Randomized CP tensor decomposition

IOP Publishing Mach. Learn.: Sci. Technol. 1 (2020) 025012 https://doi.org/10.1088/2632-2153/ab8240

MACHINE LEARNING
Science and Technology

Randomized CP tensor decomposition

N Benjamin Erichson[1,5], Krithika Manohar[2], Steven L Brunton[3] and J Nathan Kutz[4]

[1] Department of Statistics University of California, Berkeley, United States of America
[2] Department of Applied & Computational Mathematics California Institute of Technology, Pasadena, United States of America
[3] Department of Mechanical Engineering University of Washington, Seattle, United States of America
[4] Department of Applied Mathematics University of Washington, Seattle, United States of America
[5] Author to whom any correspondence should be addressed.

E-mail: erichson@berkeley.edu, kmanohar@caltech.edu, sbrunton@uw.edu and kutz@uw.edu

Keywords: randomized algorithms; randomized least squares; dimension reduction; multilinear algebra; CP decomposition; canonical polyadic tensor decomposition.

RECEIVED
1 December 2019

REVISED
13 March 2020

ACCEPTED FOR PUBLICATION
23 March 2020

PUBLISHED
27 May 2020

Abstract

The *CANDECOMP/PARAFAC* (CP) tensor decomposition is a popular dimensionality-reduction method for multiway data. Dimensionality reduction is often sought after since many high-dimensional tensors have low intrinsic rank relative to the dimension of the ambient measurement space. However, the emergence of 'big data' poses significant computational challenges for computing this fundamental tensor decomposition. By leveraging modern randomized algorithms, we demonstrate that coherent structures can be learned from a smaller representation of the tensor in a fraction of the time. Thus, this simple but powerful algorithm enables one to compute the approximate CP decomposition even for massive tensors. The approximation error can thereby be controlled via oversampling and the computation of power iterations. In addition to theoretical results, several empirical results demonstrate the performance of the proposed algorithm.

1. Introduction

Advances in data acquisition and storage technology have enabled the acquisition of massive amounts of data in a wide range of emerging applications. In particular, numerous applications across the physical, biological, social and engineering sciences generate large multidimensional, multi-relational and/or multi-modal data. Efficient analysis of this data requires dimensionality reduction techniques. However, traditionally employed matrix decompositions techniques such as the singular value decomposition (SVD) and principal component analysis (PCA) can become inadequate when dealing with multidimensional data. This is because reshaping multi-modal data into matrices, or *data flattening*, can fail to reveal important structures in the data.

Tensor decompositions overcome the information loss from flattening. The canonical CP tensor decomposition expresses an N-way tensor as a sum of rank-one tensors to extract multi-modal structure. It is particularly suitable for data-driven discovery, as shown by Hong *et al* (2020) for various learning tasks on real world data. The idea of tensor compression, for instance, eases computational bottlenecks by constructing smaller (compressed) tensors, which are then used as a proxy to efficiently compute CP decompositions. Compressed tensors may be obtained, for example, using the Tucker decomposition of a tensor into one small core tensor and N unitary matrices However, this approach requires the expensive computation of the left singular vectors for each mode.

Related work. This computational challenge can be tackled using modern randomized techniques developed to compute the SVD. Tsourakakis (2010) presents a randomized Tucker decomposition algorithm based on the random sparsification idea of Achlioptas and McSherry (2007) for computing the SVD. Zhou *et al* (2014) proposed an accelerated randomized CP algorithm using the ideas of Halko *et al* (2011) without the use of power iterations. An alternative randomized tensor algorithm proposed by Drineas and Mahoney (2007) uses random column selection. Related work by Vervliet *et al* (2014) proposed a sparsity-promoting algorithm for incomplete tensors using compressed sensing.

Another efficient approach to building large-scale tensor decompositions is the subdivision of a tensor into a set of blocks. These smaller blocks can then be used to approximate the CP decomposition of the full

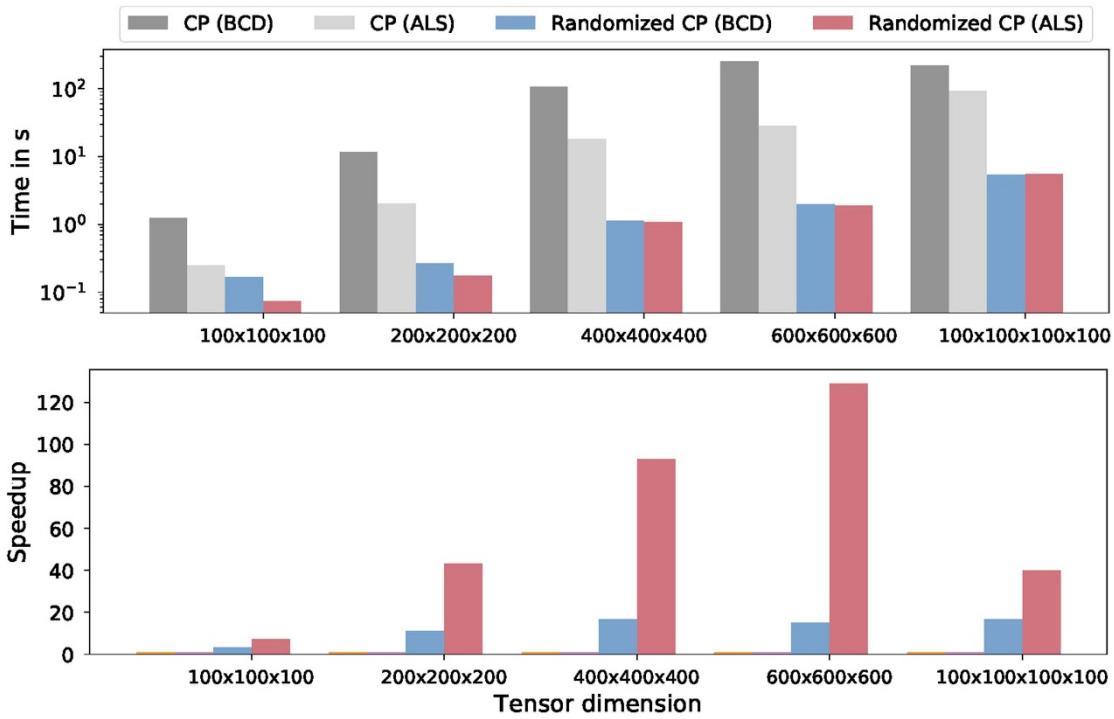


Figure 1. Algorithm runtimes and speedups for varying tensor dimensions for a rank $k = 20$ approximation. Speedups rise sharply with increasing dimensions.

tensor in a parallelized or distributed fashion (Phan and Cichocki 2011). Sidiropoulos *et al* (2014) fused random projection and blocking into a highly computationally efficient algorithm. More recently, Vervliet and Lathauwer (2016) proposed a block sampling CP decomposition method for the analysis of large-scale tensors using randomization, demonstrating significant computational savings while attaining near optimal accuracy. These block based algorithms are particularly relevant if the tensor is too large to fit into fast memory.

Contributions. We present the randomized CP algorithm, which is closely related to the randomized methods of Halko *et al* (2011). Our method proceeds in two stages. The first stage uses random projections with power iterations to obtain a compressed tensor (Algorithm 1). The second stage applies either alternating least squares (ALS) or block coordinate descent (BCD) to the compressed tensor (Algorithms 2 and 3), at significantly reduced cost. Finally, the CP decomposition of the original tensor is obtained by projecting the compressed factor matrices back using the basis derived in Algorithm 1.

Randomized algorithms for CP decomposition have been proposed (Zhou *et al* 2014, Battaglino *et al* 2018), albeit without incorporating power iterations. The power iteration concept is fundamental for modern high-performance randomized matrix decompositions, but to the best of our knowledge, has not been applied in the context of tensors. Without power iterations, the performance of randomized algorithms based on random projections can suffer considerably in the presence of white noise. We combine power iterations with oversampling to further control the error of the decomposition.

Embedding the CP decomposition into this probabilistic framework allows us to achieve significant computational savings, while producing near-optimal approximation quality. For motivation, figure 1 shows the computational savings of a rank $k = 20$ approximation for varying tensor dimensions. Here we compare the ALS and BCD solver for computing the deterministic and randomized CP decomposition. The computational advantage of the randomized algorithms is significant, while sacrificing a negligible amount of accuracy.

Organization. The paper is organized as follows: section 2 briefly reviews the CP decomposition and randomized matrix decompositions. Section 3 introduces the randomized CP tensor decomposition algorithm. Section 4 presents the evaluation of the computational performance, and examples. Finally, section 5 summarizes the research findings.

