Streaming Tensor Train Approximation

Daniel Kressner\textsuperscript{1}, Bart Vandereycken\textsuperscript{2}, and Rik Voorhaar\textsuperscript{2}

\textsuperscript{1}Institute of Mathematics, EPFL, CH-1015 Lausanne, Switzerland
\textsuperscript{2}Section of Mathematics, University of Geneva, CH-1205 Geneva, Switzerland

Abstract

Tensor trains are a versatile tool to compress and work with high-dimensional data and functions. In this work we introduce the Streaming Tensor Train Approximation (STTA), a new class of algorithms for approximating a given tensor $T$ in the tensor train format. STTA accesses $T$ exclusively via two-sided random sketches of the original data, making it streamable and easy to implement in parallel – unlike existing deterministic and randomized tensor train approximations. This property also allows STTA to conveniently leverage structure in $T$, such as sparsity and various low-rank tensor formats, as well as linear combinations thereof. When Gaussian random matrices are used for sketching, STTA is admissible to an analysis that builds and extends upon existing results on the generalized Nyström approximation for matrices. Our results show that STTA can be expected to attain a nearly optimal approximation error if the sizes of the sketches are suitably chosen. A range of numerical experiments illustrates the performance of STTA compared to existing deterministic and randomized approaches.

1 Introduction

This work proposes and analyzes a new randomized algorithm for compressing a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ of order $d$ in the so-called tensor train (TT) format \cite{Oseledets2011}. The TT format (also called matrix product state – MPS – in physics \cite{Schollwock2011}) represents a tensor through contractions of third-order tensors, the so-called TT cores. Offering a number of key advantages, the TT format belongs to the most popular low-rank tensor formats. In particular, its multilinear structure and its close connection to low-rank matrix factorizations allow one to leverage tools from (numerical) linear algebra, such as the singular value decomposition (SVD), in the design and analysis of algorithms. During the last decade, the TT format has shown its utility in a wide variety of applications in scientific computing and data analysis; see \cite{GrasedyckBachmayrKhoromskaiaHackbuschUoschmajewVandereycken2016, Khoromskaia2018, Hackbusch2019} for an overview.

The randomized SVD, called the Halko–Martinsson–Tropp (HMT) method \cite{Halko2011} in the following, is a wildly popular tool for obtaining a low-rank approximation to a matrix $A \in \mathbb{R}^{m \times n}$. It proceeds by first generating a (random) dimension reduction matrix (DRM) $X \in \mathbb{R}^{n \times r}$ with $r \ll n$. Then the product $AX$ is computed and its columns are orthonormalized by a QR factorization, which we denote by $Q = \text{orth}(AX) \in \mathbb{R}^{n \times r}$. A low-rank approximation of $A$ is obtained by setting $A \approx \hat{A}_{\text{HMT}} = Q(Q^\top A)$. While very simple and effective, the HMT method is not streamable in the sense of \cite{ClarksonWoodruff2009} since forming $Q^\top A$ requires two passes over the input data $A$. Although the sketch $AX$ is linear in $A$, which makes it cheap to update when $A$ undergoes a change of the form $A \rightarrow A + B$, this is no longer true for $Q$ and $Q^\top A$. The generalized Nyström (GN) method, originally also devised in \cite{ClarksonWoodruff2009}, mitigates this issue by performing a two-sided sketch. First, it generates two DRMs $X \in \mathbb{R}^{n \times r}$ and $Y \in \mathbb{R}^{m \times (r + \ell)}$ for some $\ell \geq 1$ such that $r + \ell \ll m$. After computing the sketches $AX$, $Y^\top A$, $Y^\top AX$, one obtains a low-rank approximation by setting

$$A \approx \hat{A}_{\text{GN}} := AX(Y^\top AX)^{\dagger}Y^\top A.$$  \hspace{1cm} (1)

Here $(\cdot)^\dagger$ denotes the Moore–Penrose pseudoinverse by which multiplying is mathematically equivalent to solving a linear least-squares problem. As all three sketches are linear in $A$ and they dominate the cost, the GN method is streamable. Moreover, the computation of the sketches can be carried out such that it requires only one pass over the data.

In this work, we present a streaming TT approximation (STTA) – an extension of the GN method to TTs. Our algorithm is streamable and requires only one pass over the data. Moreover, unlike most existing TT approximation methods, STTA parallelizes naturally with respect to the tensor modes, an attractive feature for high-order tensors. Note that the STTA method itself does not make any assumptions on the type of DRMs. For the case of Gaussian random DRMs, we will derive an upper bound on the expected error that generalizes
the matrix results from \cite{Tropp2017, Nakatsu19}. In particular, we prove that the expected error is quasi-optimal and close to the one obtained via TT-SVD using only moderate oversampling.

Several different randomized approaches for TT approximation have been proposed in the literature. A straightforward approach is to replace every truncated SVD inside the TT-SVD algorithm, the standard method for TT approximation \cite{Oseledets2011}, by the HMT method. This algorithm, called TT-HMT in the following, is analyzed for Gaussian random DRMs in \cite{Huber2017, Wolf2019} and significant savings for a (very) sparse tensor \( T \) are reported. In \cite{Che2019}, the TT-HMT method is presented as well, together with an adaptive variant. In \cite{Alger2020}, another approach is presented that is well suited for compressing high-order derivative tensors. This variant accesses the tensor through evaluations of the corresponding multilinear map and uses random Khatri–Rao product DRMs, that is, (columnwise) Khatri–Rao products of Gaussian random matrices instead of unstructured Gaussian random matrices.

Neither the TT-SVD algorithm nor any of the variants mentioned above is streamable or parallel in the modes; nestedness relations among the low-rank matrix factorizations constituting the TT format represent a major obstacle for doing so. The parallel streaming TT sketching (PSTT) algorithms described in \cite{Shi2021} overcome this obstacle by first applying the HMT method to all relevant unfoldings of \( T \) in parallel and restoring nestedness a posteriori, without losing any of the favorable approximation properties. The work by \cite{Shi2021} also discusses the use of Khatri–Rao product DRMs and proposes variants that require only one pass over the input data \( T \). The PSTT algorithm and its variants inherit from the HMT method the need to compute (large) QR decompositions, which cannot be cheaply updated after a linear update of the data as explained above.

The work by \cite{Daas2021} focuses on TT rounding, that is, the subsequent compression of a tensor that is already in TT format. The standard algorithm for TT rounding is essentially the TT-SVD and thus requires to first successively orthogonalize the TT cores of the input TT format, a costly step that prevents from attaining any significant savings when using the HMT method. An algorithm called “Randomize-then-Orthogonalize” in \cite{Daas2021} circumvents this orthogonalization step by applying the HMT method to unfoldings of the full tensor and leveraging the TT format through the use of a TT DRM, that is, a TT with Gaussian random TT cores; see also \cite{Ma2022}. A two-sided variant that uses the GN method with two TT DRMs instead of the HMT method is also presented. None of the algorithms presented in \cite{Daas2021} are easy to parallelize with respect to the modes. Also, the formulation of the algorithms does not seem to allow for streaming and requires more than one pass over the input tensor.

The STTA method proposed in this work can be viewed as a streamable, two-sided variant of the PSST algorithm from \cite{Shi2021}. Besides two-sided sketches of the input tensor, no other large-scale computation needs to be carried out in STTA. In particular, the QR decompositions of the PSST algorithm are avoided, which makes it cheaper to carry out updates, avoids the storage of large intermediate matrices, and allows one to easily exploit sparsity and other structure in the input tensor. As we will explain in Section 1.2, a particular combination of the STTA method with two TT DRMs becomes mathematically equivalent to the two-sided variant \cite{Daas2021}.

An important advantage of the one-pass nature of STTA is that it can be efficiently used to compress (or round) a structured tensor that is given in some data sparse format. For example, we explain in Section 4 in some detail the implementation for compressing Tucker, CP, and sparse tensors. Furthermore, thanks to its streamability, it is straightforward to apply STTA to a sum of all these different structures. We are not aware of an existing randomized method for which all these features have been highlighted.

1.1 Notation and preliminaries

We first recall important properties of the tensor train (TT) format that are needed throughout the paper. For more details and proofs, we refer to \cite{Bachmayr2016, Hackbusch19}.

Given a matrix \( A \) or a tensor \( T \), its entries will be indexed as \( [i_1, \ldots, i_d] \). Furthermore, when an index \( i_\mu \) is denoted as \( ; \), it represents all the entries for the \( \mu \)-th mode and will result in a slice of the matrix or tensor.

A tensor \( T \) of size \( n_1 \times n_2 \times \cdots \times n_d \) in the TT format if it can be written element-wise as

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

The third-order tensors \( C_\mu \) of size \( r_{\mu-1} \times n_\mu \times r_\mu \) are the TT cores (where \( r_0 = r_d = 1 \)). Using the matrix \( C_\mu[; \ell_\mu, ;] \) of size \( r_{\mu-1} \times r_\mu \), the relation (2) 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:

\[
T[i_1, \ldots, i_d] = C_1[i_1, ;] C_2[; i_2, ;] \cdots C_d[; i_d, 1].
\]
The TT format defines a specific way to contract the tensors \( C_\mu \) which can be depicted graphically using a tensor diagram. In particular, the contractions are represented by the diagram

\[
\begin{array}{cccc}
T & \Rightarrow & C_1 & \cdots & C_d \\
& \downarrow & r_1 & \cdots & r_{d-1} \\
n_1 & \cdots & n_d & \downarrow & n_1 & \cdots & n_d
\end{array}
\]

(3)

The tuple \((r_1, \ldots, r_{d-1})\) is called the TT representation rank of the TT defined in (2) and it determines the complexity of working with a TT. For instance, storing a tensor in TT format requires to store the \(O(dnr^2)\) entries of its TT cores, where \(n := \max\mu(n_\mu)\) and \(r := \max\mu(r_\mu)\). Any tensor can trivially be written in the TT format by choosing the TT representation ranks sufficiently large.

