \times n_d$ is in the TT-format if it can be written element-wise as

$$\mathcal{T}[i_1, \ldots, i_d] = \sum_{\ell_1=1}^{r_1} \cdots \sum_{\ell_{d-1}=1}^{r_{d-1}} C_1[1, i_1, \ell_1] C_2[\ell_1, i_2, \ell_2] \ldots C_d[\ell_{d-1}, i_d, 1]. \tag{2.10}$$

Algorithm 2.1 sGMRES

Input: Matrix $A \in \mathbb{R}^{n\times n}$, right-hand side $b \in \mathbb{R}^n$, initial guess $x_0 \in \mathbb{R}^n$, maximum basis dimension `maxit`, sketching $S \in \mathbb{R}^{s\times n}$, incomplete orthogonalization parameter ℓ, tolerance `tol`.
Output: Approximate solution x_k such that $\|S(Ax_k - b)\| \le \|Sb\| \cdot$ `tol`

1: Set $r_0 = b - Ax_0$, $V_1 = v_1 = r_0/\|r_0\|$, $W_0 = []$
2: **for** $k = 1, \dots,$ `maxit` **do**
3: Compute $\widetilde{v} = Av_k$
4: Update $W_k = [W_{k-1}, S\widetilde{v}]$
5: **for** $i = \max\{1, k-\ell+1\}, \dots, k$ **do**
6: Set $\widetilde{v} = \widetilde{v} - v_i h_{i,k}$, where $h_{i,k} = \widetilde{v}^T v_i$
7: **end for**
8: Set $h_{k+1,k} = \|\widetilde{v}\|$ and $v_{k+1} = \widetilde{v}/h_{k+1,k}$
9: Compute y_k as the solution to (2.9)
10: **if** $\|W_k y_k - Sr_0\| \le \|Sb\| \cdot$ `tol` **then**
11: Go to line 15
12: **end if**
13: Set $V_{k+1} = [V_k, v_{k+1}]$
14: **end for**
15: Set $x_k = x_0 + V_k y_k$

The third-order tensors C_μ of size $r_{\mu-1} \times n_\mu \times r_\mu$ are the TT-cores (where $r_0 = r_d = 1$). Using MATLAB notation, relation (2.10) can be written compactly as a product of d matrices (where the first and last matrices collapse to a row and column vector, respectively) as follows:

$$\mathcal{T}[i_1, \dots, i_d] = C_1[1, i_1, :]C_2[:, i_2, :] \dots C_d[:, i_d, 1].$$

In order to establish the notation, we briefly recall the basic operations on tensors that will be used in the next sections.

Unfoldings. The unfolding $\mathcal{T}_{\le\mu}$ is one of the many ways to matricize a tensor; it is a matrix of size $\prod_{k=1}^{\mu} n_k \times \prod_{k=\mu+1}^{d} n_k$ obtained from merging the first μ modes of $\mathcal{T}$ into row indices and the last $d-\mu$ modes into column indices. A particular case of unfolding is the vectorization, where we transform a tensor $\mathcal{T}$ into a vector with all its entries. This is equivalent to considering $\mathcal{T}_{\le d}$. We will implicitly make use of this tool when discussing GMRES in the Tensor-Train format.

Interface matrices. Each unfolding can be factorized in a low-rank fashion as $C_{\le\mu}C_{>\mu}^T$ where

$$C_{\le_\mu} \in \mathbb{R}^{(n_1\cdots n_\mu)\times r_\mu} \quad \text{and} \quad C_{>\mu} \in \mathbb{R}^{(n_{\mu+1}\cdots n_d)\times r_\mu}.$$

These are sometimes called interface matrices.

The tuple $(r_1, \dots, r_{d-1})$ is called the TT-representation rank of the Tensor-Train defined in (2.10) and it determines the complexity of working with a TT-tensor. For instance, storing a tensor in TT-format requires storing the $O(dnr^2)$ entries of its TT-cores, where $n := \max_\mu(n_\mu)$ and $r \approx r_\mu$ for all $\mu = 1, \dots, d$.* Any tensor

*For the sake of readability, we will often make the simplifying assumption that all TT-ranks

can be trivially written in the TT-format by choosing the TT-representation ranks sufficiently large. The TT-representation rank of a particular tensor $\mathcal{T}$ is by no means unique but there exists an (entry-wise) minimal value which is called the TT-rank of $\mathcal{T}$. The minimal value for r_μ equals the matrix rank of $\mathcal{T}_\mu$. In the rest of the paper will not distinguish between TT-rank and TT-representation rank and simply call $(r_1, \dots, r_{d-1})$ the TT-rank of the tensor $\mathcal{T}$ once relation (2.10) is satisfied for some cores C_μ.

When dealing with vectors in Tensor-Train format, to simplify the matrix-vector products, it is preferable to write matrices in the Tensor-Train operator format.

A matrix A of size $m \times n = (m_1 \times \cdots \times m_d) \times (n_1 \times \cdots \times n_d)$ is in the operator TT-format if it can be written element-wise as

$$A[i_1, \dots, i_d, j_1, \dots, j_d] = D_1[1, i_1, j_1, :]D_2[:, i_2, j_2, :] \dots D_d[:, i_d, j_d, 1]. \tag{2.11}$$

Then, given a vector v in TT-format with cores C_ks, to compute the cores $G_1, \dots, G_d$ of $y = Av$ it is possible to act on each core separately. In formulas

$$G_k[(\ell_{k-1}, \alpha_{k-1}), i_k, (\ell_k, \alpha_k)] = \sum_{j_k} D_k[\alpha_{k-1}, i_k, j_k, \alpha_k] C_k[\ell_{k-1}, j_k, \ell_k].$$

As we can see, the TT-ranks of the MatVec are bounded by the product of the TT-ranks of the matrix and the vector. In iterative schemes like GMRES, several applications of $\mathcal{A}$ are required; without rounding, this unavoidably leads to the TT-ranks becoming too large. Hence, a tensor rounding procedure, or compression, is needed. A given tensor $\mathcal{T}$ is approximated by another tensor $\widetilde{\mathcal{T}}$ with minimal possible TT-ranks $(r_1, \dots, r_{d-1})$ with a prescribed accuracy ε (or a fixed maximal TT-rank R) if:

$$\|\mathcal{T} - \widetilde{\mathcal{T}}\|_F \leq \varepsilon \|\mathcal{T}\|_F \quad (\text{or } r_k \leq R).$$

A quasi-optimal $\widetilde{\mathcal{T}}$ can be obtained by the TT-SVD algorithm [32] with $\mathcal{O}(dnr^3)$ complexity. This is based on performing QR decomposition and truncated SVDs of the interface matrices, exploiting the low-rank structure. Cheaper (and at the same time slightly less accurate) alternatives are available [1,25,32,43], and are often based on randomization.

In this work, we will focus on *streamable* and *randomized* rounding schemes, i.e., algorithms that allow us to find a low-rank representation of a sum of tensors $\mathcal{T}^{(1)} + \ldots + \mathcal{T}^{(m)}$ by performing preliminary contractions of the tensors $\mathcal{T}^{(k)}$, and reconstructing the low-rank approximation of their sum at a later stage. This choice will bring benefits in both speed and accuracy, and will be discussed in further detail in section 3.5.

2.3. TT-GMRES. TT-GMRES [11] is an extension of GMRES aimed at solving tensor equations of the form (1.1) in Tensor-Train format. The main distinction from the classic GMRES is in the representation of the basis "vectors" v_ks which are now given as Tensor-Train vectors. Moreover, TT-GMRES sees the incorporation of rounding steps throughout the process to maintain the TT-ranks of the v_ks within a specified threshold.

---

can be estimated by a single scalar r, and the dimensions n_μ by $n_\mu \approx n$. This will make writing computational complexities much easier. General results can usually be recovered by replacing terms such as dr^j with $\sum_{\mu=1}^d r_\mu^j$, and analogously for the n_μ.

Algorithm 2.2 TT-GMRES

Input: Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, right-hand side b, initial guess x_0 in TT-format, maximum basis dimension `maxit`, tolerance `tol`.
Output: Approximate solution x_k such that $\|\mathcal{A}x_k - b\| \leq \|b\| \cdot$ `tol`

1: Set $r_0 = b - \mathcal{A}x_0$, $\beta = \|r_0\|$, $V_1 = v_1 = r_0/\beta$
2: **for** $k = 1, \ldots,$ `maxit` **do**
3: Set $\eta_k = 1/\|r_{k-1}\|$
4: Compute $\widetilde{v} = \text{ROUND}(\mathcal{A}v_k, \eta_k \cdot$ `tol`$)$
5: **for** $i = 1, \ldots, k$ **do**
6: Set $\widetilde{v} = \text{ROUND}(\widetilde{v} - v_i h_{i,k}, \eta_k \cdot$ `tol`$)$, where $h_{i,k} = \widetilde{v}^T v_i$
7: **end for**
8: Set $h_{k+1,k} = \|\widetilde{v}\|$ and $v_{k+1} = \widetilde{v}/h_{k+1,k}$
9: Compute y_k as the solution to (2.6)
10: Compute $\|r_k\| = \|\underline{H}_k y_k - \beta e_1\|$
11: **if** $\|r_k\| \leq \|b\| \cdot$ `tol` **then**
12: Go to line 18
13: **end if**
14: Set $V_{k+1} = [V_k, v_{k+1}]$
15: **end for**
16: Set $x_k = x_0$
17: **for** $i = 1, \ldots, k$ **do**
18: $x_k = \text{ROUND}(x_k + v_i \cdot (e_i^T y_k), \eta_i \cdot$ `tol`$)$
19: **end for**

In [11] a truncation strategy based on the theory of inexact GMRES [45] is suggested. Heuristically, employing this procedure often keeps the TT-ranks under control. However, there is no clear theoretical link between this strategy and the growth of the ranks. Further exploration and insights in this direction would undoubtedly yield valuable contributions to the field. Similarly, the truncations taking place after the Gram-Schmidt cycle can potentially destroy the orthogonality of the basis making the analysis even trickier. This issue has been studied in [35] in the case of low-rank Krylov methods for multiterm matrix equations.