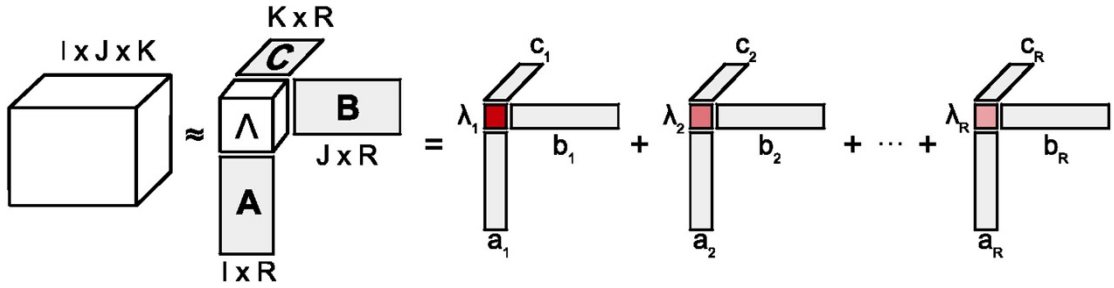


Figure 2. Schematic of the CP decomposition.

2. Background

Ideas for multi-way factor analysis emerged in the 1920s with the formulation of the polyadic decomposition by Hitchcock (1927). However, the polyadic tensor decomposition only achieved popularity much later in the 1970s with the canonical decomposition (*CANDECOMP*) in psychometrics, proposed by Carroll and Chang (1970). Concurrently, the method of parallel factors (*PARAFAC*) was introduced in chemometrics by Harshman (1970). Hence, this method became known as the CP (*CANDECOMP/PARAFAC*) decomposition. However, in the past, computation was severely inhibited by available computational power. Today, tensor decompositions enjoy increasing popularity, yet runtime bottlenecks persist.

2.1. Notation

Scalars are denoted by lower case letters x, vectors as bold lower case letters $\mathbf{x}$, and matrices by bold capitals $\mathbf{X}$. Tensors are denoted by calligraphic letters $\mathcal{X}$. The mode-n unfolding of a tensor is expressed as $\mathcal{X}_{(n)}$, while the mode-n folding of a matrix is defined as $\mathbf{X}_{(n)}$. The vector outer product, the Kronecker product, the Khatri-Rao product and the Hadamard product are denoted by $\circ$, $\otimes$, $\odot$, and $*$, respectively. The inner product of two tensors is expressed as $\langle \cdot, \cdot \rangle$, and $\| \cdot \|_F$ denotes the Frobenius norm for both matrices and tensors.

2.2. CP decomposition

The CP decomposition is the tensor equivalent of the SVD since it approximates a tensor by a sum of rank-one tensors. Here, *tensor rank* is defined as the smallest sum of rank-one tensors required to generate the tensor (Hitchcock 1927). The CP decomposition approximates these rank-one tensors. Given a third order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, the rank-$R$ CP decomposition is expressed as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{1}$$

where $\circ$ denotes the outer product. Specifically, each rank-one tensor is formulated as the outer product of the rank-one components $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$. Components are often constrained to unit length with the weights absorbed into the vector $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_R] \in \mathbb{R}^R$. Equation (1) can then be re-expressed as (See figure 2)

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \lambda_r \cdot \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{2}$$

More compactly the components can be expressed as factor matrices, i.e. $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_R]$. Using the Kruskal operator as defined by Kolda and Bader (2009), (2) can be more compactly expressed as

$$\boldsymbol{\mathcal{X}} \approx [[\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]].$$

2.3. Randomized matrix algorithms

The efficient computation of low rank matrix approximations is a ubiquitous problem in machine learning and data mining. Randomized matrix algorithms have been demonstrated to be highly competitive and

robust when compared to traditional deterministic methods. Randomized algorithms aim to construct a smaller matrix (henceforth called sketch) designed to capture the essential information of the source matrix (Mahoney 2011, Drineas and Mahoney 2016). The sketch can then be used for various learning tasks. There exist several strategies for obtaining such a sketch, with random projections being the most robust off-the-shelf approach. Randomized algorithms have been in particular studied for computing the near-optimal low-rank SVD (Frieze *et al* 2004, Liberty *et al* 2007, Woolfe *et al* 2008, Martinsson *et al* 2011).

Following the seminal work by Halko *et al* (2011), a randomized algorithm computes the low-rank matrix approximation

$$\underset{m\times n}{\mathbf{A}} \approx \underset{m\times k}{\mathbf{Q}} \underset{k\times n}{\mathbf{B}}$$

where the target rank is denoted as k, and assumed to be $k \ll \min\{m, n\}$. Intuitively, we construct a sketch $\mathbf{Y} \in \mathbb{R}^{m\times k}$ by forming a random linear weighted combination of the columns of the source matrix $\mathbf{A}$. More concisely, we construct the sketch as

$$\mathbf{Y} = \mathbf{A}\,, \tag{3}$$

where $\in \mathbb{R}^{m\times k}$ is a random test matrix with entries drawn from, for example, a Gaussian distribution. If $\mathbf{A}$ has exact rank k, then the sketch $\mathbf{Y}$ spans with high probability a basis for the column space of the source matrix. However, most data matrices do not feature exact rank (i.e. the singular values $\{\sigma_i\}_{i=k+1}^{n}$ are non-zero). Thus, instead of just using k samples, it is favorable to construct a slightly oversampled sketch $\mathbf{Y} \in \mathbb{R}^{m\times l}$ which has $l = k + p$ columns, were p denotes the number of additional samples. In most situations small values $p = \{10, 20\}$ are sufficient to obtain a good basis that is comparable to the best possible basis (Martinsson 2016). An orthonormal basis $\mathbf{Q} \in \mathbb{R}^{m\times l}$ is then obtained via the QR-decomposition $\mathbf{Y} = \mathbf{QR}$, such that

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^\top \mathbf{A}$$

is satisfied. Finally, the source matrix $\mathbf{A}$ is projected to low-dimensional space

$$\mathbf{B} = \mathbf{Q}^\top \mathbf{A}, \tag{4}$$

where $\mathbf{B} \in \mathbb{R}^{l\times n}$. (Note, that equation (4) requires a second pass over the source matrix.) The matrix $\mathbf{B}$ can then be used to efficiently compute the matrix decomposition of interest, e.g. the SVD. The approximation error can be controlled by a combination of oversampling and power iteration (Rokhlin *et al* 2009, Halko *et al* 2011, Gu 2015).

Randomized matrix algorithms are not only pass efficient, but they also have the ability to exploit modern computational parallelized and distributed computing architectures. Implementations in *MATLAB*, *C* and *R* are provided by Szlam *et al* (2014), Voronin and Martinsson (2015), and Erichson *et al* (2016).

3. Randomized CP decomposition

Given a third order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, the objective of the CP decomposition is to find a set of R normalized rank-one tensors $\{\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\}_{r=1}^{R}$ which best approximates $\boldsymbol{\mathcal{X}}$, i.e. minimizes the Frobenius norm

$$\underset{\hat{\boldsymbol{\mathcal{X}}}}{\text{minimize}} \quad \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_F^2 \quad \text{subject to} \quad \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{5}$$

The challenge is that this problem is highly nonconvex and unlike PCA there is no closed-form solution. Solution methods for this optimization problem therefore rely on iterative schemes (e.g. alternating least squares). These solvers are not guaranteed to find a global minimum, yet for many practical problems, they can find high-quality solutions. Iterative schemes, however, can be computationally demanding if the dimensions of $\boldsymbol{\mathcal{X}}$ are large.

Fortunately, only the column spaces of the modes are of importance for obtaining the factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, rather then the individual columns of the mode matricizations $\boldsymbol{\mathcal{X}}_{(1)}$, $\boldsymbol{\mathcal{X}}_{(2)}$, $\boldsymbol{\mathcal{X}}_{(3)}$ of the tensor $\boldsymbol{\mathcal{X}}$. This is because the CP decomposition learns the components based on proportional variations in inter-point distances between the components. Therefore, a compressed tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{k\times k\times k}$ must preserve pairwise Euclidean distances, where $k \geq R$. This in turn requires that column spaces, and thus pairwise distances, are approximately preserved by compression - this can be achieved by generalizing the concepts of randomized matrix algorithms to tensors. We build upon the methods introduced by Martinsson *et al* (2011) and Halko

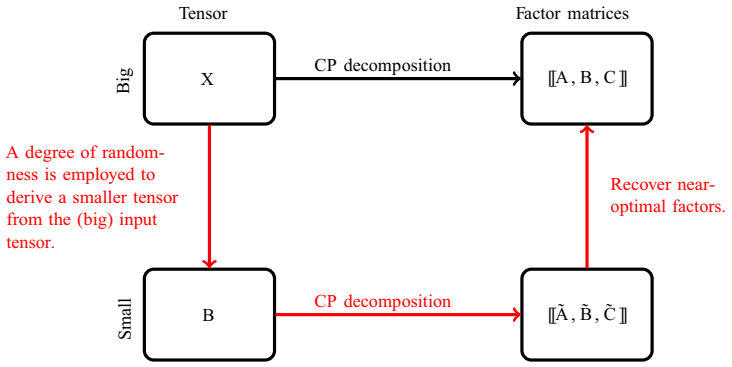


Figure 3. Schematic of the randomized CP decomposition architecture. First, a degree of randomness is employed to derive a smaller tensor $\mathcal{B}$ from the big tensor $\mathcal{X}$. Then, the CP decomposition is performed on $\mathcal{B}$. Finally, the near-optimal high-dimensional factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are recovered from the approximate factor matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$.

et al (2011), as well as related work on randomized tensors by Drineas and Mahoney (2007), who proposed a randomized algorithm based on random column selection.