The TT representation rank of a particular tensor \( T \) is by no means unique but there exists an (entry-wise) minimal value which is called the TT rank of \( T \). The minimal value for \( r_\mu \), equals the matrix rank of the \( \mu \)th unfolding \( T \), denoted as \( T^{\leq \mu} \). The unfolding \( T^{\leq \mu} \) is one of the many ways to matricize a tensor; it is a matrix of size \((n_1 \cdots n_\mu) \times (n_{\mu+1} \cdots n_d)\) obtained from merging the first \( \mu \) modes of \( T \) into row indices and the last \( d - \mu \) modes into column indices.

In the rest of the paper will not distinguish between TT rank and TT representation rank and simply call \((r_1, \ldots, r_{d-1})\) the TT rank of the tensor \( T \) once the relation (2) is satisfied for some cores \( C_\mu \).

When splitting the tensor diagram of a TT with cores \( C_\mu \) into two by cutting the edge between modes \( \mu \) and \( \mu + 1 \), one obtains two subdiagrams. Each of the two subdiagrams describes a partial contraction of the cores representing a tensor of size \(n_1 \times \cdots \times n_\mu \times r_\mu \) or \(n_{\mu+1} \times \cdots \times n_d \times r_\mu\). We denote the \( \mu \)th and \((d - \mu)\)th unfoldings of these tensors by

\[
C_{\leq \mu} \in \mathbb{R}^{(n_1 \cdots n_\mu) \times r_\mu} \quad \text{and} \quad C_{>\mu} \in \mathbb{R}^{(n_{\mu+1} \cdots n_d) \times r_\mu},
\]

which are sometimes called interface matrices. The TT relation (2) implies the (low-rank) factorization \( T^{\leq \mu} = C_{\leq \mu} C_{>\mu} \). It also implies nestedness relations among the interface matrices \( C_{\leq \mu}, C_{>\mu} \) for different \( \mu \):

\[
C_{\leq \mu} = (C_{\leq \mu-1} \otimes I) C_{\mu}^L \quad \text{and} \quad C_{>\mu} = (I \otimes C_{>\mu+1}) C_{\mu}^R.
\]

(4)

Here, \( \otimes \) denotes the (right) Kronecker product of two matrices, and the matrices \( C_{\mu}^L \) and \( C_{\mu}^R \) are convenient notations for the so-called left and right unfoldings of \( C_\mu \), respectively.

We denote by \( \|T\|_F = \left( \sum_{i_1, \ldots, i_d} T[i_1, \ldots, i_d]^2 \right)^{1/2} \) the Frobenius norm of the tensor \( T \). Because of its elementwise definition, it coincides with the Frobenius norm of any matricization of \( T \); in particular \( \|T\|_F^2 = \|T^{\leq \mu}\|_F^2 \) for every \( \mu \).

2 Randomized low-rank approximations

Before describing and analyzing our algorithms for TT, we briefly review the theory of randomized methods for finding a low-rank approximation to a matrix \( A \in \mathbb{R}^{m \times n} \).

As explained in the introduction, the HMT method constructs the approximation

\[
\hat{A}_{\text{HMT}} = Q(Q^\top A),
\]

(5)

with \( Q = \text{orth}(AX) \in \mathbb{R}^{n \times r} \) for a (random) DRM \( X \in \mathbb{R}^{n \times r} \), \( r < n \). We let \( P_{AX} = AX(AX)^\dagger \) denote the orthogonal projection onto the range of \( AX \). Because of \( P_{AX} = QQ^\top \), we can equivalently write (5) as

\[
\hat{A}_{\text{HMT}} = P_{AX} A.
\]

(6)

This approximation usually compares well with a best rank \( \hat{r} \) approximation \( A_\hat{r} \) of slightly smaller rank \( \hat{r} < r - 1 \). More specifically, when taking expectation with respect to a random Gaussian matrix \( X \) one has

\[
\mathbb{E} \| \hat{A}_{\text{HMT}} - A \|_F^2 \leq \left( 1 + \frac{\hat{r}}{r - \hat{r} - 1} \right) \|A_\hat{r} - A\|_F^2;
\]

(7)

see Theorem 10.5 in [Halko et al., 2011].

\(^1\)A random matrix is called Gaussian if its entries are iid standard normal random variables.
The generalized Nyström (GN) method uses two DRMs $X \in \mathbb{R}^{n \times r}$ and $Y \in \mathbb{R}^{m \times (r+\ell)}$, for some $\ell \geq 1$ such that $r + \ell < m$, to construct the approximation
\[
\hat{A}_{\text{GN}} = AX(Y^\top AX)^{\dagger}Y^\top A.
\] (8)

Again, this can be expressed in terms of a projection. Given two matrices $B$ and $C$ with the same number of rows, we define the oblique projector $\mathcal{P}_{B,C} = B(C^\top B)^{\dagger}C^\top$. It directly follows from (8) that
\[
\hat{A}_{\text{GN}} = \mathcal{P}_{AX,Y} A.
\] (9)

The fact that $\mathcal{P}_{AX,Y}$ is no longer orthogonal implies that the GN approximation is (slightly) worse than the HMT approximation for the same $X$. In terms of the expected error the following bound holds.

**Theorem 2.1** ([Tropp et al., 2017, Theorem 4.3]). Consider random Gaussian matrices $X$ and $Y$ of size $n \times r$ and $n \times (r + \ell)$, respectively, with $\ell > 0$. Then the expected approximation error of (8) is bounded by
\[
\mathbb{E} \| \hat{A} - A \|^2_F \leq \left( 1 + \frac{r}{\ell - 1} \right) \left( 1 + \frac{\hat{r}}{r - \hat{r} - 1} \right) \| A - A \|^2_F,
\] (10)

where $A$ is any best rank $\hat{r}$ approximation of $A$ with $\hat{r} < r - 1$.

**Remark 2.2.** The formulation of Theorem 2.1 assumes that $Y$ has more columns than $X$. Since
\[
\| (I - \mathcal{P}_{AX,Y}) A \|_F = \| (I - \mathcal{P}_{AX,Y}) A \|^\top \|_F = \| (I - \mathcal{P}_{A^\top Y,X}) A \|^\top \|_F,
\] (11)

the bound also holds in the reverse situation, that is, when $X$ has size $n \times (r + \ell)$ and $Y$ has size $n \times r$.

Let us emphasize again that the GN method is streamable and one-pass. It consists of two steps:

1. Sketch $AX$, $Y^\top A$, $Y^\top AX$.
2. Assemble $\hat{A} = AX(Y^\top AX)^{\dagger} Y^\top A$.

The first step is linear in $A$, and the sketches are cheap to store or communicate if $r \ll \min(m, n)$. Moreover, this step typically dominates the overall computational effort, requiring $O(rmn)$ operations for dense matrices, whereas the second step only requires $O((r + \ell)m^2)$ operations; see [Nakatsukasa, 2020] for implementation aspects.

## 3 Streaming TT approximation (STTA)

We now explain our streamable TT approximation algorithm for a given tensor $\mathcal{T}$ of size $n_1 \times \cdots \times n_d$. The idea is to sketch every unfolding $\mathcal{T}^{\leq \mu} \in \mathbb{R}^{(n_\mu+\cdots+n_d) \times (n_\mu+\cdots+n_1)}$ using several dimension reduction matrices (DRMs) $X_\mu$, $\Psi_\mu$ of matching sizes:
\[
X_\mu \text{ is a ‘right’ DRM of size } (n_{\mu+1} \cdots n_d) \times r_\mu^R \text{ for } 1 \leq \mu \leq d - 1;
\]
\[
Y_\mu \text{ is a ‘left’ DRM of size } (n_1 \cdots n_{\mu}) \times r_\mu^L \text{ for } 1 \leq \mu \leq d - 1.
\] (12)

We require that either $r_\mu^R < r_\mu^L - 1$ for all $\mu$ or $r_\mu^L < r_\mu^R - 1$ for all $\mu$. Our approximation will have TT rank $r_\mu := \min(r_\mu^L, r_\mu^R)$ for $\mu = 1, \ldots, d - 1$ and is obtained from the following sketches:
\[
\Psi_\mu^L = (Y_{\mu-1}^\top \otimes I) \mathcal{T}^{\leq \mu} X_\mu \text{ of size } (r_{\mu-1} n_{\mu}) \times r_\mu^R \text{ for } 1 \leq \mu \leq d;
\]
\[
\Omega_\mu = Y_{\mu}^\top \mathcal{T}^{\leq \mu} X_\mu \text{ of size } r_\mu^L \times r_\mu^R \text{ for } 1 \leq \mu \leq d - 1.
\] (13)

For simplicity we use the convention that $Y_0 = X_d = 1$, so that (13) remains valid for $\mu = 1$ and $\mu = d$. Recall that $\Psi_\mu^L$ denotes the left unfolding of the $r_{\mu-1} \times n_\mu \times r_\mu^R$ tensor $\Psi_\mu$. In terms of tensor diagrams, the relations (13) take the following form:
\[
\begin{align*}
\Psi_{\mu-1}^{r_{\mu-1}^L} & = r_{\mu-1}^L Y_{\mu-1} : n_1 \vdots n_{\mu-1} \rightarrow n_\mu \vdots n_d X_{\mu}^r, \\
\Omega_\mu^{r_\mu^L} & = r_\mu^L Y_\mu : n_1 \vdots n_\mu \rightarrow n_d X_{\mu}^r.
\end{align*}
\] (14)
Using these sketches, we construct an approximation \( \tilde{T} \) as follows:

\[
\tilde{T} := \Psi_1 r_1^R \Omega_1^t \Psi_2 r_2^R \Omega_2^t \cdots \Omega_{d-1}^t r_{d-1}^R \Psi_d
\]  

(15)

To arrive at a tensor diagram that takes the form of a TT as in [3], one needs to contract each \( \Omega^t_\mu \) with an adjacent order-three tensor. When \( r_\mu^R < r_\mu^L \), it is preferable to choose the tensor to the right, because otherwise the (representation) ranks are unnecessarily high. More specifically, we obtain a TT representation for \( \tilde{T} \) by defining each \( r_{\mu-1}^R \times n_\mu \times r_\mu^R \) core \( C_\mu \) via its right unfolding

\[
C_\mu^R := \begin{cases} 
\Psi_1, & \mu = 1, \\
\Omega^t_{\mu-1} \Psi^R_\mu, & \mu > 1.
\end{cases}
\]  

(16)

Assuming that \( \Omega_\mu \) has full (column) rank, we can equivalently obtain \( C_\mu^R \) as the solution of the linear least-squares problem

\[
\min_X \| \Omega_{\mu-1} X - \Psi_\mu^R \|_F^2.
\]  

(17)

Especially when \( \Omega_\mu \) is not well-conditioned, care needs to be taken when solving this problem numerically; see also [Nakatsukasa, 2020, §4.1]. In our implementation, we use the LAPACK routine \texttt{gelsd} which utilizes a (truncated) SVD of \( \Omega_{\mu-1} \) and discards singular values smaller than \( \epsilon_{\text{mach}} \| \Omega_{\mu-1} \|_2 \), where \( \epsilon_{\text{mach}} \) denotes machine precision.