In Algorithm 2.2 we report the overall TT-GMRES scheme. In lines 4 and 6, $\text{ROUND}(\mathcal{T}, \theta)$ denotes the TT-SVD from [32] that performs a θ-accurate low-rank truncation of the tensor $\mathcal{T}$.

Thanks to the theory of inexact Arnoldi [45], the roundings in line 4 and 6 can be made more aggressive as the method converges, which helps to maintain the basis vectors of moderate ranks. Nevertheless, the full orthogonalization step makes the overall procedure extremely time-consuming, in general. This is one of the reasons why TT-GMRES is not commonly employed for the solution of (1.1) and ALS procedures are often preferred. In the following sections we propose a sketched variant of TT-GMRES which, when equipped with a series of other randomization-based tools, turns out to be competitive with respect to state-of-the-art ALS schemes; see section 5.

3. Sketched TT-GMRES. The previous sections provided the necessary tools and theoretical background to facilitate the understanding of the sketched Tensor-Train GMRES (TT-sGMRES) method, which we present here.

The structure of this section is as follows. In Algorithm 3.2, we begin by outlining the pseudocode for adapting the sGMRES algorithm to the TT-format, akin to the

Algorithm 3.2 Sketched TT-GMRES (TT-sGMRES) – vanilla version

Input: Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, right-hand side b, initial guess x_0 in TT-format, maximum basis dimension `maxit` , tolerance `tol`, sketching S, incomplete orthogonalization parameter ℓ.
Output: Approximate solution x_k such that $\|S(\mathcal{A}x_k - b)\| \leq \|Sb\| \cdot$ `tol`

1: Set $r_0 = b - \mathcal{A}x_0$, $\beta = \|r_0\|$ $V_1 = v_1 = r_0/\beta$, $W_0 = []$
2: **for** $k = 1, \ldots,$ `maxit` **do**
3: Compute $\widetilde{v} = \text{ROUND}(\mathcal{A}v_k, \nu_k \cdot \texttt{tol})$ ▷ Choose ν_k as in sect. 3.2
4: Update $W_k = [W_{k-1}, S\widetilde{v}]$
5: **for** $i = \max\{1, k - \ell + 1\}, \ldots, k$ **do**
6: $h_{i,k} = \widetilde{v}^T v_i$
7: Set $\widetilde{v} = \text{ROUND}(\widetilde{v} - v_i h_{i,k}, \eta_k \cdot \texttt{tol})$ ▷ Choose η_k as in sect. 3.2
8: **end for**
9: Set $h_{k+1,k} = \|\widetilde{v}\|$ and $v_{k+1} = \widetilde{v}/h_{k+1,k}$
10: Compute y_k as the solution to (2.9)
11: **if** $\|W_k y_k - Sr_0\| \leq \|Sb\| \cdot$ `tol` **then**
12: Go to line 18
13: **end if**
14: Set $V_{k+1} = [V_k, v_{k+1}]$
15: **end for**
16: Set $x_k = x_0$
17: **for** $i = 1, \ldots, k$ **do**
18: $x_k = \text{ROUND}(x_k + v_i \cdot (e_i^T y_k), \eta_i \cdot \texttt{tol})$
19: **end for**

TT-GMRES approach given in Algorithm 2.2. The algorithm fundamentally expands upon sGMRES [31], adapting it to the TT-format similarly to how TT-GMRES in [11] builds upon the GMRES method. This simple generalization is not competitive with state-of-the art methods; hence, we delve into a series of refinements and techniques for its efficient implementation, that will turn it into a practical algorithm. In particular, we propose different techniques that exploit randomization to reduce the growth of the ranks, the memory requirements, and the cost of reorthogonalization; these techniques also reduce the cost and improve the stability of forming the final solution. Algorithm 3.3 summarizes these refinements in a detailed implementation.

3.1. Choice of structured sketchings. The first aspect we discuss is the choice of the sketching $S \in \mathbb{R}^{s \times \prod_{k=1}^{d} n_k}$. Notice that this transformation maps vectors in TT-format into standard vectors of $\mathbb{R}^s$. Therefore, no operations with sketched quantities as, e.g., the computation of y_k in line 10, involve tensor arithmetic.

Due to the huge number of columns, using Gaussian transformations or subsampled trigonometric transforms for S is prohibitively expensive and highlights the need for structure embeddings that exploit the TT structure of the vectors.

There are two natural ways to sketch a vector in TT-format, one based on the Kronecker product of matrices, and the other based on the Khatri-Rao product. In particular, given a set of matrices $S_1, \ldots S_d$, with $S_k \in \mathbb{R}^{s_k \times n_k}$, and a TT-vector $\mathcal{T}$ with core tensors $C_k \in \mathbb{R}^{r_k \times n_k \times r_{k+1}}$, if we define $S_\otimes := S_1 \otimes \ldots \otimes S_d$, then the product $S_\otimes \mathcal{T}$ can be easily computed as it results in a TT-vector with cores $D_k = C_k \times_2 S_k$. In other words, the product is distributed across the cores, providing an exponential

speed-up in the computation. Notice that the transformation $S_\otimes$ maps vectors of length $\prod_{i=1}^d n_i$ into vectors of length $s = \prod_{i=1}^d s_i$.

A different option is to draw matrices S_k with the same number of rows and to opt for $S_\odot = S_1 \odot \cdots \odot S_d$ where $\odot$ denotes the row-wise Khatri-Rao product, i.e., the j-th row of $S_\odot$ is the Kronecker product of the j-th rows of the matrices S_ks. The advantage of this second operator is that its application on a TT-vector still splits across the cores, reducing the embedding cost; this computational gain comes at a minimal cost in embedding power [22]. For this reason in our algorithms we opt for Khatri-Rao sketchings. Motivated by the work in [8], we choose the S_ks to be distributed as Gaussian embeddings. Specifically, each S_k is a Gaussian matrix with i.i.d. entries following $\mathcal{N}(0, s^{-1/d})$ for appropriate scaling.

The selection of s will be discussed in detail in section 3.6.

3.2. Truncation policy. One of the aspects that plays an important role in making Algorithm 3.2 competitive is the selection of the truncation tolerance for the rounding steps. Indeed, this must be able to avoid an excessive growth of the TT-ranks.

Algorithm 3.2 sees two main sources of rank growth: the application of $\mathcal{A}$ in line 3 and the linear combinations of the basis vectors which occur both in the orthogonalization phase (line 7) and in the construction of the final solution (line 18). In [11] the author suggests truncating the resulting tensors using the TT-SVD after each of these operations. In particular, as noted in [11], the truncation taking place right after the matrix-vector product $\mathcal{A}v_k$ can be interpreted as an *inexact* application of $\mathcal{A}$ to v_k. Therefore, in principle, the theory of inexact Krylov methods can be employed to select suitable truncation parameters which do not jeopardize the convergence of the overall scheme. The inexact GMRES method has been thoroughly examined by Szyld and Simoncini [45], who introduce a progressively relaxed truncation policy. They prove that the accuracy in the application of $\mathcal{A}$ can be decreased gradually during the iterations. In particular, if $\sigma_{\min}(\mathcal{A})$ denotes the smallest singular value of $\mathcal{A}$, then in [45] the authors suggest employing an iteration-dependent tolerance of the form

$$\nu_k = \frac{\sigma_{\min}(\mathcal{A})}{\text{maxit} \cdot \|r_{k-1}\|}. \tag{3.1}$$

In [11] a similar value for the truncation in the rounding procedure is chosen.

Notice that decreasing the accuracy in the application of $\mathcal{A}$ is equivalent to performing more aggressive low-rank truncations in our context. This is a rather crucial point as the TT-rank of the basis vectors v_k increases with k and being able to significantly reduce it in later iterations is thus extremely beneficial.

The proofs in [45] strongly rely on the orthogonality of the basis V_k. However, the truncation taking place after the Gram-Schmidt step (line 7 in Algorithm 3.2) may potentially destroy the orthogonality of the basis, also in case of a full orthogonalization. This drawback should not get overlooked in general. On the other hand, the basis V_k constructed by TT-sGMRES is non-orthogonal by construction as we perform only an incomplete orthogonalization. Therefore, the truncation in line 7 only affects the local orthogonality of V_k.

In our extensive numerical testing, we experimented with different parameters of the form (3.1), possibly including the conditioning of the basis at the denominator as well. However, it turned out that in our context it is good practice to not truncate the vector $\widetilde{v}_k$ in line 3 of Algorithm 3.2. Indeed, to have a reliable sketching proce-

dure, the update of W_k in line 4 should not involve any truncated quantities so that the computation of y_k in (2.9) is coherent with the original, sketched least squares problem (2.8) and not related to a *nearby* problem. See also section 3.4 for a similar discussion in case of whitening.

On the other hand, to maintain the TT-ranks of the basis vectors under control, along with selecting small values of ℓ (see section 3.4), we perform a truncation step in line 7 of Algorithm 3.2. In particular, the simple strategy of using a constant tolerance $\eta_k \equiv \eta$, for large η, seems to provide the best trade-off between efficiency (the TT-ranks remain small) and rate of convergence (no remarkable delays have been observed). For all the numerical results reported in section 5 we employed $0.1 \leq \eta \leq 0.3$.

There are a few cases, in particular when dealing with preconditioned GMRES, that we discuss in detail in section 4, where this truncation policy is not enough to maintain the TT-rank under control. When this happens, we introduce a further parameter `maxrank` and in the truncation phase we use it as a cap on the TT-ranks of the basis vectors. This can be done easily within the TT-SVD (performing truncated SVDs in all modes) as well as in the randomized schemes that we discuss in section 3.3. This action may cause the generated subspace to deviate from the Krylov subspace, losing some theoretical guarantee over the convergence. However, this does not necessarily imply that convergence is lost. For instance, our experiments show that this strategy is very effective when the application of $\mathcal{A}$ leads to an excessive growth of the ranks. Most importantly, there is no loss of accuracy in the projected and true solution, because we ensure that the action of the operator is sketched *before* performing the rounding.