Figure 3 shows the schematic of the randomized CP decomposition architecture. Our approach proceeds in two stages. The first stage, detailed in section 3.1, applies random projections with power iteration to obtain a compressed tensor $\mathcal{B}$, with expressivity analysis in section 3.1.1. In section 3.2, we describe two algorithms for performing CP on the compressed tensor $\mathcal{B}$ and approximating the original CP factor matrices. Section 3.3 provides additional details about our implementation in Python.

3.1. Randomized tensor algorithm

The aim is to use randomization as a computational resource to efficiently build a suitable basis that captures the action of the tensor $\mathcal{X}$. Assuming an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, the aim is to obtain a smaller compressed tensor $\mathcal{B} \in \mathbb{R}^{k \times \cdots \times k}$, so that its N tensor modes capture the action of the input tensor modes. Hence, we seek a natural basis in the form of a set of orthonormal matrices $\{\mathbf{Q}_n \in \mathbb{R}^{I_n \times R_n}\}_{n=1}^{N}$, so that

$$\mathcal{X} \approx \mathcal{X} \times_1 \mathbf{Q}_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^\top. \tag{6}$$

Here the operator $\times_n$ denotes tensor-matrix multiplication defined as follows.

Definition 1. *The n-mode matrix product* $\mathcal{X} \times_n \mathbf{Q}_n \mathbf{Q}_n^\top$ *multiplies a tensor by the matrices* $\mathbf{Q}_n \mathbf{Q}_n^\top$ *in mode n, i.e. each mode-n fiber is multiplied by* $\mathbf{Q}_n \mathbf{Q}_n^\top$

$$\mathcal{M} = \mathcal{X} \times_n \mathbf{Q}_n \mathbf{Q}_n^\top \quad \Leftrightarrow \quad \mathbf{M}_{(n)} = \mathbf{Q}_n \mathbf{Q}_n^\top \mathcal{X}_{(n)}.$$

Given a fixed target rank k, these basis matrices can be efficiently obtained using a randomized algorithm. First, the approximate basis for the nth tensor mode is obtained by drawing k random vectors $\omega_1, \ldots, \omega_k \in \mathbb{R}^{\prod_{i \neq n} I_i}$ from a Gaussian distribution. (Note that if N is large, it might be favorable to draw the entries of ω from a quasi random sequence, e.g. Halton or Sobol sequence. These so-called quasi-random numbers are known to be less correlated in high-dimensions.) Stacked together, the k random vectors ω form the random test matrix $_n \in \mathbb{R}^{\prod_{i \neq n} I_i \times k}$ used to sample the column space of $\mathcal{X}_{(n)} \in \mathbb{R}^{I_n \times \prod_{i \neq n} I_i}$ as follows

$$\mathbf{Y}_n = \mathcal{X}_{(n)\ n}, \tag{7}$$

where $\mathbf{Y}_n \in \mathbb{R}^{I_n \times k}$ is the sketch. The sketch serves as an approximate basis for the range of the nth tensor mode. Probability theory guarantees that the set of random vectors $\{\omega_i\}_{i=1}^{k}$ are linearly independent with high probability. Hence, the corresponding random projections $\mathbf{y}_1, \ldots, \mathbf{y}_k$ efficiently sample the range of a rank deficient tensor mode $\mathcal{X}_{(n)}$. The economic QR decomposition of the sketch $\mathbf{Y}_n = \mathbf{Q}_n \mathbf{R}_n$ is then used to obtain a natural basis, so that $\mathbf{Q}_n \in \mathbb{R}^{I_n \times k}$ is orthonormal and has the same column space as $\mathbf{Y}_n$. The final step restricts the tensor mode to this low-dimensional subspace

$$\mathcal{B}_n = \mathcal{X} \times_n \mathbf{Q}_n^\top \quad \Leftrightarrow \quad \mathbf{B}_n = \mathbf{Q}_n^\top \mathcal{X}_{(n)}. \tag{8}$$

Thus, after N iterations a compressed tensor $\mathcal{B}$ and a set of orthonormal matrices is obtained. Since this is an iterative algorithm, we set $\mathcal{X} \leftarrow \mathcal{B}_n$ after each iteration. The number of columns of the basis matrices form a trade-off between accuracy and computational performance. We aim to use as few columns as possible, yet

allow an accurate approximation of the input tensor. Assuming that the tensor $\boldsymbol{\mathcal{X}}$ exhibits low-rank structure, or equivalently, the rank R is much smaller than the ambient dimensions of the tensor, the basis matrices will be an efficient representation. In practice, the compression performance is improved through using oversampling, i.e. drawing $l = k + p$ random vectors where k is the target rank and p the oversampling parameter.

The randomized algorithm as presented requires that the mode-n unfolding of the tensor has a rapidly decaying spectrum in order to achieve good performance. However, this assumption is often not suitable, and the spectrum exhibits slow decay if the tensor is compressed several times. To overcome this issue, the algorithm's performance can be substantially improved using power iterations (Rokhlin *et al* 2009, Halko *et al* 2011, Gu 2015). Power iterations turn a slowly decaying spectrum into a rapidly decaying one by taking powers of the tensor modes. Thus, instead of sampling $\boldsymbol{\mathcal{X}}_{(n)}$ we sample from the following tensor mode

$$\boldsymbol{\mathcal{X}}^{q}_{(n)} := (\boldsymbol{\mathcal{X}}_{(n)}\boldsymbol{\mathcal{X}}^{\top}_{(n)})^{q}\boldsymbol{\mathcal{X}}_{(n)},$$

where q denotes a small integer. This power operation enforces that the singular values σ_j of $\boldsymbol{\mathcal{X}}^{q}_{(n)}$ are $\{\sigma_j^{2q+1}\}_j$. Instead of using (7), an improved sketch is computed

$$\mathbf{Y}_n = (\boldsymbol{\mathcal{X}}_{(n)}\boldsymbol{\mathcal{X}}^{\top}_{(n)})^{q}\boldsymbol{\mathcal{X}}_{(n)\ n}. \tag{9}$$

However, if (9) is implemented in this form the basis may be distorted due to round-off errors. Therefore in practice, normalized subspace iterations are used to form the sketch, meaning that the sketch is orthornormalized between each power iteration in order to stabilize the algorithm. For implementation, see Voronin and Martinsson (2015) and Szlam *et al* (2014). The combination of oversampling and additional power iterations can be used to control the trade-off between approximation quality and computational efficiency of the randomized tensor algorithm. Our results, for example, show that just $q = 2$ subspace iterations and an oversampling parameter of about $p = 10$ achieves near-optimal results. Algorithm 1 summarizes the computational steps.

3.1.1. Expressivity analysis.
The average behavior of the randomized tensor algorithm is characterized using the expected residual error

$$E\|\boldsymbol{\mathcal{E}}\|_F = \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_F, \tag{10}$$

where $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{Q}_1\mathbf{Q}_1^{\top} \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^{\top}$. Theorem 1 generalizes the matrix version of Theorem 10.5 formulated by Halko *et al* (2011) to the tensor case.

Theorem 1. *Consider a low-rank real N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. Then the expected approximation error, given a target rank $k \geq 2$ and an oversampling parameter $p \geq 2$ for each mode, is*

$$\mathbb{E}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}} \times_1 \mathbf{Q}_1\mathbf{Q}_1^{\top} \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^{\top}\|_F \quad \leq \quad \sqrt{1 + \frac{k}{p-1}} \cdot \sqrt{\sum_{n=1}^{N}\sum_{j>k}\sigma_{nj}^2}, \tag{11}$$

where σ_{nj} denotes the j singular value of the mode-n unfolding of the source tensor $\boldsymbol{\mathcal{X}}$.