When \( r_\mu^L > r_\mu^R \) it is preferable to contract \( \Omega^t_\mu \) with the tensor on the left, that is, the TT core \( C_\mu \) is an \( r_{\mu-1}^L \times n_\mu \times r_\mu^L \) tensor defined via

\[
C_\mu^L := \begin{cases} 
\Psi_\mu^L \Omega_\mu^t, & \mu < d \\
\Psi_d, & \mu = d.
\end{cases}
\]  

(18)

If \( \Omega_\mu \) has full row rank, this is equivalent to solving the linear least-squares problem \( \min_X \| X \Omega_\mu - \Psi_\mu^L \|_F^2 \).

Like the matrix case, our streaming tensor sketch algorithm (STTA) can be split into two subroutines. First we sketch the tensor \( T \), and then we assemble the TT from the sketches. These two routines are summarized below in Algorithms 3.1 and 3.2. For convenience we only state the version for \( r_\mu^R < r_\mu^L \), which produces a TT of rank \( (r_1^R, \ldots, r_{d-1}^R) \); the other version is analogous.

**Algorithm 3.1** STTA: sketch

**Input:** Tensor \( T \) of shape \( n_1 \times \cdots \times n_d \); Right DRMs \( X_\mu \) of shape \( (n_{\mu+1} \cdots n_d) \times r_\mu^R \) and left DRMs \( Y_\mu \) of shape \( (n_1 \cdots n_\mu) \times r_\mu^L \) for \( \mu = 1 \) to \( d-1 \) with \( r_\mu^R < r_\mu^L \).

**Output:** Sketches \( \Psi_\mu \) of shape \( r_{\mu-1}^L \times n_\mu \times r_\mu^R \) for \( 1 \leq \mu \leq d \); Sketches \( \Omega_\mu \) of shape \( r_\mu^L \times r_\mu^R \) for \( 1 \leq \mu \leq d-1 \).

1: for \( \mu = 1 \) to \( d \) do
2: \( T^{\leq \mu} \leftarrow \text{reshape}(T, (n_1 \cdots n_\mu) \times (n_{\mu+1} \cdots n_d)) \)
3: if \( \mu = 1 \) then
4: \( \Psi_1 \leftarrow T^{\leq 1} X_1 \)
5: else if \( \mu < d \) then
6: \( \Psi_\mu \leftarrow (Y_{\mu-1}^T \otimes I_{n_\mu}) T^{\leq \mu} X_\mu \)
7: else
8: \( \Psi_d \leftarrow (Y_{d-1}^T \otimes I_{n_d}) T^{\leq d} \)
9: end if
10: if \( \mu < d \) then
11: \( \Omega_\mu \leftarrow Y_\mu^T T^{\leq \mu} X_\mu \)
12: end if
13: end for

**DRMs:** Algorithm 3.2 takes \( 2d-2 \) right and left DRMs as input. The simplest choice for DRMs are Gaussian random matrices, which lead to matrices \( X_\mu \) and \( Y_\mu \) that are independent with respect to \( \mu \). When \( T \) is itself in low-rank format (TT, CP, and Tucker formats – see Sections 4.3 and 4.4), it is computationally advantageous to use a single TT DRM (a TT with Gaussian random TT cores) to define all right DRMs, and similarly for
Algorithm 3.2 STTA: assemble

Input: Sketches $\Psi_\mu$ of shape $r_{\mu-1}^L \times n_\mu \times r_\mu^R$ for $1 \leq \mu \leq d$; Sketches $\Omega_\mu$ of shape $r_\mu^L \times r_\mu^R$ for $1 \leq \mu \leq d - 1$.

Output: Cores $C_\mu$ of shape $r_{\mu-1}^L \times n_\mu \times r_\mu^R$ for $1 \leq \mu \leq d$ of a tensor train approximation of $\mathcal{T}$.

1: $C_1 \leftarrow \text{reshape}(\Psi_1, 1 \times n_1 \times r_1^R)$
2: for $\mu = 2$ to $d - 1$ do
3: \begin{verbatim}
4: \hspace{1em} $B_\mu^R \leftarrow \text{lstsq}(\Omega_{\mu-1}, \Psi_\mu^R)$ \hspace{1em} // Stable least-square solver for $\Omega_{\mu-1}^T \Psi_\mu^R$ as in (17)
5: \hspace{1em} $C_\mu \leftarrow \text{reshape}(B_\mu^R \cdot r_{\mu-1}^R \times n_\mu \times r_\mu^R)$
6: end for
7: for all left DRMs. In turn, the right DRMs $X_1, \ldots, X_{d-1}$ and the left DRMs $Y_1, \ldots, Y_{d-1}$ are no longer independent for different $\mu$ but coupled through the nestedness relations (4); see Definition 3.3 for details.

For the special case of TT DRMs, the approximation returned by STTA is mathematically equivalent to the one produced by Algorithm 3.3 of [Daas et al., 2021]; see also Section 4.2.

**Streaming:** Because Algorithm 3.1 is linear in $\mathcal{T}$, it is indeed streamable and one-pass. Suppose that $\mathcal{T}$ is decomposed as a sum $\mathcal{T}_1 + \cdots + \mathcal{T}_k$ and we compute the sketches $\Psi_\mu,i$, $\Omega_\mu,i$ of each summand $\mathcal{T}_i$. The sketch of the sum is then the sum of the sketches: $\Psi_\mu = \sum_{i=1}^k \Psi_\mu,i$, and similarly for $\Omega_\mu$. These sketches can be performed in a data-parallel fashion, and only one round of communication is required, involving a total $O(mr^2dk)$ of data making it very suitable for a distributed setting.

**Parallelism:** In addition to the data-parallelism due to streaming, we note that all for loops in both Algorithms 3.1 and 3.2 can be executed in parallel. This is also true for TT DRMs at the expense of some redundant computations. Moreover, the most expensive operations in Algorithm 3.2 are matrix multiplications, which enjoy a high degree of parallelism themselves.

**Adaptive rank and block sketching:** We can adaptively increase the rank of the approximation by performing a new sketch. Suppose we have computed a rank-$r$ approximation of $\mathcal{T}$ using the sketches $\Psi_\mu,1$, $\Omega_\mu,1$, and DRMs $X_\mu,1$, $Y_\mu,1$ with $1 \leq \mu \leq d$. We can then compute a rank-$(r+r')$ approximation by first generating a second pair of DRMs $X_\mu,2$, $Y_\mu,2$ of rank $r'$. Sketching $\mathcal{T}$ with the rank-$(r+r')$ DRMs $X_\mu = [X_\mu,1 X_\mu,2]$ and $Y_\mu = [Y_\mu,1 Y_\mu,2]$ is mathematically equivalent to performing the following block-wise sketches:

\[
\Psi_\mu = \begin{bmatrix}
((Y_\mu^1)^\top \otimes I)\mathcal{T}^{\leq \mu} X_\mu^1 & ((Y_\mu^1)^\top \otimes I)\mathcal{T}^{\leq \mu} X_\mu^2 \\
((Y_\mu^2)^\top \otimes I)\mathcal{T}^{\leq \mu} X_\mu^1 & ((Y_\mu^2)^\top \otimes I)\mathcal{T}^{\leq \mu} X_\mu^2
\end{bmatrix}
\]

\[
\Omega_\mu = \begin{bmatrix}
(Y_\mu^1)^\top \mathcal{T}^{\leq \mu} X_\mu^1 & (Y_\mu^1)^\top \mathcal{T}^{\leq \mu} X_\mu^2 \\
(Y_\mu^2)^\top \mathcal{T}^{\leq \mu} X_\mu^1 & (Y_\mu^2)^\top \mathcal{T}^{\leq \mu} X_\mu^2
\end{bmatrix}.
\] (19)

Note that in this scenario the top-left blocks of both sketches are $\Psi_\mu$ and $\Omega_\mu$, respectively, which were both already computed. This means that first computing a rank-$r$ sketch, and then computing a rank-$(r+r')$ sketch is not more expensive than directly computing a rank-$(r+r')$ sketch. Furthermore, (19) gives a block algorithm for computing the sketch, which further increases the available parallelism.