3.3. Randomized approximation of sums in TT-format. As already mentioned, the rounding procedure and the partial orthogonalization in line 7 of Algorithm 3.2 allow us to mitigate the growth of the TT-ranks due to performing linear combinations of basis vectors. The most immediate way to implement this operation is to perform a rounding after each summation in line 7. However, this strategy would lead to computing up to ℓ extra rounding steps with an excessive increment in the computational efforts. A similar observation applies to the final reconstruction of the solution vector in line 18.

In this section, we propose to exploit recently developed randomization techniques to reduce these costs. Our approach builds upon the algorithms described in [1, 25]. These algorithms are generalizations of randomized low-rank matrix approximation schemes to the tensor realm and provide a significant reduction in computation compared to deterministic algorithms. These approaches are particularly effective for rounding or approximating sums of multiple tensors.

The standard deterministic algorithm for TT-rounding is the TT-SVD [32] and requires first to iteratively orthogonalize the TT-cores of the input TT-format. Other approaches incorporating randomization have been proposed, such as the *Randomize-then-Orthogonalize* in [1], which circumvents this orthogonalization step by applying the randomized SVD algorithm [20] to unfoldings of the full tensor and leveraging the TT-format through the use of Gaussian TT-DRMs (DRM stands for Dimension Reduction Matrix, see Definition 3.1), or a two-sided variant based on generalized Nyström [1]. The latter has been extended in [25] to general sketchings, and is the algorithm that we will exploit in this work. Crucially, the implementation presented in [25], called Streaming Tensor-Train approximation (STTA), has the advantage of being *streamable*, namely it requires to operate with the tensor $\mathcal{A}$ only once. This

feature will be particularly important in our setting, as shown later.

DEFINITION 3.1 (Random Gaussian TT-Tensor). *Given a set of target TT-ranks* $\{\ell_k\}$*, a random Gaussian TT-tensor* $\mathcal{L} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *is such that each core tensor* $\mathcal{T}_{\mathcal{L},k} \in \mathbb{R}^{\ell_{k-1} \times n_k \times \ell_k}$ *is filled with random, independent, normally distributed entries with mean 0 and variance* $1/(\ell_{k-1} n_k \ell_k)$ *for* $1 \leq k \leq d$.

The strength of TT-DRMs is in their ability to reduce the cost of computing partial contractions. In particular, the μth right partial contraction of a TT-tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of ranks $t_1, \ldots, t_{d-1}$ with μth right interface matrix $C_{>\mu}$ and a Gaussian TT-DRM $\mathcal{R} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of ranks $r_1, \ldots, r_{d-1}$ with μth right interface matrix $X_{>\mu}$ is the $t_\mu \times r_\mu$ matrix $R_\mu = C_{>\mu}^T X_{>\mu}$. Analogously, the μth left partial contractions of $\mathcal{T}$ and a Gaussian TT-DRM $\mathcal{L} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and ranks $\ell_1, \ldots, \ell_{d-1}$ is the $\ell_\mu \times t_\mu$ matrix $L_\mu = Y_{\leq \mu}^T C_{\leq \mu}$.

Partial contractions are particularly appealing objects as they can be computed by exploiting the TT structure of the problem, making the computations of the sketchings very cheap. Moreover, having the partial contractions at hand is sufficient to recover the STTA of a tensor.

The STTA algorithm consists of three phases: the generation phase, the sketching phase, and the recovery phase. In the generation phase, we draw the sketchings, specifically Gaussian TT-DRMs in this case. During the sketching phase, we compute the partial contractions mentioned above. Finally, in the recovery phase, we recover the STTA approximant. Below is a summary of the fundamental steps. For more details, please refer to [25].

Given a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in TT-format, with ranks $t_1, \ldots, t_{d-1}$ and target ranks $r_1, \ldots, r_{d-1}$, the STTA algorithm in the generation phase draws random matrices

$$X_{>\mu} \in \mathbb{R}^{(n_{\mu+1} \cdots n_d) \times r_\mu} \quad \text{and} \quad Y_{\leq \mu} \in \mathbb{R}^{(n_1 \cdots n_\mu) \times \ell_\mu}, \quad \text{with} \quad \ell_\mu > r_\mu,$$

then in the sketching phase computes the sketchings

$$\Psi_\mu = (Y_{\leq \mu - 1}^T \otimes I) \mathcal{T}_{\leq \mu} X_{>\mu} \quad \text{and} \quad \Omega_\mu = Y_{\leq \mu}^T \mathcal{T}_{\leq \mu} X_{>\mu},$$

and finally forms the right unfoldings of the TT-cores $\widehat{C}_\mu$ as

$$\widehat{C}_\mu^R = \Omega_{\mu-1}^\dagger \Psi_\mu.$$

A possible way to construct the sketching matrices $X_{>\mu}$ and $Y_{\leq \mu}$ is to use respectively the right and left interface matrices of two Gaussian TT-DRMs of appropriate size.

These steps describe how to compute the STTA approximation of a tensor. To compute the STTA approximant of a linear combination of tensors $a_1 \mathcal{T}^{(1)} + \cdots + a_s \mathcal{T}^{(s)}$, first use the same DRMs to sketch each $\mathcal{T}^{(i)}$ obtaining the $\Psi_\mu^{(i)}$ and $\Omega_\mu^{(i)}$. Next, compute the linear combinations $\Psi_\mu = a_1 \Psi_\mu^{(1)} + \cdots + a_s \Psi_\mu^{(s)}$ and $\Omega_\mu = a_1 \Omega_\mu^{(1)} + \cdots + a_s \Omega_\mu^{(s)}$. Finally, proceed as described above to recover the final approximant.

The STTA algorithm can be exploited in TT-sGMRES during the orthogonalization phase, to compute the weighted sum in line 7 of Algorithm 3.2, and in line 18 to compute the final solution. In particular, since we only need the sketched matrices $\Omega_\mu^{(v_k)}$ and $\Psi_\mu^{(v_k)}$ of each basis vector v_k to form the final solution x_k using STTA, we can get rid of the basis vectors that are no longer needed in the incomplete orthogonalization and store only their sketches. This is particularly beneficial in situations where memory constraints pose a challenge. This means that we can exploit the full

potential of the incomplete orthogonalization also in terms of storage demand while avoiding the possible extra costs coming from a two-pass strategy.

In practice, we have implemented the rounding schemes proposed in [25], and obtained two routines, called STTA_SKETCH and STTA_RECOVER, that perform the following actions:

STTA_SKETCH takes as input a tensor $\mathcal{T}$ and X, Y as described above, and computes the corresponding sketches Ψ_μ and Ω_μ.

STTA_RECOVER takes as input the sketches Ψ_μ and Ω_μ (resp. a linear combination of sketchings) and reconstruct an approximation to the original tensor $\mathcal{T}$ (resp. the linear combination of the tensors).

Throughout the algorithm, we assume that the tensors X, Y have been chosen at the beginning, with suitable dimensions r_μ, ℓ_μ, which we discuss in further detail in section 3.6. We are not able to recommend a choice for these parameters that is suitable for all cases; in the algorithms we let the user provide the values of these parameters.

3.4. Incomplete orthogonalization, restarting, and whitening. From a computational point of view, being able to perform only a local orthogonalization in line 7 of Algorithm 3.2 is key to attain a competitive solver. However, choosing a suitable value of ℓ, the scalar that controls the number of vectors to orthogonalize the newly computed basis vector against, is not straightforward. This is a common issue also in the case of truncated Krylov methods for standard linear systems of equations; see, e.g., [41].

In our context, employing smaller values of ℓ not only decreases the cost of the orthogonalization step itself, thanks to fewer orthogonalizations to perform, but it also induces smaller TT-ranks in the result by reducing the number of tensor sums. This means that adopting a very small ℓ has an impact on the whole solution procedure and is extremely beneficial in reducing the computational efforts devoted to every operation involving the basis vectors in TT-format. In most of our experiments we select $\ell = 1$ obtaining a very successful solution process; see section 5.

If selecting a small ℓ looks very appealing from a computational point of view, such a selection most likely leads to a basis V_k which is terribly ill-conditioned. In [31, Section 5.3] the authors suggest to restart the iterative scheme whenever a too ill-conditioned basis V_k is detected. In particular, if at iteration m, V_m turns out to be (close to) singular, we may construct the residual vector $r_m = b - \mathcal{A}x_m$, and restart the TT-sGMRES iteration using r_m as new initial residual vector in line 1 of Algorithm 3.2. Even though this machinery may help in reducing the impact of working with an ill-conditioned basis, it can potentially lead to important delays in the convergence of the overall solution process. In practice, we have never needed to employ this strategy in our numerical experiments. Moreover, in [17] it has been observed that having an ill-conditioned V_k is not the primary cause of the possible numerical instabilities of sGMRES. Therefore, we do not adopt any restarting strategy in our numerical examples.

A different approach to stabilize sketched Krylov methods is the so-called *whitening*, namely performing an explicit full orthogonalization of the sketched basis SV_k. This inexpensive procedure has a rather important impact in our context as it allows us to rewrite the minimization problem (2.9) in a different way, reminiscent of the projected formulation (2.5) of (standard) GMRES. In particular, in [37] a sketched Arnoldi relation has been derived in the context of Krylov approximations to matrix function evaluations. Let V_k be constructed by a truncated Arnoldi scheme for which

the Arnoldi relation (2.4) holds true. Moreover, let $Q_k T_k = SV_k$ be the skinny QR factorization of the sketched basis SV_k and

$$SV_{k+1} = [Q_k, q_{k+1}] \begin{bmatrix} T_k & t_{k+1} \\ 0 & \tau_{k+1} \end{bmatrix}.$$

Then, we can write

$$SA\widehat{V}_k = S\widehat{V}_k(\widehat{H}_k + \widehat{h}e_k^T) + h_{k+1,k} S\widehat{v}_{k+1} e_k^T = S\widehat{V}_{k+1} \begin{bmatrix} \widehat{H}_k + \widehat{h}e_k^T \\ [0, \dots, 0, h_{k+1,k}] \end{bmatrix}, \tag{3.2}$$

where $\widehat{V}_{k+1} = [\widehat{v}_1, \dots, \widehat{v}_{k+1}] = V_{k+1} T_{k+1}^{-1}$, $\widehat{H}_k = T_k H_k T_k^{-1}$, and $\widehat{h} = t_{k+1} h_{k+1,k} / \tau_k$; see [37, Equation 9]. Even though the transformed basis $\widehat{V}_{k+1}$ is not explicitly available, it is important to notice that this is orthogonal with respect to the sketched inner product $S^T S$, namely $\widehat{V}_{k+1}^T S^T S \widehat{V}_{k+1} = I$. Moreover, at a first glance, the inversion of T_k may look problematic as this matrix carries over the possible ill-conditioning of the nonorthogonal basis V_k. However, in [37, Section 7] it has been shown that, thanks to the triangular pattern of T_k, the forward error attained by computing $z = T_k^{-1} y$ in finite arithmetic behaves much better than what predicted by solely looking at the condition number of T_k.