For the proof see appendix A. Intuitively, the projection of each tensor mode onto a low-dimensional space introduces an additional residual. This is expressed by the double sum on the right hand side. If the low-rank approximation captures the column space of each mode accurately, then the singular values $j > k$ for each mode n are small. Moreover, the error can be improved using the oversampling parameter. The computation of additional power (subspace) iterations can improve the error further. This result again follows by generalizing the results of Halko *et al* (2011) to tensors. Sharper performance bounds for both oversampling and additional power iterations can be derived following, for instance, the results by Witten and Candes (2015).

3.2. Optimization strategies

There are several optimization strategies for minimizing the objective function defined in (12), of which we consider, alternating least squares (ALS) and block coordinate descent (BCD). Both methods are suitable to operate on a compressed tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{k \times \cdots \times k}$, where $k \geq R$. The optimization problem (5) is reformulated

$$\underset{\hat{\boldsymbol{\mathcal{B}}}}{\text{minimize}} \quad \|\boldsymbol{\mathcal{B}} - \hat{\boldsymbol{\mathcal{B}}}\|_F^2 \quad \text{subject to} \quad \hat{\boldsymbol{\mathcal{B}}} = \sum_{r=1}^{R} \tilde{\mathbf{a}}_r \circ \tilde{\mathbf{b}}_r \circ \tilde{\mathbf{c}}_r. \tag{12}$$

Algorithm 1 A prototype randomized tensor compression algorithm.

Require: An N-way tensor $\boldsymbol{\mathcal{X}}$, and a desired target rank k.
Optional: Parameters p and q to control oversampling, and power iterations.

(1) $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{X}}$ initialize compressed tensor
(2) **for** $n = 1, \ldots, N$ iterateoveralltensormodes
(3) $l = k + p$ slight over sampling
(4) $I, J = \texttt{dim}(\boldsymbol{\mathcal{B}}_{(n)})$ dimension of the $n-$th tensor mode
(5) $= \texttt{rand}(J, l)$ generate random test matrix
(6) $\mathbf{Y} = \boldsymbol{\mathcal{B}}_{(n)}$ compute sampling matrix
(7) **for** $j = 1, \ldots, q$ normalized power iterations (optional)
(8) $[\mathbf{Q}, \sim] = \texttt{lu}(\mathbf{Y})$
(9) $[\mathbf{Z}, \sim] = \texttt{lu}(\boldsymbol{\mathcal{B}}_{(n)}^\top \mathbf{Q})$
(10) $\mathbf{Y} = \boldsymbol{\mathcal{B}}_{(n)} \mathbf{Z}$
(11) **end for**
(12) $[\mathbf{Q}_n, \sim] = \texttt{qr}(\mathbf{Y})$ orthonormalize sampling matrix
(13) $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}} \times_n \mathbf{Q}_n^\top$ project tensor to smaller space
(14) **end for**

Return: Compressed tensor $\boldsymbol{\mathcal{B}}$ of dimension $l \times \cdots \times l$, and a set of orthonormal basis matrices $\{\mathbf{Q}_n \in \mathbb{R}^{I_n \times l}\}_{n=1}^N$.

Remark 1 *Due to the convenient mathematical properties of the normal distribution, a Gaussian random test matrix is often assumed in theoretical results, however, in practice a uniform distributed random test matrix is sufficient. The performance can be further improved using structured random matrices (Woolfe et al 2008). If information is uniformly distributed across the data, randomly selected columns can also be used to build a suitable basis as well, which avoids the matrix multiplication in step (6).*

Remark 2 *For numerical stability, normalized power iterations using the pivoted LU decomposition are computed in step 7-11. We recommend a default value of $q = 2$.*

Remark 3 *In practice, the user can decide which modes to compress and specify different oversampling parameters for these modes.*

Once the compressed factor matrices $\tilde{\mathbf{A}} \in \mathbb{R}^{k \times R}$, $\tilde{\mathbf{B}} \in \mathbb{R}^{k \times R}$, $\tilde{\mathbf{C}} \in \mathbb{R}^{k \times R}$ are estimated, the full factor matrices can be recovered

$$\mathbf{A} \approx \mathbf{Q}_1 \tilde{\mathbf{A}}, \ \mathbf{B} \approx \mathbf{Q}_2 \tilde{\mathbf{B}}, \ \mathbf{C} \approx \mathbf{Q}_3 \tilde{\mathbf{C}}, \tag{13}$$

where $\mathbf{Q}_1 \in \mathbb{R}^{I \times k}$, $\mathbf{Q}_2 \in \mathbb{R}^{J \times k}$, $\mathbf{Q}_3 \in \mathbb{R}^{K \times k}$ denote the orthonormal basis matrices. For simplicity we focus on third order tensors, but the result generalizes to N-way tensors.

3.2.1. ALS algorithm.

Due to its simplicity and efficiency, ALS is the most popular method for computing the CP decomposition (Comon *et al* 2009, Kolda and Bader 2009). We note that the optimization (12) is equivalent to

$$\underset{\mathfrak{A}, \mathfrak{B}, \mathfrak{C}}{\text{minimize}} \quad \|\boldsymbol{\mathcal{B}} - \sum_{r=1}^{R} \tilde{\mathbf{a}}_r \circ \tilde{\mathbf{b}}_r \circ \tilde{\mathbf{c}}_r\|_F^2$$

with respect to the factor matrices $\mathfrak{A}, \mathfrak{B}$ and $\mathfrak{C}$. The tensor $\boldsymbol{\mathcal{B}}$ can further be expressed in matricized form

$$\boldsymbol{\mathcal{B}}_{(1)} \approx \tilde{\mathbf{A}}(\tilde{\mathbf{C}} \odot \tilde{\mathbf{B}})^\top, \ \boldsymbol{\mathcal{B}}_{(2)} \approx \tilde{\mathbf{B}}(\tilde{\mathbf{C}} \odot \tilde{\mathbf{A}})^\top, \ \boldsymbol{\mathcal{B}}_{(3)} \approx \tilde{\mathbf{C}}(\tilde{\mathbf{B}} \odot \tilde{\mathbf{A}})^\top,$$

where $\odot$ denotes the Khatri-Rao product. The optimization problem in this form is non-convex. However, an estimate for the factor matrices can be obtained using the least-squares method as follows. The ALS algorithm updates one component, while holding the other two components fixed, in an alternating fashion until convergence. It iterates over the following subproblems

$$\tilde{\mathbf{A}}^{j+1} = \underset{\tilde{\mathbf{A}}}{\text{argmin}} \|\boldsymbol{\mathcal{B}}_{(1)} - \tilde{\mathbf{A}}(\tilde{\mathbf{C}}^j \odot \tilde{\mathbf{B}}^j)^\top\|, \tag{14}$$

$$\tilde{\mathbf{B}}^{j+1} = \underset{\tilde{\mathbf{B}}}{\text{argmin}} \|\boldsymbol{\mathcal{B}}_{(2)} - \tilde{\mathbf{B}}(\tilde{\mathbf{C}}^j \odot \tilde{\mathbf{A}}^{j+1})^\top\|, \tag{15}$$

$$\tilde{\mathbf{C}}^{j+1} = \underset{\tilde{\mathbf{C}}}{\text{argmin}} \|\boldsymbol{\mathcal{B}}_{(3)} - \tilde{\mathbf{C}}(\tilde{\mathbf{B}}^{j+1} \odot \tilde{\mathbf{A}}^{j+1})^\top\|. \tag{16}$$

Algorithm 2 A prototype randomized CP algorithm using ALS.

Require: An $I \times J \times K$ tensor $\boldsymbol{\mathcal{X}}$, and a desired target rank R.
Optional: Parameters p and q to control oversampling, and power iterations.

(1) $\boldsymbol{\mathcal{B}}, \mathbf{Q}_A, \mathbf{Q}_B, \mathbf{Q}_C = \texttt{compress}(\boldsymbol{\mathcal{X}}, R, p, q)$ — compress tensor using algorithm
(2) $\mathbf{B}, \mathbf{C} = [\texttt{eig}(\boldsymbol{\mathcal{B}}_{(2)}, \boldsymbol{\mathcal{B}}_{(2)}^\top), \texttt{eig}(\boldsymbol{\mathcal{B}}_{(3)}, \boldsymbol{\mathcal{B}}_{(3)}^\top)]$ — use first R eigenvectors for initialization
(3) **repeat**
(4) $\quad \mathbf{A} = \boldsymbol{\mathcal{B}}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^\top\mathbf{C} * \mathbf{B}^\top\mathbf{B})^\dagger$
(5) $\quad \mathbf{A} = \mathbf{A}/\texttt{norm}(\mathbf{A})$
(6) $\quad \mathbf{B} = \boldsymbol{\mathcal{B}}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^\top\mathbf{C} * \mathbf{A}^\top\mathbf{A})^\dagger$
(7) $\quad \mathbf{B} = \mathbf{B}/\texttt{norm}(\mathbf{B})$
(8) $\quad \mathbf{C} = \boldsymbol{\mathcal{B}}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^\top\mathbf{B} * \mathbf{A}^\top\mathbf{A})^\dagger$
(9) $\quad \boldsymbol{\lambda} = \texttt{norm}(\mathbf{C})$
(10) $\quad \mathbf{C} = \mathbf{C}/\boldsymbol{\lambda}$
(11) **until** convergence criterion is reached
(12) $\mathbf{A}, \mathbf{B}, \mathbf{C} = [\mathbf{Q}_A\mathbf{A}, \mathbf{Q}_B\mathbf{B}, \mathbf{Q}_C\mathbf{C}]$ — recover factor matrices
(13) re-normalize the factor matrices and update the scaling vector $\boldsymbol{\lambda}$

Return: Normalized factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the scaling vector $\boldsymbol{\lambda}$.