### 3.1 Error analysis

For the error analysis of STTA, it is beneficial to express the obtained approximation in terms of projectors, like for the HMT method in (6) and for the GN method in (9). For this purpose, let us define the oblique projectors

\[
P_\mu := P_{\mathcal{T}^{\leq \mu} X_\mu Y_\mu} = \mathcal{T}^{\leq \mu} X_\mu (Y_\mu^\top \mathcal{T}^{\leq \mu} X_\mu)^\top Y_\mu = \mathcal{T}^{\leq \mu} X_\mu \Omega_\mu^\top Y_\mu^\top.
\] (20)

We claim that the approximation (15) of STTA satisfies

\[
\tilde{\mathcal{T}}^{\leq d-1} = (P_1 \otimes I) \cdots (P_{d-2} \otimes I) P_{d-1} \mathcal{T}^{\leq d-1},
\] (21)

where the size of the identity matrices $I$ is such that all matrix products are well defined. In terms of a tensor diagram, this expression takes the form

```
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
P_1 \\
\vdots \\
P_{d-2} \\
P_{d-1}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
```

(22)

This is assuming that the DRMs are not communicated. In practice, the DRMs are always pseudo-random so that they can be efficiently generated locally on each node using a seeded method. This will be discussed further in Section 4.
To show (21), we first note that the definitions (13) and (20) imply
\[
\left( [\Psi_1 \Omega_1^\dagger] \otimes I \right) \Psi_2 = \left( [T^{\leq 1} X_1 (Y_1^\dagger T^{\leq 1} X_1)^\dagger] \otimes I \right) T^{\leq 2} X_2 = (P_1 \otimes I) T^{\leq 2} X_2,
\]
which corresponds to the diagram

Comparing with (15), this shows that the left part of the network representing \( \tilde{T} \) can be replaced by \( P_1 \). More generally, we have the relation
\[
\left( [T^{\leq \mu} X_\mu \Omega_\mu^\dagger] \otimes I \right) \Psi_{\mu + 1} = \left( [T^{\leq \mu} X_\mu (Y_\mu^\dagger T^{\leq \mu} X_\mu)^\dagger] \otimes I \right) T^{\leq \mu + 1} X_{\mu + 1} = (P_\mu \otimes I) T^{\leq \mu + 1} X_{\mu + 1},
\]
which allows us to successively replace, from the left to the right, the network representing \( \tilde{T} \) with the projectors \( P_\mu \), until we arrive at

In turn, we obtain the network (22) as claimed.

**Proposition 3.1.** The approximation \( \tilde{T} \) returned by STTA satisfies
\[
\| \tilde{T} - T \|_F \leq \sum_{\mu=1}^{d-1} \left\| \prod_{\alpha<\mu} (P_\alpha \otimes I)(I - P_\mu) T^{\leq \mu} \right\|_F. \tag{23}
\]

**Proof.** Using (21), we have that
\[
\| T - \tilde{T} \|_F = \left\| [I - (P_1 \otimes I) \cdots (P_{d-2} \otimes I) P_{d-1}] T^{\leq d-1} \right\|_F \leq \left\| [I - (P_1 \otimes I) \cdots (P_{d-2} \otimes I) P_{d-1}] T^{\leq d-1} \right\|_F + \left\| (P_1 \otimes I) \cdots (P_{d-2} \otimes I) [I - P_{d-1}] T^{\leq d-1} \right\|_F.
\]

We can now continue the argument inductively since
\[
\| [I - (P_1 \otimes I) \cdots (P_{d-2} \otimes I)] T^{\leq d-1} \|_F = \left\| [I - (P_1 \otimes I) \cdots P_{d-2}] T^{\leq d-2} \right\|_F.
\]

**Gaussian DRMs.** We now focus on the particular case when Gaussian DRMs \( X_\mu, Y_\mu \) are used and extend the bounds from Section 2 to the STTA algorithm.

\[\text{Note that the non-commuting product } \prod_{\alpha<\mu} (P_\alpha \otimes I) \text{ in (23) is evaluated in natural increasing order with respect to } \alpha. \text{ Furthermore, for } \mu = 1 \text{ the product is empty and corresponds to the identity matrix.}\]
Theorem 3.2. Suppose that the DRMs $X_\mu, Y_\mu$ defined in [12] are independent standard Gaussian matrices with $r_\mu < r_\mu^R - 1$ and that the rank of $T^{\leq \mu}$ is not smaller than $r_\mu^R$ for $\mu = 1, \ldots, d - 1$. Then for any $(\tilde r_1, \ldots, \tilde r_{d - 1})$ such that $\tilde r_\mu < r_\mu^R - 1$, the STTA approximation $\tilde T$ of $T$ obtained from Algorithm 3.2 satisfies
\[
\mathbb{E} \left\| \tilde T - T \right\|_F \leq \sum_{\mu=1}^{d-1} \left[ \prod_{\alpha=1}^{\mu-1} c_\alpha \right] c'_\mu \sum_{k > \tilde r_\mu} \| \mathbf{c}(T^{\leq \mu}) \|^2 \leq \sum_{\mu=1}^{d-1} \left[ \prod_{\alpha=1}^{\mu-1} c_\alpha \right] c'_\mu \| \mathbf{T}_F - T \|_F, \tag{24}
\]
where $\mathbf{T}_F$ is any best TT approximation of $T$ of TT rank $(\tilde r_1, \ldots, \tilde r_d)$, and $c_\mu, c'_\mu$ are given by
\[
c_\mu := 1 + \frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1}, \quad c'_\mu := 1 + \frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1}. \tag{25}
\]
Proof. By the assumptions, both $T^{\leq \mu} X_\mu$ and $Y_\mu^T T^{\leq \mu} X_\mu$ have full column rank almost surely. We consider arbitrary $1 \leq \mu < d - 1$ and temporarily consider $X_\mu$ fixed (such that the full-rank condition holds). The QR decomposition $T^{\leq \mu} X_\mu = QR$ yields an invertible factor $R$ of size $r_\mu^R \times r_\mu^R$, which allows us to rewrite the definition (20) of $\mathbf{P}_\mu$ as follows:
\[
\mathbf{P}_\mu = T^{\leq \mu} X_\mu (Y_\mu^T T^{\leq \mu} X_\mu)^\dagger Y_\mu^T = Q R (Y_\mu^T Q R)^\dagger Y_\mu^T = Q (Y_\mu^T Q)^\dagger Y_\mu^T.
\]
We now complete the $r_\mu^R$ columns of $Q$ to a square orthogonal matrix $[Q \quad Q_\perp]$. Using $Q Q^T + Q_\perp Q_\perp^T = I$, we obtain for any matrix $A$ of size $(n_1 \cdot \ldots \cdot n_\mu) \times (n_{\mu+1} \cdot \ldots \cdot n_d)$ that
\[
\| \mathbf{P}_\mu A \|_F \leq \| (Y_\mu^T Q)^\dagger Y_\mu^T Q Q^T A \|_F + \| (Y_\mu^T Q)^\dagger Y_\mu^T Q_\perp Q_\perp^T A \|_F \leq \| A \|_F + \| (Y_\mu^T Q)^\dagger Y_\mu^T Q_\perp Q_\perp^T A \|_F.
\]
By the orthogonal invariance of Gaussian random matrices, $Z_1 := Y_\mu^T Q$ and $Z_2 := Y_\mu^T Q_\perp$ are independent Gaussian random matrices of size $r_\mu^R \times r_\mu^R$ and $r_\mu^R \times (n_1 \cdot \ldots \cdot n_\mu - r_\mu^R)$, respectively. This allows us to apply Propositions 10.1 and 10.2 from [Hackbusch et al., 2014] to obtain
\[
\mathbb{E}_{Z_1, Z_2} \| Z_1 Z_2 Q_\perp^T A \|_F \leq \sqrt{\mathbb{E}_{Z_1, Z_2} \| Z_1^2 Z_2^2 Q_\perp^2 \|_F} = \sqrt{\mathbb{E}_{Z_1} \| Z_1^2 \|_F} \| Q_\perp^T A \|_F = \sqrt{\frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1} \| Q_\perp^T A \|_F}.
\]
In summary, we have
\[
\mathbb{E}_{X_1, Y_1} \| \mathbf{P}_\mu A \|_F \leq c_\mu \| A \|_F.
\]
Now, by the law of total expectation, it holds that
\[
\mathbb{E}_{X_1, Y_1, \ldots, X_\mu, Y_\mu} \| \prod_{\alpha=1}^{\mu-1} (\mathbf{P}_\alpha \otimes I) (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F
\]
\[
= \mathbb{E}_{X_1, Y_1, \ldots, X_\mu, Y_\mu} \left[ \mathbb{E}_{X_1, Y_1} \| (\mathbf{P}_1 \otimes I) \prod_{\alpha=1}^{\mu-1} (\mathbf{P}_\alpha \otimes I) (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F \right] \leq c_1 \mathbb{E}_{X_1, Y_1, \ldots, X_\mu, Y_\mu} \left[ \| \prod_{\alpha=1}^{\mu-1} (\mathbf{P}_\alpha \otimes I) (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F \right],
\]
where we used that only $\mathbf{P}_1$ depends on $X_1, Y_1$. Continuing inductively we obtain
\[
\mathbb{E}_{X_1, Y_1, \ldots, X_\mu, Y_\mu} \| \prod_{\alpha=1}^{\mu-1} (\mathbf{P}_\alpha \otimes I) (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F \leq c_1 \cdots c_{\mu-1} \mathbb{E}_{X_1, Y_1} \| (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F. \tag{26}
\]
The last factor coincides with the GN approximation error, which – according to Theorem 2.1 – is bounded by
\[
\mathbb{E}_{X_1, Y_1} \| (I - \mathbf{P}_\mu) T^{\leq \mu} \|_F^2 \leq \left( 1 + \frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1} \right) \left( 1 + \frac{\tilde r_\mu}{r_\mu^L - \tilde r_\mu - 1} \right) \sum_{k > \tilde r_\mu} \sigma_k(T^{\leq \mu})^2 \leq (c'_\mu)^2 \sum_{k > \tilde r_\mu} \sigma_k(T^{\leq \mu})^2
\]
for any $\tilde r_\mu < r_\mu^R - 1$, where $\sigma_k(\cdot)$ denotes the $k$th largest singular value of a matrix. Finally, we apply Proposition 3.1 to obtain the first inequality of (24). The second inequality follows directly from the general bound
\[
\sum_{k > \tilde r_\mu} \sigma_k(T^{\leq \mu})^2 \leq \| T - \mathbf{T}_F \|_F^2; \tag{27}
\]
see, e.g., [Hackbusch, 2019] Thm. 11.6].
Observe that if one uses \( r^L_\mu = kr^R_\mu \) for a constant \( \kappa > 1 \) in Theorem 3.2, then \( c_\mu = O(1) \). As \( \kappa \) increases, \( c_\mu \) approaches one, which mitigates the exponential dependence on \( d \) in the constants.