Thanks to (3.2) and the $S^T S$-orthogonality of $\widehat{V}_k$, the minimization problem (2.8) can be reformulated as

$$\begin{aligned} y_k &= \operatorname{argmin}_y \|SAV_k y - Sr_0\|_2 = \operatorname{argmin}_y \|SAV_k T_k^{-1} T_k y - Sr_0\|_2 \\ &= \operatorname{argmin}_{y = T_k^{-1} z} \|S\widehat{V}_{k+1} z - Sr_0\|_2 \\ &= \operatorname{argmin}_{y = T_k^{-1} z} \left\| \begin{bmatrix} \widehat{H}_k + \widehat{h}e_k^T \\ [0, \dots, 0, h_{k+1,k}] \end{bmatrix} z - \beta e_1 \right\|_2, \quad \beta = \|Sr_0\|_2. \end{aligned} \tag{3.3}$$

If the vector y_k is computed as above, the sketched norm of the residual vector associated to the solution $x_k = x_0 + V_k y_k$, namely $r_k = b - Ax_k$, can be cheaply computed as

$$\|r_k\| = \|S(AV_k y_k - r_0)\| = \|S(A\widehat{V}_k z_k - r_0)\| = \left\| \begin{bmatrix} \widehat{H}_k + \widehat{h}e_k^T \\ [0, \dots, 0, h_{k+1,k}] \end{bmatrix} z_k - \beta e_1 \right\|_2. \tag{3.4}$$

We would like to mention that, to the best of our knowledge, the derivations above result to be new, even though they come from a straightforward combination of the original sGMRES scheme from [31] and the sketched Arnoldi relation presented in [37].

If one wanted to adopt whitening, the only operations to change in Algorithm 3.2 would be the computation of y_k in line 10 and the residual norm evaluation in line 11. Moreover, the storage of the matrix W_k would be no longer necessary whereas the updating of the skinny QR factorization of SV_k would have to be introduced.

Even though it has been shown that whitening is an extremely beneficial practice in contexts like matrix function approximations [37] and the numerical solution of matrix equations [36], we must mention that it does present some peculiar drawbacks in our framework. In particular, the computation of the coefficients collected in the matrix H_k takes place before truncating the current basis vector $\widetilde{v}$ in line 7 of Algorithm 3.2. On the other hand, the sketching S is applied to v_{k+1}, the truncated (and normalized) version of $\widetilde{v}$. Sv_{k+1} is then used to update the skinny QR of SV_{k+1} and

thus obtain the coefficients in T_{k+1} necessary for computing the quantities involved in the projected problem (3.3). As it turned out from our vast numerical testing, this discrepancy in the construction of H_k and T_k may lead to a disagreement between the actual sketched residual norm $\|SAV_k y_k - Sr_0\|$ and its computed value on the right-hand side of (3.4), whenever y_k is computed as in (3.3). We did not observe such a trend when computing y_k by (2.9). Indeed, the use of the pseudoinverse of SAV_k is equivalent to performing an explicit projection without relying on the sketched Arnoldi relation (3.2). Therefore, in all the experiments reported in section 5 the vector y_k is computed by (2.9).

3.5. Building the final solution . The final step of the TT-sGMRES algorithm is the computation of the solution $x_k = x_0 + V_k y_k = x_0 + \sum_i^k v_i[y_k]_i$. For this task, we propose to use the STTA algorithm.

Compared with the classic way to perform this linear combination (adding one term at a time and rounding after each addition), this algorithm offers several advantages, some of which we have already described at the beginning of section 3. In particular, this strategy has lower computational costs and avoids the storage of the basis. Another advantage is that when the basis V_k is not orthogonal, possibly badly conditioned, the classic procedure may face numerical cancellation. On the other hand, our results show that STTA is not affected by this undesirable issue. There is, however, a drawback in using STTA. Indeed, this strategy requires knowing in advance the numerical TT-rank of the solution, or at least an overestimate thereof, which is not available in general. For the moment, we lack valid automatic strategies for estimating the TT-rank of the final solution and in our routines we rely on a user-provided value. That said, for many problems of interest, the TT-ranks of the solution are very low, even lower than those of a single v_i, so that any reasonable heuristic could work.

The reconstructed solution is truncated using a tolerance $\eta \cdot \texttt{tol}$, where `tol` is the prescribed tolerance for the algorithm, and $0 < \eta < 1$ is a fixed parameter. As discussed in the next section, the parameter η is chosen ensuring that the accuracy in the reconstructed solution is maintained.

3.6. Putting it all together. In Algorithm 3.3 we report the TT-sGMRES pseudocode enhanced with all the tools and considerations discussed in the previous sections. In particular, as mentioned in section 3.2, we refrain from performing any low-rank truncation after the application of $\mathcal{A}$ in line 5 whereas we employ a rather large, constant value η (η is either 0.1 or 0.3 in our experiments, and we choose it to ensure that the prescribed tolerance is reached) in the truncations in line 12 and in the final reconstruction. Moreover, any linear combinations involving the basis TT-vectors (line 7 and 21) is carried out by the STTA_RECOVER routine described in section 3.3. To this end, we compute the sketch of the newly defined basis vector v_{k+1} by STTA_SKETCH in line 14. The parameter ℓ_μ for the STTA algorithm (the oversampling) is set to 20.

The number of rows of the sketch S for the TT-sGMRES method is based on the maximum number of iterations. If the user specifies a maximum number `maxit`, the number of rows of S is chosen as twice that number. Optionally, in our code we allow to further tweak this parameters, or to specify a custom sketching S.

REMARK 3.2. In the pseudocode of Algorithm 3.3 we use STTA_SKETCH and STTA_RECOVER to perform the partial reorthogonalization. This is useful especially for sizable values of ℓ. However, in our experiments we often choose $\ell = 1$, for

Algorithm 3.3 Sketched TT-GMRES (TT-sGMRES)

Input: Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, right-hand side b, initial guess x_0 in TT-format, maximum basis dimension `maxit`, tolerance `tol`, sketching S, incomplete orthogonalization parameter ℓ, rounding threshold η.
Output: Approximate solution x_k such that $\|S(\mathcal{A}x_k - b)\| \leq \|Sb\| \cdot$ `tol`

1: Set $r_0 = b - \mathcal{A}x_0$, $\beta = \|r_0\|$ $V_1 = v_1 = r_0/\beta$, $\beta^{[S]} = \|Sb\|$, $W_0 = []$
2: $[\Phi^{(1)}, \Psi^{(1)}] = \text{STTA\_Sketch}(v_1, X, Y)$,
3: **for** $k = 1, \ldots,$ `maxit` **do**
4: Compute $\widetilde{v} = \mathcal{A}v_k$
5: Update $W_k = [W_{k-1}, S\widetilde{v}]$
6: **for** $i = \max\{1, k - \ell + 1\}, \ldots, k$ **do**
7: Set $h_{i,k} = \widetilde{v}^T v_i$ ▷ Only ℓ previous vectors are kept in memory
8: **end for**
9: **for** $\mu = 1, \ldots, d$ **do**
10: Set $\widetilde{\Phi}_\mu = h_{1,k}\Phi_\mu^{(1)} + \ldots + h_{1,k}\Phi_\mu^{(k)}$ and $\widetilde{\Psi}_\mu = h_{1,k}\Psi_\mu^{(1)} + \ldots + h_{1,k}\Psi_\mu^{(k)}$
11: **end for**
12: Set $\widetilde{v} = \text{STTA\_Recover}(\widetilde{\Phi}, \widetilde{\Psi}, \eta \cdot$ `tol`$)$
13: Set $h_{k+1,k} = \|\widetilde{v}\|$ and $v_{k+1} = \widetilde{v}/h_{k+1,k}$
14: Compute $[\Phi^{(k+1)}, \Psi^{(k+1)}] = \text{STTA\_Sketch}(v_{k+1}, X, Y)$
15: Compute y_k as the solution to (2.9)
16: **if** $\|W_k y_k - Sr_0\| \leq \beta^{[S]} \cdot$ `tol` **then**
17: Go to line 21
18: **end if**
19: Set $V_{k+1} = [V_k, v_{k+1}]$
20: **end for**
21: **for** $\mu = 1, \ldots, d$ **do**
22: Set $\widetilde{\Phi}_\mu = [y_k]_1\Phi_\mu^{(1)} + \ldots + [y_k]_k\Phi_\mu^{(k)}$ and $\widetilde{\Psi}_\mu = [y_k]_1\Psi_\mu^{(1)} + \ldots + [y_k]_k\Psi_\mu^{(k)}$
23: **end for**
24: Set $x_k = \text{STTA\_Recover}(\widetilde{\Psi}, \widetilde{\Phi}, \eta \cdot$ `tol`$)$

which it is instead preferable to maintain in memory the last vector and perform the reorthogonalization and rounding explicitly in the TT-format. In our implementation we let the user choose between the two strategies. ◇

4. Preconditioning. It is well-known that, to get a fast rate of convergence in terms of number of iterations, Krylov methods require preconditioning in general. This applies to our TT-sGMRES scheme as well. However, due to the peculiarity of our framework, preconditioners for (1.1) may pose further challenges with respect to preconditioninig operators for standard linear systems. Indeed, in addition to be effective in reducing the number of iterations at a reasonable computational cost, the preconditioner operator must not dramatically increase the rank of the current basis vector. Otherwise, the cost of all the remaining operations in TT-sGMRES would increase possibly jeopardizing the gains coming from running fewer iterations. A similar scenario holds for standard TT-GMRES as well.