Each step therefore involves a least-squares problem which can be solved using the Khatri-Rao product pseudo-inverse. Algorithm 2 summarizes the computational steps.

Definition 2. *The Khatri-Rao product pseudo-inverse is defined as*

$$(\mathbf{A} \odot \mathbf{B})^\dagger = (\mathbf{A}^\top\mathbf{A} * \mathbf{B}^\top\mathbf{B})^\dagger(\mathbf{A} \odot \mathbf{B})^\top,$$

*where the operator * denotes the Hadamard product, i.e. the elementwise multiplication of two equal sized matrices.*

There exist few general convergence guarantees for the ALS algorithm (Uschmajew 2012, Wang and Chu 2014). Moreover, the final solution tends to depend on the initial guess $\mathbf{A}^0, \mathbf{B}^0$ and $\mathbf{C}^0$. A standard initial guess uses the eigenvectors of $\boldsymbol{\mathcal{B}}_{(1)}\boldsymbol{\mathcal{B}}_{(1)}^\top$, $\boldsymbol{\mathcal{B}}_{(2)}\boldsymbol{\mathcal{B}}_{(2)}^\top$, $\boldsymbol{\mathcal{B}}_{(3)}\boldsymbol{\mathcal{B}}_{(3)}^\top$ (Bader and Kolda 2015). Further, it is important to note that normalization of the factor matrices is necessary after each iteration to achieve good convergence. This prevents singularities of the Khatri-Rao product pseudo-inverse Kolda and Bader (2009). The algorithm can be further improved by reformulating the above subproblems as regularized least-squares problems, see for instance Li *et al* (2013) for technical details and convergence results. The structure imposed by the ALS algorithm on the factor matrices permits the formulation of non-negative, or sparsity-constrained tensor decompositions (Cichocki *et al* 2009).

3.2.2. BCD algorithm.
While ALS is the most popular algorithm for computing the CP decomposition, many alternative algorithms have been developed. One alternate approach is based on block coordinate descent, BCD (Xu and Yin 2013). Cichocki and Phan (2009) first proposed this approach for computing non-negative tensor factorizations. The BCD algorithm is based on the idea of successive rank-one deflation. Unlike ALS, which updates the entire factor matrix at each step, BCD computes the rank-1 tensors in a hierarchical fashion. Therefore, the algorithm treats each component $\mathbf{A}_r, \mathbf{b}_r, \mathbf{c}_r$ as a block. First, the most correlated rank-1 tensor is computed; then the second most correlated rank-1 tensor is learned on the residual tensor, and so on. Assuming that $\tilde{R} = r - 1$ components have been computed, then the r-th compressed residual tensor $\boldsymbol{\mathcal{Y}}_{\text{res}}$ is defined

$$\boldsymbol{\mathcal{Y}}_{\text{res}} = \boldsymbol{\mathcal{B}} - \sum_{r=1}^{\tilde{R}} \tilde{\mathbf{a}}_r \circ \tilde{\mathbf{b}}_r \circ \tilde{\mathbf{c}}_r. \tag{17}$$

The algorithm then iterates over the following subproblems

$$\tilde{\mathbf{a}}_r^{j+1} = \underset{\tilde{\mathbf{a}}_r}{\text{argmin}} \|\boldsymbol{\mathcal{Y}}_{\text{res}(1)} - \tilde{\mathbf{a}}_r(\tilde{\mathbf{c}}_r^j \odot \tilde{\mathbf{b}}_r^j)^\top\|, \tag{18}$$

Algorithm 3 A prototype randomized CP algorithm using BCD.

Require: An $I \times J \times K$ tensor $\mathcal{X}$, and a desired target rank R.
Optional: Parameters p and q to control oversampling, and power iterations.

(1) $\mathcal{B}, \mathbf{Q}_A, \mathbf{Q}_B, \mathbf{Q}_C = \texttt{compress}(\mathcal{X}, R, p, q)$ — compress tensor using Algorithm 1
(2) $\mathbf{B}, \mathbf{C} = [\texttt{eig}(\mathcal{B}_{(2)}, \mathcal{B}_{(2)}^\top), \texttt{eig}(\mathcal{B}_{(3)}, \mathcal{B}_{(3)}^\top)]$ — use first R eigenvectors for initialization
(3) $\mathcal{Y} = \mathcal{B}$ — initialize residual tensor
(4) **for** $r = 1, \ldots, R$ — compute rank−r approximation
(5) **repeat**
(6) $\quad \mathbf{a}_r = \mathcal{Y}_{(1)}(\mathbf{c}_r \odot \mathbf{b}_r)(\mathbf{c}_r^\top \mathbf{c}_r * \mathbf{b}_r^\top \mathbf{b}_r)^\dagger$
(7) $\quad \mathbf{a}_r = \mathbf{a}_r / \texttt{norm}(\mathbf{a}_r)$
(8) $\quad \mathbf{b}_r = \mathcal{Y}_{(2)}(\mathbf{c}_r \odot \mathbf{a}_r)(\mathbf{c}_r^\top \mathbf{c}_r * \mathbf{a}_r^\top \mathbf{a}_r)^\dagger$
(9) $\quad \mathbf{b}_r = \mathbf{b}_r / \texttt{norm}(\mathbf{b}_r)$
(10) $\quad \mathbf{c}_r = \mathcal{Y}_{(3)}(\mathbf{b}_r \odot \mathbf{a}_r)(\mathbf{b}_r^\top \mathbf{b}_r * \mathbf{a}_r^\top \mathbf{a}_r)^\dagger$
(11) $\quad \boldsymbol{\lambda}_r = \texttt{norm}(\mathbf{c}_r)$
(12) $\quad \mathbf{c}_r = \mathbf{c}_r / \boldsymbol{\lambda}_r$
(13) **until** convergence criterion is reached
(14) $\mathcal{Y} = \mathcal{B} - [\![\boldsymbol{\lambda}_{[1:r]}; \mathbf{A}_{[:,1:r]}, \mathbf{B}_{[:,1:r]}, \mathbf{C}_{[:,1:r]}]\!]$ — update residual tensor
(15) **end for**
(16) $\mathbf{A}, \mathbf{B}, \mathbf{C} = [\mathbf{Q}_A \mathbf{A}, \mathbf{Q}_B \mathbf{B}, \mathbf{Q}_C \mathbf{C}]$ — recover factor matrices
(17) re-normalize the factor matrices and update the scaling vector $\boldsymbol{\lambda}$

Return: Normalized factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the scaling vector $\boldsymbol{\lambda}$.

$$\tilde{\mathbf{b}}_r^{j+1} = \underset{\tilde{\mathbf{b}}_r}{\operatorname{argmin}} \|\mathcal{Y}_{\text{res}(2)} - \tilde{\mathbf{b}}_r(\tilde{\mathbf{c}}_r^j \odot \tilde{\mathbf{a}}_r^{j+1})^\top\|, \tag{19}$$

$$\tilde{\mathbf{c}}_r^{j+1} = \underset{\tilde{\mathbf{c}}_r}{\operatorname{argmin}} \|\mathcal{Y}_{\text{res}(3)} - \tilde{\mathbf{c}}_r(\tilde{\mathbf{b}}_r^{j+1} \odot \tilde{\mathbf{a}}_r^{j+1})^\top\|. \tag{20}$$

Note that the computation can be more efficiently evaluated without explicitly constructing the residual tensor $\mathcal{Y}_{\text{res}}$ (Kim *et al* 2014). Algorithm 3 summarizes the computation.