The condition \( r^R_\mu < r^L_\mu - 1 \) for all \( \mu \) of Theorem 3.2 can be replaced by \( r^L_\mu < r^R_\mu - 1 \), using a modification of the proof that proceeds in the other direction, from ‘right to left’. The approximation bound (24) still holds but with the roles of \( r^L_\mu \) and \( r^R_\mu \) reversed in (25).

**Random TT DRMs.** The proof of Theorem 3.2 heavily relies on the DRMs being Gaussian. As mentioned before and demonstrated in [Daas et al., 2021, Ma and Solomonik, 2022], it can often be computationally beneficial to use TT DRMs, for which we now give a formal definition.

**Definition 3.3 (TT DRM).** Consider a tensor in TT format with random TT cores \( C_\mu \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) for which the entries are i.i.d. normal random variables with zero mean and variance \( 1/r^L_\mu \). Then left TT DRMs \( (Y_1, \ldots, Y_d) \), required as input by Algorithm 3.1, are defined via the (left) interface matrices

\[
Y_\mu := C_{\leq \mu} \in \mathbb{R}^{(n_1 \cdots n_d) \times r_\mu}, \quad \mu = 1, \ldots, d - 1,
\]

which satisfy the nestedness relations (1), that is, \( Y_\mu = (Y_{\mu-1} \otimes I)C^L_\mu \).

Analogously, right TT DRMs \( (X_1, \ldots, X_d) \) are defined through \( X_\mu := C_{> \mu} \) for another (indepedent) set of random TT cores \( C_\mu \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) with variance \( 1/r^L_\mu \).

The normalization of the variance in Definition 3.3 is chosen so that we have \( \mathbb{E} \|X_{C_\mu}^L\|_F^2 = \|X\|_F^2 \) for each TT core \( C_\mu \) and any fixed matrix \( X \) of appropriate size. This avoids numerical difficulties, such as over- or underflow, in particular when dealing with tensors of high order as in Section 5.7.

Proving an analogue to Theorem 3.2 for TT DRMs is difficult. As a first, preliminary result, we prove that the product \( \mathcal{P} = \mathcal{P}_1 \cdots \mathcal{P}_{d-2} \otimes I \mathcal{P}_{d-1} \) of left TT DRMs \( (Y_1, \ldots, Y_{d-1}) \), which produces the STTA approximation \( \tilde{T} = \mathcal{P}T \), is itself an oblique projector. This intriguing property does not hold for Gaussian DRMs.

**Lemma 3.4.** Suppose that \( r^L_\mu \geq r^R_\mu \) for all \( \mu \), and \( (Y_1, \ldots, Y_{d-1}) \) is a left TT DRM. If \( \text{rank}(T^{\leq \mu}) \geq r^R_\mu \) for all \( \mu \), then \( \mathcal{P} := \mathcal{P}_1 \cdots \mathcal{P}_{d-2} \otimes I \mathcal{P}_{d-1} \) is an oblique projector almost surely.

**Proof.** We first establish \( Y_\mu (Y_{\mu-1} \otimes I) = Y_\mu \) from the nestedness relations (1):

\[
Y_\mu (Y_{\mu-1} \otimes I) = (C^L_\mu)^T (Y_{\mu-1} \otimes I) [T^{\leq \mu-1} X_{\mu-1} (Y_{\mu-1}^T T^{\leq \mu-1} X_{\mu-1})^T] Y_{\mu-1} \otimes I
\]

\[
= (C^L_\mu)^T [Y_{\mu-1}^T T^{\leq \mu-1} X_{\mu-1} (Y_{\mu-1}^T T^{\leq \mu-1} X_{\mu-1})^T] Y_{\mu-1} \otimes I
\]

\[
= (C^L_\mu)^T [Y_{\mu-1}^T \otimes I] = Y_\mu.
\]

Here, we used that the \( r^L_\mu \times r^R_\mu \) matrix \( Y_{\mu-1}^T T^{\leq \mu-1} X_{\mu-1} \) has full row rank almost surely because of \( r^R_\mu \geq r^L_\mu \). It now follows that

\[
\mathcal{P}_\mu (\mathcal{P}_{\mu-1} \otimes I) = \mathcal{T}^{\leq \mu} X_{\mu} \mathcal{O}^T X_{\mu} \mathcal{P}_{\mu-1} \otimes I = \mathcal{T}^{\leq \mu} X_{\mu} \mathcal{O}^T \mathcal{P}_{\mu-1} \otimes I = \mathcal{P}_\mu.
\]

We now show that \( \mathcal{P}_1 \cdots \mathcal{P}_{d-2} \otimes I \mathcal{P}_{d-1} \) is a projector. For convenience, we omit writing the \( \otimes I \) factors.

Observe that by applying (28) multiple times, we get the identities

\[
\mathcal{P}_\mu \cdots \mathcal{P}_2 \mathcal{P}_1 = \mathcal{P}_\mu \cdots \mathcal{P}_3 \mathcal{P}_2 = \cdots = \mathcal{P}_\mu
\]

and

\[
(\mathcal{P}_\mu \cdots \mathcal{P}_2 \mathcal{P}_1)(\mathcal{P}_1 \cdots \mathcal{P}_2 \mathcal{P}_1 \cdots \mathcal{P}_\mu) = \mathcal{P}_\mu \cdots \mathcal{P}_3 \mathcal{P}_2 \cdots \mathcal{P}_\mu = \mathcal{P}_\mu \cdots \mathcal{P}_3 \mathcal{P}_2 \mathcal{P}_1 \cdots \mathcal{P}_\mu = \cdots = \mathcal{P}_\mu,
\]

which allow us to show that

\[
(\mathcal{P}_1 \cdots \mathcal{P}_\mu)(\mathcal{P}_1 \cdots \mathcal{P}_\mu) = \mathcal{P}_1 \cdots \mathcal{P}_\mu(\mathcal{P}_\mu \cdots \mathcal{P}_1 \mathcal{P}_1 \cdots \mathcal{P}_\mu) = (\mathcal{P}_1 \cdots \mathcal{P}_\mu),
\]

as required. \( \square \)

The assumption \( r^R_\mu \geq r^L_\mu \) can be replaced by \( r^L_\mu \geq r^R_\mu \). In this case, the STTA approximation satisfies \( \tilde{T} = \mathcal{P}T \) for an oblique projector \( \mathcal{P} \) associated with right TT DRMs \( (X_1, \ldots, X_{d-1}) \).
4 Sketching structured tensors

We recall that STTA requires the sketches $\Psi_{\mu}$, $\Omega_{\mu}$ defined in [14]. For a general order-$d$ tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the straightforward computation of these sketches requires a cost of $O(dn^d)$, which becomes excessive for large $d$. On the other hand, large-scale tensors usually have some structure that allows for faster sketching. In this section, we will explain how the STTA algorithm can be efficiently implemented for tensors with the following structures:

- sparse tensors;
- TT tensors (recompression);
- CP tensors;
- Tucker tensors.

Among these, only sparse tensors can be efficiently sketched using Gaussian DRMs; for the other formats we will instead use TT DRMs for sketching. Although our error bounds only apply to Gaussian DRMs, we will see that TT DRMs deliver similar performance.

Because of the linearity of sketching, we automatically obtain efficient algorithms for approximating linear combinations of any of these types of structured tensors. For example, one might be interested in sketching a sum of a sparse tensor and a TT.

4.1 Sketching sparse tensors

Suppose we want to approximate an $n_1 \times \cdots \times n_d$ sparse tensor $\mathcal{T}$ with $N \ll n_1 \cdots n_d$ nonzero entries. These entries are represented via a list $\mathcal{E} = (\mathcal{E}_1, \ldots, \mathcal{E}_N) \in \mathbb{R}^N$ of values and a list $\mathcal{J}$ of $N$ multi-indices $(i^{(1)}_j, \ldots, i^{(d)}_j)$, with $1 \leq i^{(\mu)}_j \leq n_\mu$ and $1 \leq j \leq N$. Given arbitrary DRMs $X_\mu$, $Y_\mu$, the sketches $\Omega_{\mu}$, $\Psi_{\mu}$ can be computed entry-wise as follows:

$$\Omega_{\mu}[k,l] = (Y^T \mathcal{T}^{\leq \mu} X_\mu)[k,l] = \sum_{j=1}^{N} \mathcal{E}_j Y_\mu^{(1)}(i^{(1)}_j, \ldots, i^{(\mu)}_j, k) X_\mu^{(\mu+1)}(i^{(\mu+1)}_j, \ldots, i^{(d)}_j, l) \quad (29)$$

$$\Psi_{\mu}[k,l,m] = ((Y^{(\mu-1)}_\mu \otimes I)\mathcal{T}^{\leq \mu} X_\mu)[k,l,m] = \sum_{j=1}^{N} \mathcal{E}_j Y_{\mu-1}^{(1)}(i^{(1)}_j, \ldots, i^{(\mu-1)}_j, k) X_\mu^{(\mu+1)}(i^{(\mu+1)}_j, \ldots, i^{(d)}_j, m). \quad (30)$$

Note that we tacitly reshaped $Y_\mu$ into an $n_1 \times \cdots \times n_\mu \times r^L_\mu$ tensor in order to access its rows by the multi-index $(i^{(1)}_j, \ldots, i^{(\mu)}_j)$, and similarly for $X_\mu$ and $Y_{\mu-1}$. Furthermore, $\Psi_{\mu}$ is viewed as a $r^{\mu}_{\mu-1} \times n_\mu \times r^R_\mu$ tensor.