Note that, in principle, thanks to the incomplete orthogonalization we perform, TT-sGMRES is less penalized than the standard TT-GMRES [11] if a large number of iterations to converge is needed. Nevertheless, for several practical problems (for instance the ones arising from PDEs, where the condition number of the problem grows

with the problem dimension), preconditioning is essential to ensure convergence in a reasonable amount of time.

Few options for preconditioning tensor equations of the form (1.1) are available in the literature. In [16], a low-rank approximation to $\mathcal{A}^{-1}$ is employed as preconditioner for (1.1). Exponential sums have been proposed in [10, 18, 19, 38].

The main limitation when dealing with preconditioning in tensor Krylov methods is that the operator $\mathcal{A}\mathcal{P}^{-1}$ is usually of a much higher tensor rank than $\mathcal{A}$, and therefore induces a much faster rank growth in the basis. Hence, even if the number of iterations necessary for convergence can be greatly reduced, this does not necessarily correspond to a reduction in computational cost. In the next section, we discuss how sketching can be helpful in this context as well, by limiting the maximum TT-rank that can be reached in the GMRES basis.

We could consider left or right preconditioning, or both at once. We choose to only discuss right preconditioning because it ensures that the residual of the preconditioned problem and of the original one coincide. In a nutshell, assuming the availability of a preconditioner $\mathcal{P}$, right preconditioning modifies lines 4 and 21 in Algorithm 3.3 as follows:

$$\widetilde{v} = \mathcal{A}\mathcal{P}^{-1}v_k, \qquad x_k = \mathcal{P}^{-1}\left[\text{STTA\_RECOVER}(\widetilde{\Psi}, \widetilde{\Phi}, \texttt{tol})\right].$$

As we discuss in section 4.2, this does not always lead to better performances even when the preconditioner works nicely, and extra care is needed to avoid an excessive rank growth. In particular, it turned out that coupling preconditioning with a "maximum rank" rounding step and sketching often leads to competitive results.

4.1. Exponential sum preconditioning. In this work, we have considered preconditioners based on *exponential sums*, that are often suitable for problems arising from PDEs; see, e.g., [10, 18, 19, 38]. In order to construct such preconditioner, it is first necessary to split the operator $\mathcal{A}$ into the following form

$$\mathcal{A} = \widehat{\mathcal{A}} + \bigoplus_{i=1}^{d} A_i, \qquad \bigoplus_{i=1}^{d} A_i := A_d \otimes I \otimes \ldots \otimes I + \ldots + I \otimes \ldots \otimes I \otimes A_1,$$

where the second term (called "Kronecker sum", denoted by $\bigoplus$) is the dominant part of the operator. The Kronecker sum is a summation of d terms, each with a single entry in the Kronecker product different from the identity, which form a commutative family. Then, we precondition by considering $\mathcal{P}$ such that $\mathcal{P}^{-1} \approx (\bigoplus_{i=1}^{d} A_i)^{-1}$. Instead of computing explicitly such $\mathcal{P}$, we directly write $\mathcal{P}^{-1}$. To accomplish this, we rely on exponential sums, that is we determine an approximant for the inverse function $\frac{1}{z}$ of the form

$$\frac{1}{z} = \sum_{j=1}^{\zeta} \alpha_j e^{-\beta_j z} =: E_\zeta(z),$$

where ζ is a positive integer, and such that the approximation is accurate over the spectrum (or better, over the field of values) of $\bigoplus_{i=1}^{d} A_i$. Then, we consider

$$\mathcal{P}^{-1} := E_\zeta\left(\bigoplus_{i=1}^{d} A_i\right) = \sum_{i=1}^{\zeta} \alpha_i \bigotimes_{j=1}^{d} e^{-\beta_i A_j}.$$

In particular, applying $\mathcal{P}^{-1}$ to a tensor $\mathcal{X}$ requires to sum ζ tensors, obtained by performing j-mode multiplications with $e^{-\beta_i A_j}$ for all j. Since j-mode multiplications

do not increase the TT-rank, applying this preconditioner generally increases the TT-ranks of $\mathcal{X}$ by a factor of (at most) ζ.

The difficulty in designing a preconditioner in this class lies in determining the coefficients α_i, β_i. In this work we rely on the procedure described in [10]; we refer the interested reader to [18] and [19, Appendix D] for an in-depth overview. Determining the optimal α_i, β_i is often challenging even when the spectrum is real and known a-priori (see [18]); hence, we often prefer to rely on suboptimal approximations recovered from integral representations of $\frac{1}{z}$ (as done in [10]). It is worth noting that another approach to preconditioning this class of problems involves techniques based on tensor Sylvester equations, such as those presented in [9].

4.2. Sketching and bounded rank roundings. We note that several techniques discussed in the previous sections (e.g., incomplete reorthogonalization) might become less relevant when using a good preconditioner as this leads to convergence in a small number of steps, in general. On the other hand, preconditioning often leads to fast rank growth, possibly making the overall solution process impractical. To mitigate this annoying side-effect, we propose to rely on a low-rank rounding step of the basis with a prescribed maximum rank. This gives little control over the truncation accuracy, making the analysis of the method even trickier. In particular, the distance between truncated and original (not truncated) quantities cannot be quantified in general. However, sketching-based GMRES still works fine in practice and the maximum-rank rounding often leads to important computational advantages. Nevertheless, we must mention that this rounding may induce a slightly larger (but faster) number of iterations when compared to the scenario where this is not performed.

To implement the maximum-rank rounding, when we call the rounding procedure in line 12, we enforce that the TT-rank of v_{k+1} cannot be larger than a maximum prescribed value $r_{\max}$ (component-wise). The choice of this $r_{\max}$ is arbitrary and the optimal value is problem dependent: smaller ranks correspond to faster iterations but slower convergence, whereas higher ranks lead to fewer iterations but with a higher computational cost per iteration.

5. Numerical illustration. In this section, we analyze the proposed enhanced TT-sGMRES algorithm through two distinct applications: one involving convection-diffusion PDEs and another arising from Markov chains in performance and reliability analysis. We compare its performance against other solvers in the TT-format, including TT-GMRES, the vanilla version of TT-sGMRES, and AMEn.

A key aspect of the enhanced TT-sGMRES algorithm is that it provides access only to the sketched residual (2.8), which is typically slightly smaller than the actual residual. To ensure fair comparisons, we set the tolerance for TT-sGMRES lower than that of TT-GMRES. In all our numerical experiments this allowed us to consistently achieve the desired accuracy across all tested scenarios.

The section is divided into two main blocks, in which we analyze respectively the behaviours of the algorithms without and with preconditioning. Before presenting these two blocks experiments in sections 5.2 and 5.3, respectively, we briefly describe the two case studies. In all unpreconditioned experiments, the maximum number of iteration for TT-sGMRES is set to 200 (and thus the sketch S has 400 rows), whereas in the preconditioned examples this number is set to 20 (and S has 40 rows).

The code to replicate the numerical experiments in this section can be downloaded from https://github.com/numpi/tt-sgmres. It requires MATLAB and the TT-Toolbox [33].

5.1. Case studies. Throughout the numerical experiments, we will consider two classes of linear systems, that are briefly described here. The first arises from the discretization of a PDE, whereas the second stems from the analysis of a high-dimensional Markov chain.

5.1.1. A convection-diffusion problem. We consider the computation of the steady-state for a convection-diffusion equation on a d-dimensional box

$$K\Delta u + \langle w, \nabla u\rangle + f = 0, \qquad u : [-1,1]^d \to \mathbb{R},$$

with zero Dirichlet boundary conditions. We choose the parameters $K = 10^{-2}$ and $w = 10^{-2} \cdot [1, \dots, 1] \in \mathbb{R}^d$. The source term is chosen as $f(x) = e^{-10\|x\|_2^2}$. When discretized with finite differences this yields the linear system

$$\left(\bigoplus_{i=1}^{d} [L + D_i]\right) u + f = 0,$$

where f contains the samplings of the source term at the grid points, and the matrices L and D_i discretize the diffusion and convection operators, and are defined as follows:

$$L = \frac{K}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}, \qquad D_i = \frac{w_i}{h}\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}.$$

The choice of the source term $f(x,y) = e^{-10(x^2+y^2)}$ guarantees that, when represented in tensor form, the vector f has rank exactly equal to 1. We remark that the matrices $A_i := L + D_i$ are a natural candidate for building a preconditioner using exponential sums.

5.1.2. High-dimensional Markov chains. Our second test case arises from the description of a Markov chain. The case study we describe is often found when dealing with the evaluation of performance and reliability measures of complex systems, for which a high-dimensional state-space naturally appears. Consider a set of d systems that evolve stochastically as a continuous time Markov chain, each of them endowed with a state-space $\mathcal{S}_i$, with $|\mathcal{S}_i| = n$. Even though the combined state space would be $\mathcal{S} := \prod_{i=1}^{d} \mathcal{S}_i$, which has cardinality n^d, this high-dimensional Markov chain is relatively easy to analyze because every system evolves independently of each other.

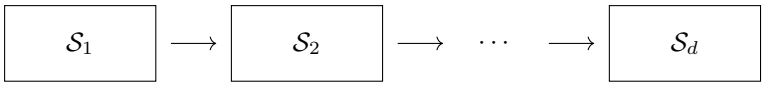
We now modify the Markov chain allowing some state transitions inside $\mathcal{S}$ that involve more than one system (called *synchronizations*). This situation may arise for instance when analyzing computer networks, where failure of one server may impact one or more other services. With this modification, the systems cannot be analyzed independently anymore, and the problem is truly high-dimensional. The computation of the steady-state probabilities can be recast to solving a linear system of the form

$$(Q + W - D)\pi = e, \qquad Q = \bigoplus_{i=1}^{d} Q_i,$$

where Q_i encodes the transition rates of the systems when viewed independently, W adds the synchronization transitions, e is the vector of all ones, and D is a diagonal

matrix to ensure that the row-sum is zero. The vector π contains the steady-state probabilities.