3.3. Implementation details

The algorithms we present are implemented in the programming language *Python*, using numerical linear algebra tools provided by the *SciPy* (Open Source Library of Scientific Tools) package (Jones *et al* 2001). *SciPy* provides MKL (Math Kernel Library) accelerated high performance implementations of *BLAS* and *LAPACK* routines. Thus, all linear algebra operations are threaded and highly optimized on Intel processors. The implementation of the CP decomposition follows the *MATLAB Tensor Toolbox* implementation (Bader and Kolda 2015). This implementation normalizes the components after each step to achieve better convergence. Furthermore, we use eigenvectors (see above) to initialize the factor matrices. Interestingly, randomly initialized factor matrices have the ability to achieve slightly better approximation errors, but re-running the algorithms several times with different random seeds can display significant variance in the results. Hence only the former approach is used for initialization. We note that the randomized algorithm introduces some randomness and slight variations into the CP decompositions as well. However, randomization can also act as an implicit regularization on the CP decomposition (Mahoney 2011), meaning that the results of the randomized algorithm can be in some cases even 'better' than the results of the corresponding deterministic implementation. Finally, the convergence criterion is defined as the change in fit, following Bader and Kolda (2015). The algorithm therefore stops when the improvement of the fit ρ is less then a predefined threshold, where the fit is computed using $\rho = 1 - (\|\mathcal{X}\|_F^2 + \|\hat{\mathcal{X}}\|_F^2 - 2 \cdot \langle \hat{\mathcal{X}}, \mathcal{X} \rangle)/\|\mathcal{X}\|_F^2$.

4. Numerical Results

The randomized CP algorithm is evaluated on a number of examples where the near optimal approximation of massive tensors can be achieved in a fraction of the time using the randomized algorithm. Approximation accuracy is computed with the relative error $\|\mathcal{X} - \hat{\mathcal{X}}\|_F / \|\mathcal{X}\|_F$, where $\hat{\mathcal{X}}$ denotes the approximated tensor.

4.1. Computational performance

The robustness of the randomized CP algorithm is first assessed on random low-rank tensors. Here we illustrate the approximation quality in the presence of additive white noise. Figure 4 shows the average relative errors over 100 runs for varying signal-to-noise ratios (SNR). In the case of high SNRs, all algorithms

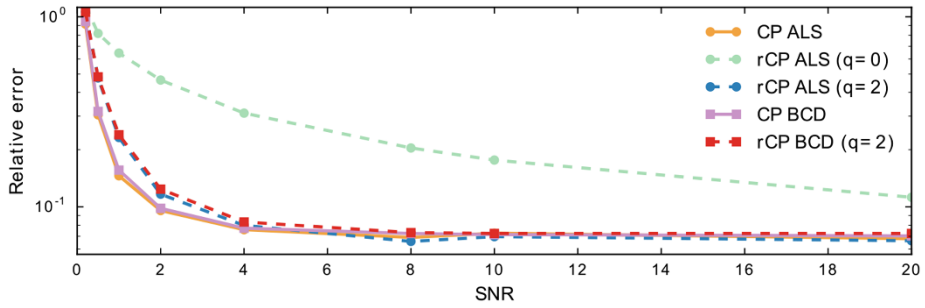


Figure 4. Average relative error, plotted on a log scale, against increasing signal to noise ratio. The analysis is performed on a rank $R = 50$ tensor of dimension $100 \times 100 \times 100$. Power iterations improve the approximation accuracy considerably, while the performance without power iterations is poor.

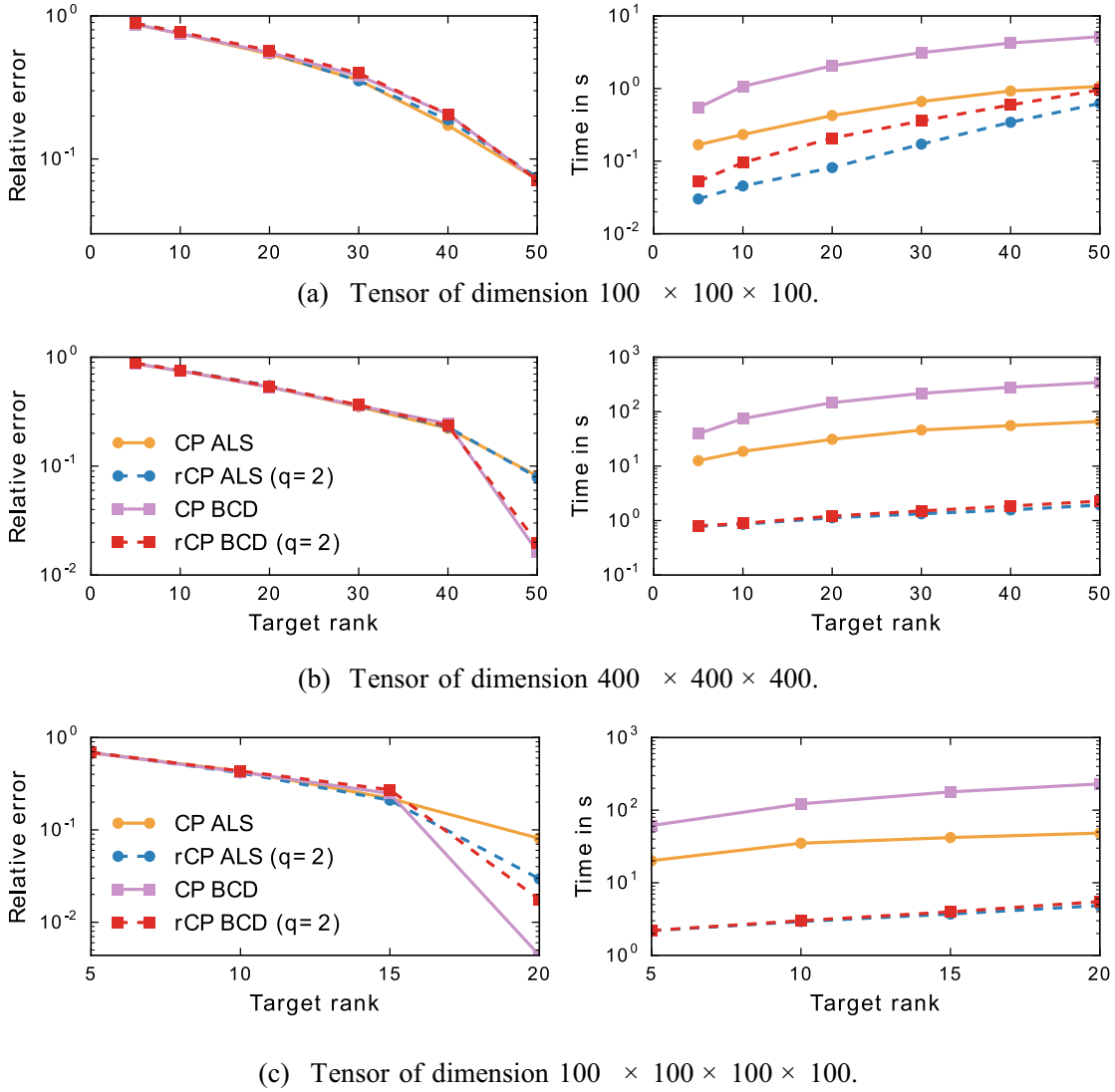


Figure 5. Random tensor approximation and performance for rank $R = 50$ tensors: rCP methods achieve speedups by 1-2 orders of magnitude and the same accuracy as their deterministic counterpart.

converge towards the same relative error. However, at excessive levels of noise (i.e. SNR< 4) the deterministic CP algorithms exhibit small gains in accuracy over the randomized algorithms using $q = 2$ power iterations, with which both the ALS and BCD algorithm show the same performance. The performance of the randomized algorithm without power iterations ($q = 0$) is, however, poor, and stresses the importance of the power operation for real applications. The oversampling parameter for the randomized algorithms is set to $p = 10$. Increasing p can further improve accuracy.

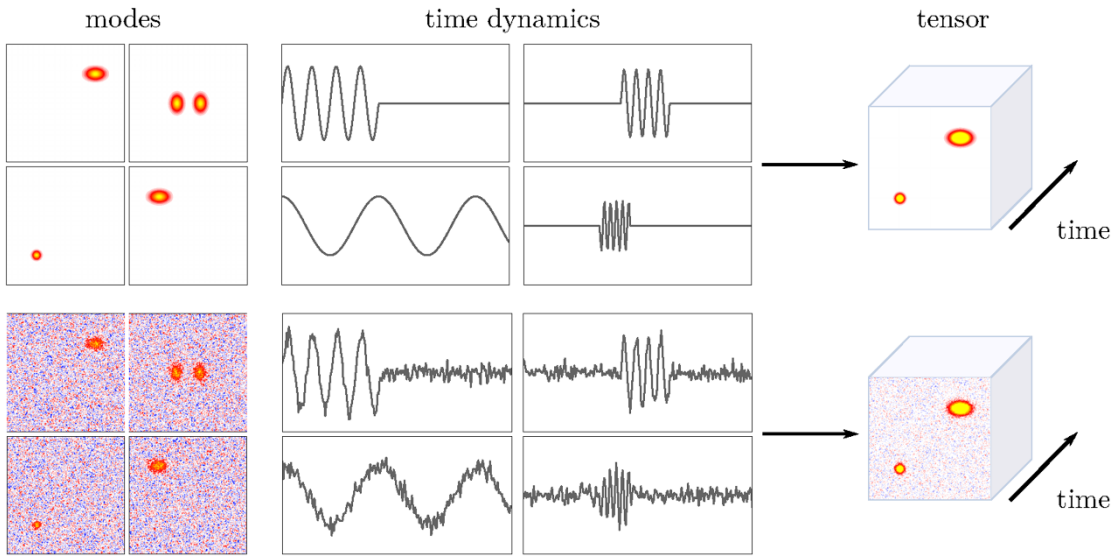


Figure 6. Illustration of the multiscale toy video. The system is governed by four spatial modes experiencing intermittent oscillations in the temporal direction. The top row shows the clean system, while the bottom row shows the noisy system with a signal-to-noise ratio of 2.