Gaussian DRMs. For larger $d$ and modest $N$, most rows of $Y_\mu$ and $X_\mu$ are never needed in $\Psi_{\mu}$. This allows us to efficiently use Gaussian DRMs in this context if we only generate those rows of $X_\mu$ and $Y_\mu$ that will actually be accessed. In order to do this in a distributed setting, we need to generate rows of a DRM on demand and consistently for each mode $\mu$. One such way is to use a simple hashing algorithm to convert the indices $(i^{(1)}_j, \ldots, i^{(\mu)}_j)$ indexing the row of the DRM into a pseudorandom integer, which is then cast into a floating point number between 0 and 1, and finally converted into a normally distributed number. In the context of sketching this seems to produce pseudorandom numbers that are as good as those generated by more advanced pseudorandom number generators.

The formal procedure is detailed below in Algorithm 4.1. The seed $s$ is specific to each mode $\mu$ and ensures that the entries of $X_\mu$ (or $Y_\mu$) are consistent in a distributed setting. The uniform numbers are converted to normally distributed numbers by applying the inverse CDF. For the hashing function, we used splitmix64 which generates adequate pseudo-random numbers.

Using Algorithm 4.1 we only generate at most $N$ rows of each DRM, requiring a total cost of $O(Nrd)$ to generate all DRMs. The asymptotic cost of computing $\Omega_{\mu}$ using $\Psi_{\mu}$ is therefore $O(Nr^2d)$ flops. Computing $\Psi_{\mu}$ using $\Psi_{\mu}$ has the same cost. This gives a total asymptotic cost for all the sketches with Gaussian DRMs of $O(Nr^2d)$, with straightforward parallelization with respect to the number of nonzero entries $N$.

\footnote{See https://rosettacode.org/wiki/Pseudo-random_numbers/Splitmix64 and https://docs.oracle.com/javase/8/docs/api/java/util/SplittableRandom.html}

\footnote{We first set the leftmost 3 bits of $h_j$ to 001 to avoid NaN/infinite flags, and ensure a nonzero exponent. This does not affect the mantissa. Recall that the mantissa of a floating point number with nonzero exponent lies in the interval $[0.5, 1]$.}
Let us now consider a tensor $\mathcal{T}$.

4.2 Sketching tensor trains

**Algorithm 4.1** Hashed pseudorandom number generator that returns one row of a Gaussian DRM.

**Input:** Indices $(i_1, \ldots, i_\mu)$ and integers $(n_1, \ldots, n_\mu)$ such that $1 \leq i_j \leq n_j$, integer $r \geq 1$, seed $s$.

**Output:** $(x_1, \ldots, x_\mu)$ pseudorandom numbers with i.i.d. normal distribution

1: $m \leftarrow 0$
2: for $j = 1$ to $\mu$ do
3: $m \leftarrow n_j(m + i_j)$ // Turn multi-index into linear index
4: end for
5: $m \leftarrow mr + \text{hash}(s)$
6: for $j = 1$ to $r$ do
7: $h_j \leftarrow \text{hash}(m + j)$ // hash produces unsigned integers
8: $x_j \leftarrow \text{Interpret } h_j \text{ as floating point number and extract mantissa.}$
9: $x_j \leftarrow 2x_j - 1.$
10: $x_j \leftarrow \text{normal\_inverse\_cdf}(x_j)$ // Apply inverse CDF of the normal distribution
11: end for

**TT DRMs.** TT DRMs can also be used to efficiently sketch sparse tensors. This is because of the nestedness relations [1], which allow us to efficiently retrieve the entries with indices $[i_1^{(1)}, \ldots, i_\mu^{(1)}, k]$ of the left interface matrices $C^{<\mu}$ subsequently for $\mu = 1, \ldots, d$. Algorithm 4.2 describes the resulting procedure. This allows us to compute within $O(Nr^2d)$ operations all the entries of the TT DRMs $Y_\mu$ and $X_\mu$ needed for sketching in one sweep for all $\mu$. With these entries at hand, we can sketch with TT DRMs using [20] and [30] at the same asymptotic cost as when sketching with Gaussian DRMs.

**Algorithm 4.2** Selected entries of left interface matrices for a TT

**Input:** Tensor train cores $C_1, \ldots, C_\mu$ with each $C_\mu$ of size $r_{\mu-1} \times n_\mu \times r_\mu$;

list of indices $\mathcal{J} = \{(i_1^{(1)}, \ldots, i_1^{(d)}), \ldots, (i_N^{(1)}, \ldots, i_N^{(d)})\}$.

**Output:** Matrices $V_1, \ldots, V_d$ of sizes $N \times r_\mu$ with entries of left interface matrices $C^{<\mu}$ corresponding to $\mathcal{J}$.

1: $V_1 \leftarrow \mathbf{0}_{N \times r_1}$
2: for $j = 1$ to $N$ do
3: $V[j, :] \leftarrow C_1[i_j^{(1)}, :]$
4: end for
5: for $\mu = 2$ to $d$ do
6: $V_\mu \leftarrow \mathbf{0}_{N \times r_\mu}$
7: for $\ell = 1$ to $n_\mu$ do
8: $M \leftarrow C_\mu[i, \ell, :]$
9: for $j = 1$ to $N$ such that $i_j^{(\mu)} = \ell$ do
10: $V_\mu[j, :] \leftarrow V_{\mu-1}[j, :]M$
11: end for
12: end for
13: end for

4.2 Sketching tensor trains

Let us now consider a tensor $\mathcal{T}$ in TT format with a relatively high TT rank $(s_1, \ldots, s_{d-1})$, which we aim to compress to lower TT rank $(r_1, \ldots, r_{d-1})$, that is, $r_\mu < s_\mu$. It is clear that Gaussian DRMs are not well suited for this purpose simply because all entries of the DRMs need to be accessed at least once during the sketch computation, yielding a complexity of at least $O(n^d)$ for computing the sketches. On the other hand, we will see that sketching with TT DRMs yields a computational cost that is linear in $d$ (assuming fixed TT ranks).

Let $B_\mu$ denote the cores of a random TT of TT rank $(r_1^L, \ldots, r_{d-1}^L)$ that defines a left TT DRM $(Y_1, \ldots, Y_{d-1})$ according to Definition 3.3. Similarly, let $A_\mu$ denote the cores of a random TT of TT rank $(r_1^R, \ldots, r_{d-1}^R)$ that defines a right TT DRM $(X_1, \ldots, X_{d-1})$. Denoting the cores of $\mathcal{T}$ by $C_\mu$ and inserting the TT diagrams for
We remark that Algorithm 4.3 can be adapted for sketching matrix product operators (MPO) [Orús, 2014], an extension of the TT format to linear operators. This is because an MPO can be transformed into a TT through reshaping its cores.

Remark 4.1. Algorithm 4.3 partly reproduces the two-sided method, Algorithm 3.3, from [Daas et al., 2021]. In particular, both algorithms compute the matrices $L_\mu = B_{\leq \mu}^\top C_{\leq \mu}$ and $R_\mu = C_{>\mu} A_{>\mu}^\top$ in the same fashion. There are however also differences. In the notation of this paper, Algorithm 3.3 from [Daas et al., 2021] computes a factored pseudo-inverse of $\Omega_\mu = L_\mu R_\mu$ and then uses these factors to compress the cores $C_\mu$ of the input tensor. This last step has the same complexity as the overall algorithm; it is however not linear in $C_\mu$ and thus not streamable. While mathematically equivalent, our algorithms (Algorithm 4.3 combined with Algorithm 3.2) avoid the explicit computation of the factored pseudo-inverse. Through the use of $\Psi_\mu$, the majority of the computation
(Algorithm 4.3) is linear in each $C_\mu$ and thus streamable. Algorithm 3.2 also does not involve the (potentially large) cores of the input tensor.

### 4.3 Sketching canonical polyadic decompositions

The canonical polyadic (CP) decomposition [Kolda and Bader, 2009] represents a tensor $\mathcal{T}$ as a sum of $N$ rank-1 tensors. The CP decomposition can be expressed in terms of factor matrices $V_1 \in \mathbb{R}^{N \times n_1}, \ldots, V_d \in \mathbb{R}^{N \times n_d}$ as follows:

$$
\mathcal{T} = \sum_{i=1}^{N} V_1[i,:] \otimes V_2[i,:] \otimes \cdots \otimes V_d[i,:].
$$

The tensor diagram corresponding to this equation is given by

As in the TT case, sketching $\mathcal{T}$ with Gaussian DRMs would be too expensive and we therefore resort again to left/right TT DRMs with cores $A_1, \ldots, A_d$ and $B_1, \ldots, B_d$, respectively. The sketch $\Omega_\mu = Y_\mu \mathcal{T}^{\leq \mu} X_\mu$ is obtained from

$$
\Omega_\mu = L_\mu R_\mu := [B_{\leq \mu}^T (V_1 \otimes \cdots \otimes V_\mu)] \left[ (V_{\mu+1} \otimes \cdots \otimes V_d) A_{\geq \mu}^T \right],
$$

where $\otimes$ denotes the row-wise Khatri-Rao product, that is, $V_\mu \otimes V_{\mu+1}$ is the $N \times n_\mu n_{\mu+1}$ matrix that contains in the $i$th row all products $V_\mu[i_1, i_2] V_{\mu+1}[i_1, i_3]$. The relation (32) in terms of diagrams:

Similarly we obtain the sketch $\Psi_\mu$ using the following contraction

$$
\Psi_\mu[i_1, i_2, i_3] = \sum_j L_{\mu-1}[i_1, j] V_{\mu}[j, i_2] R_\mu[j, i_3],
$$

or in terms of diagrams:

The matrices $L_\mu$ and $R_\mu$ can be efficiently computed in a single sweep for all $1 \leq \mu \leq d$ at a total cost of $O(N n r^2 d)$ flops. Given $L_\mu$ and $R_\mu$, then $\Omega_\mu$ can be computed at a cost of $O(N r^2)$ flops, and $\Psi_\mu$ at a cost of $O(N n r^2)$ flops. This gives a total cost of $O(N n r^2 d)$ to sketch a CP tensor using TT DRMs. The procedure is summarized in Algorithm 4.4.
Algorithm 4.4 STTA: sketch a tensor in CP decomposition with TT DRMs

**Input:** CP to be sketched with factor matrices $V_1, \ldots, V_d$, left TT DRM with cores $B_1, \ldots, B_d$, right TT DRM with cores $A_1, \ldots, A_d$.