This kind of system has been previously analyzed in [27, 28]. We refer the interested reader to these works and the references therein for further details on the model. In this work, we assume that we have a family of d systems with the following interaction topology:



We assume that when particular transitions in system $\mathcal{S}_i$ are triggered, they change the state in system $\mathcal{S}_{i+1}$, for all $i < d$. As mentioned above, these particular transitions are called *synchronizations*. Note that this fits well with the underlying topology of indices in Tensor-Train, and often allows to represent the steady-state vector in this low-rank format efficiently. The transition rates are chosen as follows:

- Each system behaves as a random walk, with transition rates η_k and μ_k to move forward and backward from state k chosen with a random uniform distribution from $[1, 2]$. All transition rates are chosen independently (that is, the systems are not equidistributed).
- Systems i and $i + 1$ have a synchronized transition such that when both systems are in state $n - 1$, they move together to state n (in the model, this represents the failure of both systems at once). The rate of "joint failure" is equal to 0.1 in our model.

From the linear algebra point of view, this means that the matrices Q_i are all tridiagonal, and W is the sum of matrices obtained by the Kronecker product of $d - 2$ identity matrices (corresponding to the systems not impacted by the failure) and 2 matrices with only one non-zero entry.

REMARK 5.1. The sparse structure of the matrices could be exploited for both case studies in sections 5.1.1 and 5.1.2 to accelerate the matvec operations. For the sake of simplicity, generality, and readability of the code we avoided doing so, but we expect that this could be a further speed-up to our experiments. ◇

5.2. Unpreconditioned GMRES. In this section, we analyze the performance of TT-sGMRES without preconditioning, applied to the two nonsymmetric problems described above: the convection-diffusion case study and the Markov chain one. In these problems, the condition number depends polynomially on n, and therefore we only consider small values of n, and test the scaling with the number of dimensions.

5.2.1. Loss in accuracy of vanilla TT-sGMRES. The first experiment has the aim of showing that the "vanilla" TT-sGMRES presented in Algorithm 3.2 has accuracy problems in the reconstruction of the solution, whereas this is not the case in the "enhanced" TT-sGMRES that we presented in Algorithm 3.3. In fact, since the matrix W_k obtained by running Arnoldi with partial reorthogonalization becomes increasingly poorly conditioned, we expect to find large cancellations when reconstructing the final solution. This leads to poor accuracy if successive relative truncations are performed while computing the sum, which are instead avoided when approximating the sum all at once with the STTA scheme of section 3.3.

For this case we set $\ell = 1$, and run the vanilla and enhanced version of TT-sGMRES on the same problem with $n = 34$ and $d = 4$, for 80 iterations. The two algorithms are exactly the same with the only exception of the final reconstruction described in line 18 of Algorithm 3.2. We then show the value of the residual (recom-

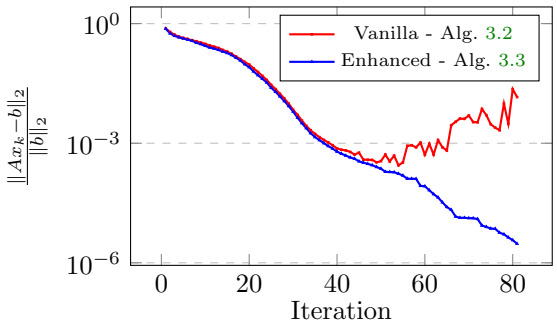


FIG. 1. *Actual residuals of the vanilla and enhanced TT-sGMRES algorithms computed after each iteration for the PDE problem in section 5.1.1 with $d = 4$ and $n = 34$.*

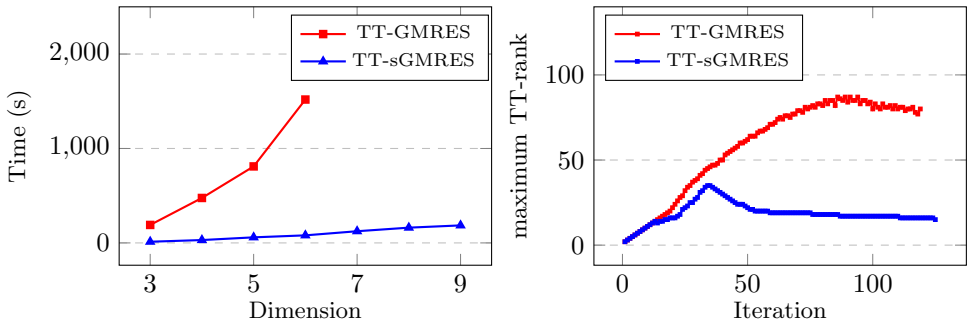


FIG. 2. *On the left, we report the runtime of the TT-GMRES and TT-sGMRES algorithms on convection-diffusion PDE problems of size $n = 64$ across various dimensions d and accuracy 10^{-4}. On the right, we plot the maximum TT-ranks of the base vectors generated by TT-GMRES and TT-sGMRES with $d = 6$, $n = 64$ and $\texttt{tol} = 10^{-4}$. In the right experiment, TT-GMRES converged in 1528.22 seconds with respect to the 80.03 seconds of TT-sGMRES.*

puted exactly) at each iteration, and report it for both schemes in Figure 1. While the enhanced version shows a nice convergence plot, the vanilla one has a semiconvergent behavior, and starting from iteration 40 the cancellation errors completely dominate with respect to the achieved accuracy.

Since the one depicted in Figure 1 is a common behavior of the vanilla TT-sGMRES, in the following we focus only on Algorithm 3.3.

5.2.2. TT-GMRES vs TT-sGMRES. In the second experiment we consider again the PDE problem from section 5.1.1, and we compare the timings of the enhanced TT-sGMRES with the standard TT-GMRES. The problem is considered for d ranging from 3 to 9, and n fixed to 64. The stopping criterion is $\texttt{tol} = 10^{-4}$, and we aborted the execution if the runtime exceeded one hour. The results are reported in Figure 2 (left).

In this test, the enhanced TT-sGMRES is faster than TT-GMRES for all dimensions. The speedup arises from two phenomena: we only perform partial reorthogonalization and the TT-ranks remain smaller. To better describe the latter phenomenon we provide another plot in Figure 2 (right), in which we show the maximum TT-rank of the vectors v_k generated by the two algorithms for $d = 5$ (for other dimensions we obtained analog results). We can see that TT-GMRES operates with higher TT-ranks with respect to the enhanced TT-sGMRES. On one side, higher TT-ranks lead

to more expensive arithmetic operations, and on the other side the fact that TT-GMRES performs full orthogonalization increases the number of dot products; the enhanced TT-sGMRES, instead, only requires a constant number of these dot products per iteration. We also observe that the enhanced TT-sGMRES requires a few more iterations to converge than TT-GMRES, mostly because the sketched tolerance is set to $0.3 \cdot \texttt{tol}$, with `tol` being the TT-GMRES threshold, in order to accommodate with the estimation error for the residual.

5.2.3. Gap between sketched and actual residual. In the previous examples we have set the tolerance for the stopping criterion in TT-sGMRES slightly smaller than the one for TT-GMRES. This is because the stopping criterion for the former relies on the sketched relative residual $\|S(Ax_k - b)\|/\|Sb\|$, which in practice is often a good estimate of the true residual ($\|b - Ax_k\|/\|b\|$) up to a small constant.

In this experiment, we show the distance between the sketched and the true residuals, for various dimensions $d = 3, 5, 7, 9$. The results along all the iterations for the PDE problem with $n = 64$ are reported in Figure 3. The maximum number of iterations is set to 500, and the number of rows of S to 1000, so at the end of the algorithm the dimension of the sketched space is about twice as the dimension of the subspace where the residual lives. The tolerance was set to $\texttt{tol} = 10^{-6}$.

The plots show that the gaps are larger for higher values of d. One of the causes is that the embedding power of the Khatri-Rao embeddings depends on the dimension d, the other, and most impactful, is that STTA recovers an approximate low-rank approximation up to some constants depending exponentially on d.

In this experiment and in the tests that we ran, this gap has always been less than 10; however, for higher dimensions, this gap could become significant, because of the loss of accuracy of the STTA approximation. It is possible to compensate this effect and reduce the STTA constants by increasing the parameter ℓ_μ in the generation of the sketchings phase. For further details, see [25].

5.2.4. Markov case study without preconditioning. We replicated the experiments for the PDE problems on the Markov case study, which led to a similar behavior. We report in this section the timings for running TT-GMRES and TT-sGMRES, which are plotted in Figure 4, on the left. We can see that, as in the PDE case study, the proposed algorithm can deal with the increasing dimensionality without a significant increase in computational times (with respect to TT-GMRES).

On the right, in the same figure, the ranks throughout the iterations are reported. In contrast to the PDE example, the rank of the operator describing the Markov chain grows with d (linearly), and therefore the problem becomes increasingly challenging for high dimensions.

We remark, however, that without preconditioning the performances of the algorithm are still far from those of AMEn (which requires less than 1 second to converge for $d = 4, 5, 6$). Therefore in the next section, we focus on the preconditioned case.