Next, the reconstruction errors and runtimes for tensors of varying dimensions are compared. Figure 5 shows the average evaluation results over 100 runs for random low-rank tensors of different dimensions, and for varying target ranks k. The randomized algorithms achieve near optimal approximation accuracy while demonstrating substantial computational savings. Interestingly, the relative error achieved by the BCD decreases sharply by about one order of magnitude when the target rank k matches the actual tensor rank (here $R = 50$).

The computational advantage becomes more pronounced with increasing tensor dimensions, and as the number of iterations required for convergence increase. Using random tensors as presented here, all algorithms rapidly converge after about 4 to 6 iterations. However, it is evident that the computational cost per iteration of the randomized algorithm is substantially lower. Thus, the computational savings can be even more substantial in real world applications that may require several hundred iterations to converge. Overall, the ALS algorithm is computationally more efficient than BCD, i.e. the deterministic ALS algorithm is faster than the BCD by nearly one order of magnitude, while the randomized algorithms exhibit similar computational timings.

Similar performance results are achieved for higher order tensors. Figure 5(c) shows the computational performance for a 4-way tensor of dimension $100 \times 100 \times 100 \times 100$. Again, the randomized algorithms achieve speedups of 1-2 orders of magnitude, while attaining good approximation errors. Once more, the BCD algorithm achieves better approximation at the true tensor rank.

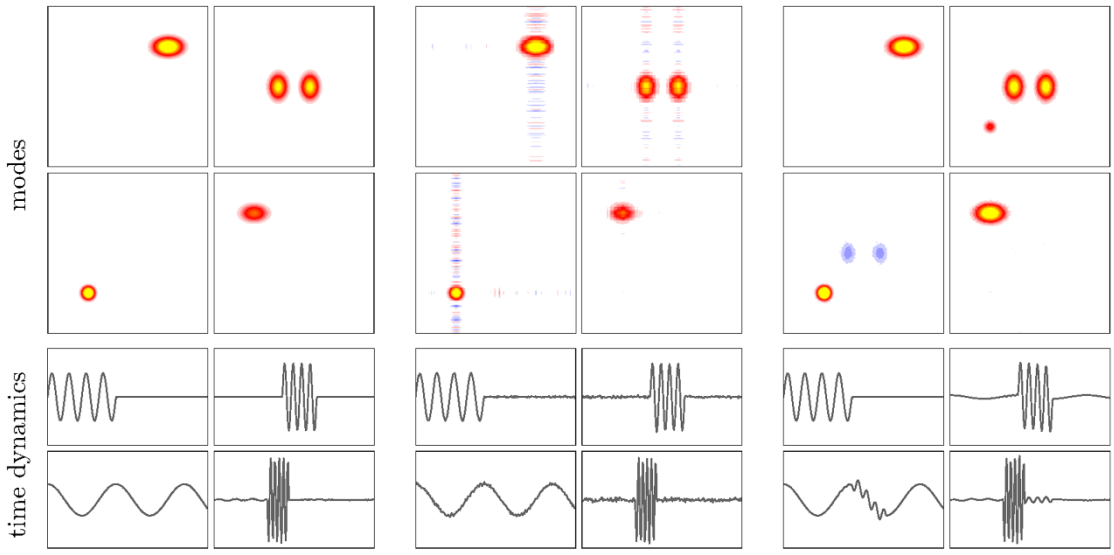
4.2. Examples

The examples demonstrate the advantages of the randomized CP decomposition. The first is a multiscale toy video example, and the second is a simulated flow field behind a stationary cylinder. Due to the better and more natural interpretability of the BCD algorithm, only this algorithm is considered in subsequent sections.

4.2.1. Multiscale toy video example.

The approximation of the underlying spatial modes and temporal dynamics of a system is a common problem in signal processing. In the following, we consider a toy example presenting multiscale intermittent dynamics in the time direction. The data consists of four Gaussian modes, each undergoing different frequencies of intermittent oscillation, for 215 times steps in the temporal direction on a two-dimensional spatial grid (200×200). The resulting tensor is of dimension $200 \times 200 \times 215$. Figure 6 shows the corresponding modes and the time dynamics. This problem becomes even more challenging when the underlying structure needs to be reconstructed from noisy measurements. Traditional matrix decomposition techniques such as the SVD are known to face difficulties capturing such intermittent, multiscale structure.

Figure 7 displays the decomposition results of the noisy (SNR $= 2$) toy video for both the randomized CP decomposition and the SVD. The first subplot shows the results of a rank $k = 4$ approximation computed using the rCP algorithm with $q = 2$ power iterations, and a small oversampling parameter $p = 10$. The



(A) rCP ($q = 2$). (B) rCP ($q = 0$). (C) SVD

Figure 7. Toy video decomposition results. The randomized CP with power iterations ($q = 2$) accurately reconstructs the original spatiotemporal dynamics from noise-corrupted data, while SVD and rCP without power iterations ($q = 0$) yield poor reconstruction results.

Table 1. Summary of the computational results for the noisy toy video.

	Parameters	Time (s)	Speedup	Iterations	Error
CP BCD	$k = 4$	2.31	—	9	0.017 1
rCP BCD	$k = 4, p = 10, q = 0$	0.13	17	9	0.494
	$k = 4, p = 10, q = 1$	0.14	16	10	0.019 1
	$k = 4, p = 10, q = 2$	0.15	15	10	0.016 4
SVD	$k = 4$	0.52	—	—	0.137

method faithfully captures the underlying spatial modes and the time dynamics. For illustration, the second subplot shows the decomposition results without additional power iterations. It can clearly be seen that this approach introduces distinct artifacts, and the approximation quality is relatively poor. The SVD, shown in the last subplot, demonstrates poor performance at separating the modes and mixes the spatiotemporal dynamics of modes 2 & 3.

Table 1 further quantifies the observed results. Interestingly, the relative error using the randomized algorithm with $q = 2$ power iterations is slightly better than the deterministic algorithm. This is due to the intrinsic regularizing behavior of randomization. However, the reconstruction error without power iterations is large, as is the error resulting from the SVD.

The achieved speedup of randomized CP is substantial, with a speedup factor of about 15.

4.2.1.1. A note on compression.

The CP decomposition provides a more parsimonious representation of the data. Comparing the compression ratios between the CP decomposition and SVD illustrates the difference. For a rank $R = 4$ tensor of dimension $100 \times 100 \times 100$, the compression ratios are

$$c_{SVD} = \frac{I \cdot J \cdot K}{R \cdot (I \cdot J + K + 1)} = \frac{100^3}{4 \cdot (100^2 + 100 + 1)} \approx 24.75,$$
$$c_{CP} = \frac{I \cdot J \cdot K}{R \cdot (I + J + K + 1)} = \frac{100^3}{4 \cdot (100 + 100 + 100 + 1)} \approx 830.56.$$

Note, the SVD requires the tensor to be reshaped in some direction. The comparison illustrates the striking difference between compression ratios. It is evident that the CP decomposition requires computing many fewer coefficients in order to approximate the tensor. Thus, less storage is required to approximate the data. This can be of importance if the bandwidth is constrained and only a limited amount of information can be transmitted, in which case the CP decomposition may be advantageous. While the CP decomposition

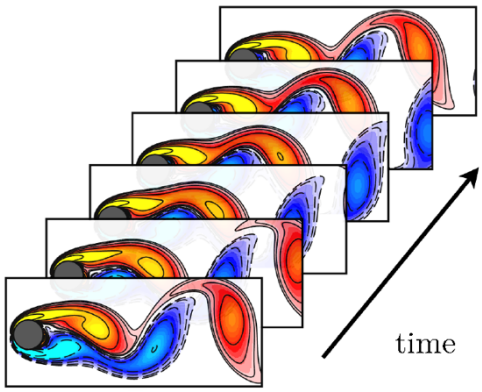


Figure 8. Snapshots of the fluid flow behind a cylinder.