**Output:** Sketches $\Psi_1, \ldots, \Psi_d$ and $\Omega_1, \ldots, \Omega_{d-1}$.

1. $L_1 \leftarrow B_1^T V_1^T$
2. for $\mu = 2$ to $d - 1$ do
   3. $L_\mu[i_1, i_2] \leftarrow \sum_{j_1, j_2} L_{\mu-1}[j_1, i_2] V_\mu[i_2, j_2] B_\mu[j_1, j_2, i_1]$
   4. end for
5. $R_{d-1} \leftarrow V_d A_d^T$
6. for $\mu = d - 2$ to 2 do
   7. $R_\mu[i_1, i_2] \leftarrow \sum_{j_1, j_2} R_{\mu+1}[i_1, j_1] V_{\mu+1}[i_2, j_2] B_{\mu+1}[j_1, j_2, i_1]$
   8. end for
9. for $\mu = 1$ to $d - 1$ do
   10. $\Omega_\mu \leftarrow L_\mu R_\mu$
   11. end for
12. $\Psi_1[1, i_1, i_2] \leftarrow \sum_j V_1[j, i_1] R[j, i_2]$
13. for $\mu = 2$ to $d - 2$ do
   14. $\Psi_\mu[i_1, i_2, i_3] = \sum_j L_{\mu-1}[j, i_1] V_\mu[j, i_2] R_{\mu}[j, i_3]$
   15. end for
16. $\Psi_d[i_1, i_2, 1] \leftarrow \sum_j V_\mu[j, i_1] R_{\mu}[j, i_2]$.

4.4 Sketching Tucker decomposition

The Tucker decomposition [Kolda and Bader, 2009] represents a tensor $T$ of size $n_1 \times \cdots \times n_d$ by a core tensor $C$ of (usually smaller) size $s_1 \times \cdots \times s_d$ multiplied in each mode by a factor matrix $U_\mu \in \mathbb{R}^{n_\mu \times s_\mu}$:

$$T[i_1, \ldots, i_d] = \sum_{j_1, \ldots, j_d} C[j_1, \ldots, j_d] U_1[i_1, j_1] U_2[i_2, j_2] \cdots U_d[i_d, j_d].$$

If the columns of every $U_\mu$ are orthonormal, an effective way to obtain a TT approximation of $T$ is to first approximate the core $C$ and then contract the resulting TT with the factor matrices; see, e.g., [Shi et al., 2021]. While orthonormality can always be ensured by computing QR decompositions of the factor matrices, the resulting updates of the core tensor come at a cost of $O(d s_\mu + 1)$ and typically destroy all sparsity structure in $C$. Sketching $T$ directly avoids these disadvantages and comes with further benefits; for example, its linearity allows us to combine tensors with different types of structure.

We again use left/right TT DRMs with cores $A_1, \ldots, A_d$ and $B_1, \ldots, B_d$, respectively, for sketching a tensor in Tucker format. The sketches $\Omega_\mu$ and $\Psi_\mu$ for a Tucker tensor $T$ are computed according to the following tensor diagrams:
Carrying out the multiplications $B^T_{\mu}(U_1 \otimes \cdots \otimes U_d)$ and $(U_{\mu+1} \otimes \cdots \otimes U_d)^T A_{\mu}^T$ within the TT format takes $O(dnr^2s)$ flops. The subsequent explicit computation of these matrices by carrying out the contractions in a suitable order requires $O(r^2sd^{-1})$. The cost of computing each of the sketches $\Omega_{\mu}$ and $\Psi_{\mu}$ is bounded by $O(rs^d + r^2sd^{-1} + nr^2s)$. This gives a total cost for sketching the Tucker representation using TT DRMs of $O(d(nr^2s + rs^d + r^2sd^{-1}))$.

5 Numerical experiments

In the following, we compare STTA numerically with the classical TT-SVD algorithm and the TT-HMT method mentioned in the introduction. The TT-SVD method is used as baseline since it usually has the smallest approximation error for a fixed rank. The method is however very costly since it requires truncated SVDs of large matrixizations. It is also not streamable and less suited for large structured tensors.

STTA and TT-HMT on the other hand are designed for larger tensors. While both methods have similar flop counts for dense tensors, TT-HMT does not support streaming and is difficult to implement efficiently in a distributed setting. In its standard formulations of [Huber et al., 2017] [Wolf, 2019] [Che and Wei, 2019], TT-HMT is also not well suited for structured tensors (but we present a reformulation in the next section that partly alleviates this).

When varying the TT rank in the experiments, we increase it uniformly among all modes whenever possible. At the left/right borders of the TT, the TT ranks may be constrained by the size of the tensor: Given a prescribed rank $r$ we set $r_1 = \min\{r, n_1\}$, $r_{d-1} = \min\{r, n_d\}$, and so on. Unless noted otherwise we set $r^L = 2r^R$ for the STTA method, were $r^L_{\mu}$ and $r^R_{\mu}$ are the rank of the left and right DRMs $X_{\mu}$ and $Y_{\mu}$; see (42). Since STTA and TT-HMT involve random quantities, we perform 30 trials and show the spread between the 20th and 80th percentile of the relative error $\|T - \tilde{T}\|_F/\|T\|_F$ for the computed approximation $\tilde{T}$.

A Python implementation of STTA and the code to reproduce all the experiments is freely available at [https://github.com/RikVoorhaar/tt-sketch](https://github.com/RikVoorhaar/tt-sketch).

5.1 Baseline method TT-HMT

We recall how the HMT method can be used for sketching tensors leading to what we call the TT-HMT method, summarized in Algorithm 5.1. This algorithm produces the same approximation as the algorithms proposed in [Huber et al., 2017] [Wolf, 2019] [Che and Wei, 2019]. An important numerical and implementation difference is that Algorithm 5.1 continues to operate on the original tensor $T$ whereas the other algorithms successively reduce the size of $T$, like the TT-SVD method. The benefit is that our formulation can exploit structure in $T$ since only the original $T$ needs to be accessed during the algorithm.

Superficially TT-HMT method is very similar to the STTA method. However, instead of performing a two-sided sketch using two independent DRMs $X_{\mu}$ and $Y_{\mu}$, the sketches $\Psi_{\mu}$ on line 4, 6 and 8 in Algorithm 5.1 are computed using only a right DRM $X_{\mu}$ and on the left we multiply by the partial contraction $C_{\leq \mu-1}$ of the TT-cores previously computed. Effectively, this partial contraction $C_{\leq \mu-1}$ can be viewed as a non-random left TT-DRM, and therefore the sketches $\Psi_{\mu}$ can be computed for any of the structured tensors mentioned in Section 4. Note that the TT-HMT method is not streamable due to the orthogonalization on line 11, and requires at least $d$ passes of the data. Furthermore this means it is not possible to parallelize the for loop starting on line 1. On the other hand, the HMT method uses fewer and smaller sketches, making it faster than STTA for smaller rank approximations (for larger rank, this is no longer always true; see Section 5.8), and it also produces orthogonalized TTs. It has a theoretical error bound in expectation with a smaller error constant than STTA; see [Huber et al., 2017].

5.2 Sketching dense tensors

We first consider the problem of sketching dense tensors of shape $n_1 \times \cdots \times n_d$. In particular, we will test the performance of the algorithms on the Hilbert tensor

$$T_{\text{Hilbert}}[i_1, \ldots, i_d] = (i_1 + \cdots + i_d - d + 1)^{-1}$$

and the square-root-sum tensor

$$T_{\text{sqrt}}[i_1, \ldots, i_d] = \sqrt{\sum_{j=1}^{d} \frac{n_j - i_j}{n_j - 1} a + \frac{i_j - 1}{n_j - 1} b},$$

with $0 < a < b$. Both tensors have exponential decay of the singular values of the unfoldings $T_{\leq \mu}$ for all $\mu$; see, e.g., [Hackbusch, 2019]. For $T_{\text{sqrt}}$, the decay worsens as $a$ approaches zero. In our experiments we take $a = 0.2$
Algorithm 5.1 TT-HMT

Input: Tensor $\mathcal{T}$ of shape $n_1 \times \cdots \times n_d$; Right DRMs $X_{\mu}$ of size $(n_{\mu+1} \cdots n_d) \times r_{\mu}$.
Output: Cores $C_{\mu}$ of size $r_{\mu-1} \times n_{\mu} \times r_{\mu}$ for $1 \leq \mu \leq d$ of a tensor train approximation of $\mathcal{T}$.