5.3. Numerical tests with preconditioning. In this section, we reconsider the case studies presented above, and include an option to precondition the TT-sGMRES iteration. In both cases, this is necessary when the dimensions n_i become large, because the condition number grows polynomially in n. We will use exponential sums to build preconditioners for all examples for simplicities, but we do not expect major differences in case other preconditioners are used. Since AMEn requires access to the Tensor-Train operator [12] (and not only the matvec operation), preconditioning cannot be easily incorporated. Hence, we compare the results with AMEn on the

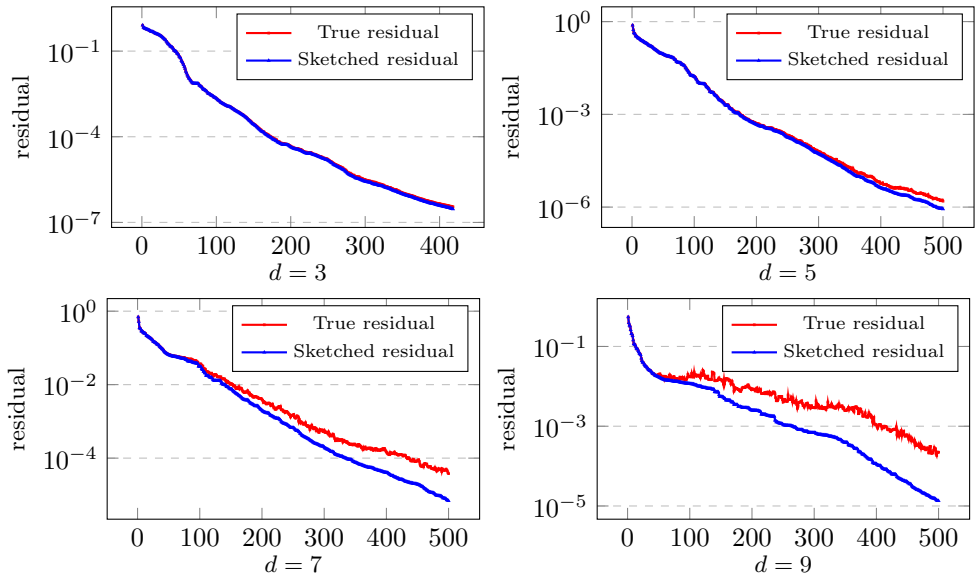


FIG. 3. *The above plots report the difference between the sketched residual and the true residual, for different values of d.*

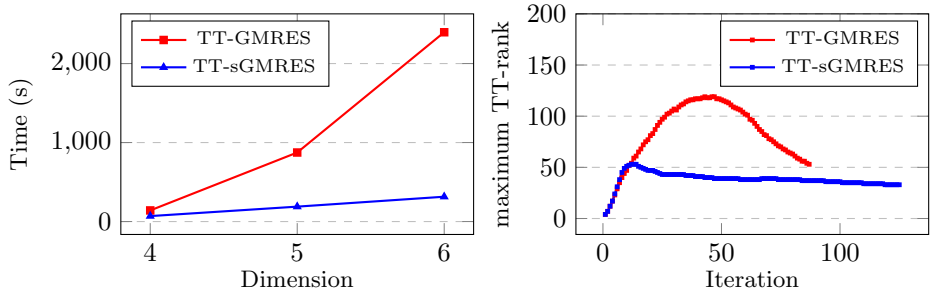


FIG. 4. *On the left, the comparison between running TT-GMRES and TT-sGMRES for the Markov test case, with different values of d and n = 64. On the right, the behavior of ranks of the basis vectors during the iterations, in the case d = 5.*

unpreconditioned problem.

5.3.1. Convection-diffusion. For the convection-diffusion problem in the case $d = 5$, we employed an exponential sum preconditioner with

$$\mathcal{P}^{-1} = \sum_{i=1}^{\zeta} \alpha_i \bigotimes_{j=1}^{d} e^{-\beta_i A_j},$$

as detailed in section 4. We selected $\zeta = 17$. In addition, we tested different values of `maxrank` for the basis recompression. As a rule of thumb, we expect smaller values of `maxrank` to yield faster iterations, but slower convergence, or even stagnation. On the other hand, higher values of `maxrank` will be closer to the GMRES iteration without rounding and usually yield a better convergence, but with a much higher computational cost per iteration.

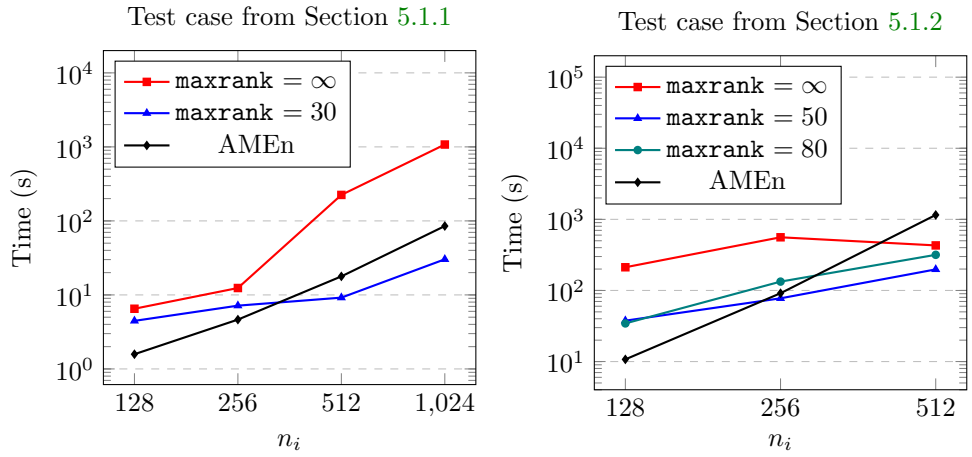


FIG. 5. *Left: Runtime of TT-sPGMRES iteration for the convection-diffusion problem in section 5.1.1 with variable n_i and $d = 5$; the target tolerance in this example is 10^{-8}, and different values of* `maxrank` *are used. Right: Runtime of TT-sPGMRES iteration for the Markov problem in section 5.1.2 with variable n_i and $d = 5$; the target tolerance in this example is 10^{-6}, and different values of* `maxrank` *are used. AMEn is run with standard parameters, and is taken from the TT-Toolbox [33].*

For this example, we tested $\texttt{maxrank} = \infty$ and $\texttt{maxrank} = 30$; in addition, we have compared the performance with the AMEn solver in the TT-Toolbox (with default parameters, and a maximum number of sweeps set to 200 in order to achieve the target tolerance). The target tolerance was set to 10^{-8}, and as usual we reduced it by a factor 10 in TT-sPGMRES, to account for the constant in the estimation of the residual by sketching.

All approaches achieved the required accuracy, and the timings for different values of n_i are reported in Figure 5 (left). We see from the results in Figure 5 (left) that allowing the ranks to grow unbounded does not yield optimal performances. With both `maxrank` set to ∞ and 30, TT-sPGMRES convergences in 4 iterations to the desired tolerance with this choice of preconditioner. Moreover, when choosing $\texttt{maxrank} = 30$ our algorithm becomes competitive and, for this example, it is faster than AMEn.

Without preconditioning, the ranks stay nicely bounded, but the number of iterations is so large that the method cannot be competitive with the choices above. With $\texttt{maxrank} = \infty$, the iteration reaches rank 433 for $n_i = 1024$, so it is rather memory demanding. Hence, this example shows how using a bounded rank can be essential when incorporating preconditioning.

5.3.2. Preconditioning for the Markov test case. We ran a similar experiment for the test case arising from Markov chains. In that case, a natural choice for the preconditioner is to consider the infinitesimal generator Q obtained by ignoring all interactions between the different systems, and dropping the matrix W (following the notation used in section 5.1.2).

The matrix Q is a Kronecker sum, and therefore its approximate inverse can be constructed by exponential sums, exactly as for the convection-diffusion test case. For this problem, we selected $\zeta = 33$. We ran the same tests, using systems with a number of states ranging from 128 to 512, and requiring tolerance 10^{-6}. This problem is more

challenging than the PDE case, and we ran our algorithm with $\texttt{maxrank} \in \{50, 80, \infty\}$. As for the PDE case, we used $\eta = 0.1$ as a safety factor to make sure that if the sketched residual is below $\eta \cdot \epsilon$ then the true residual is around ϵ or less. The results are reported in Figure 5 (right).

When running with $\texttt{maxrank} = \infty$ we encountered the same behavior of the PDE case study of the previous section: the rank grows quickly (up to about 220 in this example), the algorithm is slowed down and can easily encounter memory issues. On the other hand, using lower values of `maxrank` makes the algorithm competitive with AMEn, and even faster for large values of n_i, and corresponding badly conditioned problems. In this example, $\texttt{maxrank} = 50$ only manages to reach a true accuracy of about 10^{-5}, whereas $\texttt{maxrank} = 80$ achieves the target of 10^{-6}.

6. Conclusions. In this work, we presented and analyzed a sketched version of TT-GMRES, called TT-sGMRES, a novel algorithm that combines the winning strategies of sketch GMRES and TT-GMRES. Through various methodological refinements, we demonstrated that the introduction of sketching and randomization brings significant benefits, primarily by greatly reducing the cost of orthogonalization and limiting the ranks of tensors during the iteration. Additionally, the approach based on a streamable method allowed us to overcome one of the classic storage problems, namely the allocation of the whole basis. In particular, once the vectors of the Krylov basis are computed, they are sketched and then discarded, and this is sufficient to recover the solution upon convergence.

The experiments conducted validate the effectiveness of the proposed method. Not only did the TT-sGMRES prove to be significantly superior to the classical TT-GMRES, but in many cases, it was also competitive with established solvers such as AMEn. Another advantage of our method is the possibility of leveraging preconditioners to further improve its performance, making it an extremely promising method for a wide range of applications.

Although we focused on the TT-format, many of the improvements introduced can be tested and exploited in a broader range of cases where vectors can be compressed in a low-rank format and streamable algorithms for their linear combinations are available. For example, this approach could be applied to the Tucker format using the methods in [6,7,47], and efforts could be made to extend it to the case of the Tree Tensor Network format.

In conclusion, TT-sGMRES represents a significant advancement in the state of the art, offering an efficient and scalable scheme for solving high-dimensional linear systems.

Acknowledgments. The authors are members of the INdAM Research Group GNCS that partially supported this work through the funded project GNCS2024 with reference number CUP_E53C23001670001.