1: for $\mu = 1$ to $d$ do
2: $\mathcal{T}^{\leq \mu} \leftarrow \text{reshape}(\mathcal{T}, (n_1 \cdots n_{\mu}) \times (n_{\mu+1} \cdots n_d))$
3: if $\mu = 1$ then
4: $\Psi_1 \leftarrow \mathcal{T}^{\leq 1} X_1$
5: else if $\mu < d$ then
6: $\Psi_{\mu} \leftarrow (C_{\leq \mu-1} \otimes I_{n_{\mu}}) \mathcal{T}^{\leq \mu} X_{\mu}$
7: else
8: $\Psi_d \leftarrow (C_{\leq d-1} \otimes I_{n_d}) \mathcal{T}^{\leq d}$
9: end if
10: if $\mu < d$ then
11: $C_{\mu} \leftarrow \text{orth}(\Psi_{\mu}^{L})$ \hfill // Q factor of QR decomposition
12: $C_{\leq \mu} \leftarrow (C_{\leq \mu-1} \otimes I_{n_{\mu}}) C_{\mu}^{L}$
13: $C_{\mu} \leftarrow \text{reshape}(C_{\mu}^{L}, r_{\mu-1} \times n_{\mu} \times r_{\mu}^{L})$
14: else
15: $C_d \leftarrow \Psi_d$
16: end if
17: end for

and $b = 2$. Furthermore, we take $d = 7$ and $n_1 = \cdots = n_7 = 5$ for $\mathcal{T}_{\text{Hilbert}}$, and $d = 5$ and $n_1 = \cdots = n_5 = 10$ for $\mathcal{T}_{\text{sqrt}}$. The approximation errors for different TT ranks are shown in Fig. 1 and Fig. 2.

We observe that the error of STTA and TT-HMT are comparable and only slightly worse than TT-SVD, and has acceptable variance. For both STTA and TT-HMT, there is not much difference between the use of Gaussian and TT DRMs when comparing approximation error and its variance. This is surprising since TT DRMs have less randomness than Gaussian DRMs.

5.3 Sketching a sum of TT and a sparse tensor

One of the major advantages of STTA is that it can approximate the sum of several tensors in parallel, possibly in different formats. As an example we consider the problem of approximating the sum of a low-rank TT and a sparse tensor.

We consider a rank-5 TT of size $(10, 10, 10, 10, 10)$ consisting of cores with i.i.d. $N(0, 1/r^2)$ entries. To this we add a random sparse tensor with 100 nonzero entries. The nonzero entries are normally distributed with standard deviation ranging log-uniformly between $10^{-3}$ and $10^{-20}$. The resulting approximation is shown in Fig. 3.

Figure 1: Approximation errors for the Hilbert tensor different TT-ranks for the TT-SVD, STTA and TT-HMT methods. The markers are ordered left-to-right in the same order as the legend.
5.4 Sketching a sum of TT

Next we consider the problem of sketching a sum of low-rank TTs. Specifically we take a sum of 20 rank-3 TTs of size $10 \times 10 \times 10 \times 10 \times 10$ of form

$$T = \sum_{i=0}^{19} 10^{-i}T_i,$$

where each $T_i$ is a random TT of rank 3 consisting of cores with i.i.d. $\mathcal{N}(0, 1/r^2n)$ entries. The resulting approximation errors for the different algorithms are shown below in Fig. 4.

5.5 Sketching a CP tensor

We consider the problem of sketching a CP tensor. The CP tensor is of form

$$T = \sum_{i=1}^{100} \frac{1}{r^5} v_{1,i} \otimes \cdots \otimes v_{5,i},$$

where each $v_{\mu,i} \in \mathbb{R}^{10}$ is a normalized Gaussian vector. The results are shown in Fig. 5.

5.6 The effect of oversampling

The rank $r^L_\mu$ of the left DRMs $X_\mu$ is required to be smaller than the rank $r^R_\mu$ of the left DRMs $Y_\mu$ for all $\mu$ (or vice versa). Let $r^R_\mu = r^L_\mu + \ell$. We investigate the effect of varying the oversampling $\ell > 0$ on the approximation
error. As test problem we will use the same sum of TTs as in Section 5.4 and we fix the TT rank of the approximation at $r_\mu = r = 10$ for all $\mu$. The result of this experiment is shown below in Fig. 6.

Observe that increasing the oversampling $\ell$ improves both the median and the variance of the approximation error. (Note that the scale for the error in Fig. 6 is logarithmic.) However, in this particular case there are quickly diminishing returns after $\ell \approx 10$. Since the cost of all algorithms depends at most linearly on $\ell$, setting $\ell = r$ (as we have done in the experiments above), will only double the computational cost compared to no oversampling at all. This provides a reasonable trade-off between approximation error and speed.

5.7 Dependence of the error on the tensor order

Theorem 3.2 implies that the approximation constant of the STTA method with Gaussian DRMs scales exponentially with the number of modes $d$. In practice, it appears that the approximation constant scales as $\sqrt{d - 1}$ for Gaussian and TT DRMs, like with the classical TT-SVD and the TT-HMT method using Gaussian DRMs.

To test this, we compute a rank 10 approximation of a TT of uniform TT-rank $r = 30$ and size $30^d$ for different values of $d$. For each unfolding, the TT has singular values decaying exponentially between $\sqrt{r}$ and $\sqrt{r} \cdot 10^{-20}$ (this keeps the Frobenius norm of the TT approximately constant as $d \to \infty$).

In Fig. 7 we compare the scaling of the approximation error with the tensor order $d$ for TT-HMT and STTA using TT DRMs. We normalize the approximation error by dividing by the approximation error of the TT-SVD method for the same rank. Note that in this context by TT-SVD we mean the standard SVD-based rounding procedure for TTs.

We observe that the STTA method converges to an approximation error roughly 13 times worse than the TT-SVD method, whereas the HMT method converges to an approximation error roughly 8 times worse than the TT-SVD method. This suggests that the approximation constant of the STTA method asymptotically scales
Effect of right oversampling $\ell$ on approximation error

Figure 6: Effect of oversampling parameter $\ell_\mu = |r^L_\mu - r^R_\mu|$ on the approximation error for STTA.

as $\sqrt{d-1}$, just like the TT-SVD and TT-HMT methods.

Dependence of the error on the tensor order

Figure 7: Normalized approximation errors for TTs with varying tensor orders for rank 10 approximations obtained with TT-HMT (left markers) and STTA (right markers). The approximation error is normalized by dividing by the approximation error of the TT-SVD approximation for the same rank.

We remark that the scaling $1/r^L_\mu$ mentioned in Definition 3.3 of TT-DRMs becomes essential for the large number of modes used in this experiment. Different scalings of the cores of the TT-DRMs leads to severe problems with underflow or overflow.

5.8 Timing comparison

We consider the trade-off between speed, rank and approximation error for STTA and the TT-HMT method. In particular we approximate a random TT of size $150 \times 150 \times 150 \times 150 \times 150$ of TT-rank 150. The singular values of each unfolding were made to decay exponentially between 1 and $10^{-10}$. In Fig. 8 we report the time taken and relative error of approximation for the TT-HMT and STTA method using TT-DRMs of varying ranks. All experiments were performed on a 2x64 core AMD Epyc 7742. For the approximation error and the time taken we report the median over 100 trials.

While the TT-HMT method is faster for a low-rank approximation, the STTA method is faster than TT-HMT for sufficiently large rank approximations. This is because the cost of the orthogonalization step in TT-HMT becomes dominant for sufficiently large ranks, and offsets the cost of the second sketch in STTA. This is especially true if we use a low amount of oversampling ($r^L_\mu = r^R_\mu + 3$).
Figure 8: Relation between timing and error (left) and the TT-rank of the TT-DRM and time taken (right) for TT-HMT, and STTA with respectively $r_L^\mu = r_R^\mu + 3$ (STTA+3) and $r_L^\mu = 2r_R^\mu$ (STTAx2).

6 Conclusions

This paper introduced Streaming Tensor Train Approximation (STTA) – an algorithm for approximating a tensor in the tensor train format that is both streamable and one-pass. Since the method computes the approximation using only two-sided sketches of the given tensor, STTA can efficiently update the approximation after linear updates without needing to access the original tensor. In addition, STTA can be applied to specific structures of the tensor such as (linear combinations of) sparsity and various low-rank tensor formats (CP, Tucker, tensor train). In case of sketching with Gaussian dimension reduction matrices and with sufficiently large oversampling, we proved a quasi-optimal error bound in expectation by extending existing results from the Generalized Nyström method for matrices. In addition, the numerical experiments showed that the approximation error concentrates relatively quickly around its mean without needing much oversampling in practice. This is also true when sketching using structured dimension reduction matrices based on tensor trains. Due to the high-level of parallelism, STTA is also faster than computing tensor train approximations based on the Halko–Martinsson–Tropp method in a multi-core environment.

References

[Alger et al., 2020] Alger, N., Chen, P., and Ghattas, O. (2020). Tensor train construction from tensor actions, with application to compression of large high order derivative tensors. arXiv:2002.06244. doi:10.48550/arXiv.2002.06244.

[Bachmayr et al., 2016] Bachmayr, M., Schneider, R., and Uschmajew, A. (2016). Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Foundations of Computational Mathematics, 16. doi:10.1007/s10208-016-9317-9.

[Che and Wei, 2019] Che, M. and Wei, Y. (2019). Randomized algorithms for the approximations of Tucker and the tensor train decompositions. Adv Comput Math, 45(1):395–428. doi:10.1007/s10444-018-9622-8.

[Clarkson and Woodruff, 2009] Clarkson, K. L. and Woodruff, D. P. (2009). Numerical linear algebra in the streaming model. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pages 205–214. Association for Computing Machinery. doi:10.1145/1536444.1536445.

[Daas et al., 2021] Daas, H. A., Ballard, G., Cazeaux, P., Hallman, E., Miedlar, A., Pasha, M., Reid, T. W., and Saibaba, A. K. (2021). Randomized algorithms for rounding in the Tensor-Train format. arXiv:2110.04393. doi:10.48550/arXiv.2110.04393.

[Grasedyck et al., 2013] Grasedyck, L., Kressner, D., and Tobler, C. (2013). A literature survey of low-rank tensor approximation technique. GAMM-Mitteilungen, 36(1):53—78. doi:10.1002/gamm.201310004.

[Hackbusch, 2019] Hackbusch, W. (2019). Tensor Spaces and Numerical Tensor Calculus, volume 56 of Springer Series in Computational Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, second edition. doi:10.1007/978-3-642-28027-6.