The work of Alberto Bucci and Davide Palitta was partially supported also by the European Union - NextGenerationEU under the National Recovery and Resilience Plan (PNRR) - Mission 4 Education and research - Component 2 From research to business - Investment 1.1 Notice Prin 2022 - DD N. 104 of 2/2/2022, entitled "Low-rank Structures and Numerical Methods in Matrix and Tensor Computations and their Application", code 20227PCCKZ – CUP J53D23003620006. Alberto Bucci was partially supported by Charles University Research program PRIMUS/21/SCI/009. The work of Leonardo Robol was partially supported by the National Research Center in High Performance Computing, Big Data and Quantum Computing (CN1 – Spoke 6), by the MIUR Excellence Department Project awarded to the Department

of Mathematics, University of Pisa (CUP I57G22000700001), and by the Italian Ministry of University and Research (MUR) through the PRIN 2022 "MOLE: Manifold constrained Optimization and LEarning", code: 2022ZK5ME7 MUR D.D. financing decree n. 20428 of November 6th, 2024 (CUP I53C24002260006).

REFERENCES

[1] H. Al Daas, G. Ballard, P. Cazeaux, E. Hallman, A. Miedlar, M. Pasha, T. W. Reid, and A. K. Saibaba, *Randomized algorithms for rounding in the tensor-train format*, SIAM J. Sci. Comput., 45 (2023), pp. A74–A95, https://doi.org/10.1137/21M1451191.

[2] M. Bachmayr, *Low-rank tensor methods for partial differential equations*, Acta Numerica, 32 (2023), p. 1–121, https://doi.org/10.1017/S0962492922000125.

[3] O. Balabanov and A. Nouy, *Randomized linear algebra for model reduction. Part I: Galerkin methods and error estimation*, Adv. Comput. Math., 45 (2019), pp. 2969–3019.

[4] M. Bolten, K. Kahl, and S. Sokolović, *Multigrid Methods for Tensor Structured Markov Chains with Low Rank Approximation*, SIAM Journal on Scientific Computing, 38 (2016), pp. A649–A667, https://doi.org/10.1137/140994447.

[5] A. Bouras and V. Frayssé, *A relaxation strategy for inexact matrix-vector products for krylov methods*, tech. report, Technical Report TR/PA/00/15, CERFACS, France, 2000.

[6] A. Bucci and B. Hashemi, *A sequential multilinear Nyström algorithm for streaming low-rank approximation of tensors in Tucker format*, Applied Mathematics Letters, (2024), p. 109271.

[7] A. Bucci and L. Robol, *A Multilinear Nyström Algorithm for Low-Rank Approximation of Tensors in Tucker Format*, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 1929–1953, https://doi.org/10.1137/23M1599343.

[8] Z. Bujanović, L. Grubišić, D. Kressner, and H. Y. Lam, *Subspace embedding with random khatri-rao products and its application to eigensolvers*, arXiv preprint arXiv:2405.11962, (2024).

[9] A. A. Casulli, *Tensorized block rational krylov methods for tensor sylvester equations*, arXiv preprint arXiv:2306.00705, (2023).

[10] O. Coulaud, L. Giraud, and M. Iannacito, *A robust GMRES algorithm in Tensor Train format*, arXiv preprint arXiv:2210.14533, (2022).

[11] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian J. Numer. Anal. Math. Modelling, 28 (2013), pp. 149–172, https://doi.org/10.1515/rnam-2013-0009.

[12] S. V. Dolgov and D. V. Savostyanov, *Alternating Minimal Energy Methods for Linear Systems in Higher Dimensions*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2248–A2271, https://doi.org/10.1137/140953289.

[13] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Subspace sampling and relative-error matrix approximation: Column-based methods*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2006, pp. 316–326, https://doi.org/10.1007/11830924_30.

[14] S. Etter, *Parallel ALS Algorithm for Solving Linear Systems in the Hierarchical Tucker Representation*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2585–A2609, https://doi.org/10.1137/15M1038852.

[15] I. Georgieva and C. Hofreither, *Greedy low-rank approximation in Tucker format of solutions of tensor linear systems*, Journal of Computational and Applied Mathematics, 358 (2019), pp. 206–220, https://doi.org/https://doi.org/10.1016/j.cam.2019.03.002.

[16] L. Giraldi, A. Nouy, and G. Legrain, *Low-Rank Approximate Inverse for Preconditioning Tensor-Structured Linear Systems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1850–A1870, https://doi.org/10.1137/130918137.

[17] S. Güttel and I. Simunec, *A Sketch-and-Select Arnoldi Process*, SIAM Journal on Scientific Computing, 46 (2024), pp. A2774–A2797, https://doi.org/10.1137/23M1588007.

[18] W. Hackbusch, *Computation of best L^∞ exponential sums for 1/x by Remez'algorithm*, Computing and Visualization in Science, 20 (2019), pp. 1–11.

[19] W. Hackbusch et al., *Hierarchical matrices: algorithms and analysis*, vol. 49, Springer, 2015.

[20] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[21] Y. Ji, Q. Wang, X. Li, and J. Liu, *A Survey on Tensor Techniques and Applications in Machine Learning*, IEEE Access, 7 (2019), pp. 162950–162990, https://doi.org/10.1109/

ACCESS.2019.2949814.
[22] R. Jin, T. G. Kolda, and R. Ward, *Faster Johnson-Lindenstrauss transforms via Kronecker products*, Inf. Inference, 10 (2021), pp. 1533–1562, https://doi.org/10.1093/imaiai/iaaa028.
[23] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, *Third-Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 148–172, https://doi.org/10.1137/110837711.
[24] D. Kressner, M. Steinlechner, and B. Vandereycken, *Preconditioned low-rank riemannian optimization for linear systems with tensor product structure*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2018–A2044, https://doi.org/10.1137/15M1032909.
[25] D. Kressner, B. Vandereycken, and R. Voorhaar, *Streaming tensor train approximation*, SIAM Journal on Scientific Computing, 45 (2023), pp. A2610–A2631.
[26] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, Zürich: EMS, 2008.
[27] G. Masetti and L. Robol, *Computing performability measures in Markov chains by means of matrix functions*, J. Comput. Appl. Math., 368 (2020), p. 19, https://doi.org/10.1016/j.cam.2019.112534. Id/No 112534.
[28] G. Masetti, L. Robol, S. Chiaradonna, and F. Di Giandomenico, *Stochastic evaluation of large interdependent composed models through Kronecker algebra and exponential sums*, in Application and theory of Petri nets and concurrency. 40th international conference, PETRI NETS 2019, Aachen, Germany, June 23–28, 2019, Proceeding, Cham: Springer, 2019, pp. 47–66, https://doi.org/10.1007/978-3-030-21571-2_3, hdl.handle.net/11568/1000240.
[29] H.-D. Meyer, F. Gatti, and G. A. Worth, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, Weinheim: Wiley-VCH, 2009.
[30] R. B. Morgan, *Gmres with deflated restarting*, SIAM Journal on Scientific Computing, 24 (2002), pp. 20–37.
[31] Y. Nakatsukasa and J. A. Tropp, *Fast and Accurate Randomized Algorithms for Linear Systems and Eigenvalue Problems*, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 1183–1214, https://doi.org/10.1137/23M1565413.
[32] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.
[33] I. V. e. a. Oseledets, *GitHub - oseledets/TT-Toolbox: The git repository for the TT-Toolbox — github.com*. https://github.com/oseledets/TT-Toolbox. [Accessed 14-08-2024].
[34] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Physical review letters, 75 (1995), p. 3537.
[35] D. Palitta and P. Kürschner, *On the convergence of Krylov methods with low-rank truncations*, Numer Algor, 88 (2021), pp. 1383–1417, https://doi.org/10.1007/s11075-021-01080-2.
[36] D. Palitta, M. Schweitzer, and V. Simoncini, *Sketched and truncated polynomial Krylov methods: Matrix Sylvester equations*, Math Comp, (2024), https://doi.org/10.1090/mcom/4002.
[37] D. Palitta, M. Schweitzer, and V. Simoncini, *Sketched and truncated polynomial Krylov methods: Evaluation of matrix functions*, Numerical Linear Algebra with Applications, 32 (2025), https://doi.org/https://doi.org/10.1002/nla.2596.
[38] M. Röhrig-Zöllner, M. J. Becklas, J. Thies, and A. Basermann, *Performance of linear solvers in tensor-train format on current multicore architectures*, arXiv preprint arXiv:2312.08006, (2023).
[39] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
[40] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.
[41] Y. Saad and K. Wu, *DQGMRES: a Direct Quasi-minimal Residual Algorithm Based on Incomplete Orthogonalization*, Numerical Linear Algebra with Applications, 3 (1996), pp. 329–343, https://doi.org/10.1002/(SICI)1099-1506(199607/08)3:4⟨329::AID-NLA86⟩3.0.CO;2-8.
[42] T. Sarlós, *Improved approximation algorithms for large matrices via random projections*, in 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 143–152.
[43] T. Shi, M. Ruth, and A. Townsend, *Parallel algorithms for computing the tensor-train decomposition*, SIAM J. Sci. Comput., 45 (2023), pp. C101–C130, https://doi.org/10.1137/21M146079X.

[44] V. Simoncini and Y. Hao, *Analysis of the Truncated Conjugate Gradient Method for Linear Matrix Equations*, SIAM Journal on Matrix Analysis and Applications, 44 (2023), pp. 359–381, https://doi.org/10.1137/22M147880X.

[45] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM Journal on Scientific Computing, 25 (2003), pp. 454–477.

[46] I. H. Sloan and H. Woźniakowski, *When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals?*, Journal of Complexity, 14 (1998), pp. 1–33, https://doi.org/10.1006/jcom.1997.0463.

[47] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, *Low-rank tucker approximation of a tensor from streaming data*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1123–1150.

[48] X. Wang and I. H. Sloan, *Why Are High-Dimensional Finance Problems Often of Low Effective Dimension?*, SIAM Journal on Scientific Computing, 27 (2005), pp. 159–183, https://doi.org/10.1137/S1064827503429429.

[49] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014), pp. 1–157.