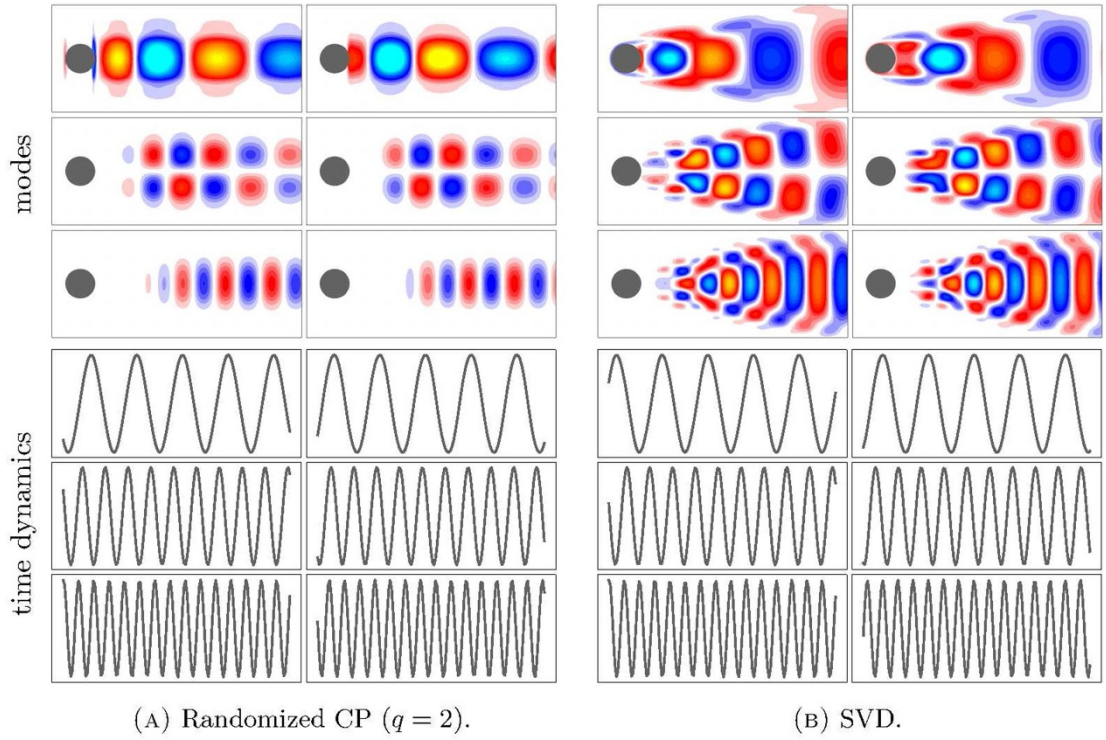


Figure 9. Fluid flow decomposition, no noise. Both methods capture the same dominant frequencies from the time dynamics, while randomized CP requires more rank-1 outer products in x and y to represent single frequency spatial dynamics.

yields a parsimonious approximation that is more robust to noise, the advantage of the SVD is that data can be approximated with an accuracy as low as machine precision.

4.2.2. Flow behind a cylinder.

Extracting the dominant coherent structures from fluid flows helps to better characterize them for modeling and control (Brunton and Noack 2015). The workhorse algorithm in fluid dynamics and for model reduction is the SVD. However, fluid simulations generate high-resolution spatiotemporal grids of data which naturally manifest as tensors. In the following we examine the suitability of the CP decomposition for decomposing flow data, and compare the results to those of the SVD.

The fluid flow behind a cylinder, a canonical example in fluid dynamics, is presented here. The data are a time-series generated from a simulation of fluid vorticity behind a stationary cylinder (Colonius and Taira 2008). The corresponding flow tensor has dimension $199 \times 449 \times 151$, consisting of 151 snapshots on a 449×199 spatial grid. Figure 8 shows a few example snapshots of the fluid flow.

Table 2. Summary of the computational results for the noise-free cylinder flow.

	Parameters	Time (s)	Speedup	Iterations	Error
CP BCD	$k = 30$	115.55	–	458	0.117
rCP BCD	$k = 30, p = 10, q = 0$	1.27	91	533	0.122
	$k = 30, p = 10, q = 1$	1.41	82	517	0.121
	$k = 30, p = 10, q = 2$	1.56	74	437	0.118
SVD	$k = 30$	0.57	–	–	4.25×10^{-05}

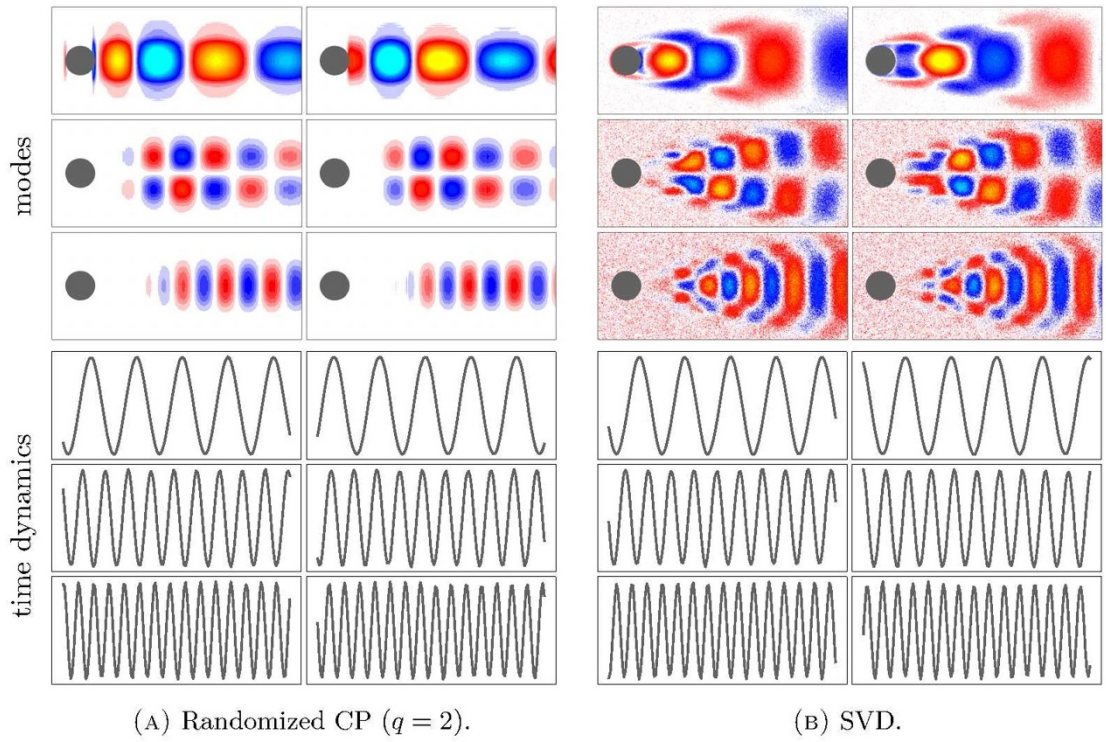


Figure 10. Fluid flow decomposition noisy (SNR = 2). Randomized CP modes are robust to additive noise and SVD spatial modes are corrupted by noise.

4.2.2.1. Results in absence of white noise.

Figure 9 shows both the approximated spatial modes and the temporal dynamics for the randomized CP decomposition and the SVD. Observe that both methods extract similar patterns, although unlike SVD, rCP spatial modes are sparse on the domain. The reason is two-fold: (i) CP does not impose orthogonality constraints (which in general reveal dense structure), and (ii) CP imposes rank-one outer product structure in the x and y directions via the columns of the factor matrices. In doing so, CP isolates the contributions of single wavenumbers (spatial frequencies) to the steady-state vortex shedding regime. These are commonly analyzed to study energy transfer between scales in complex flows and turbulence (Meneveau and Katz 2000, Sharma and McKeon 2013), hence CP can help reveal new physically meaningful insights. In particular, randomization enables tensor decomposition of high-resolution vector fields that otherwise may not be tractable using deterministic methods. Thus rCP provides a novel decomposition of multimodal *fields* for downstream tasks in model reduction, pattern extraction and control.

Note, that for a fixed target rank of $k = 30$ across all methods, the SVD achieves a substantially lower reconstruction error (see table 2).

However, the compression ratios for the CP and SVD methods are $c_{CP} \approx 562.17$ and $c_{SVD} \approx 5.02$, i.e. the CP compresses the data nearly two orders of magnitude more.

4.2.2.2. Results in presence of white noise.

Next, the analysis of the same flow is repeated in the presence of additive white noise. While this is not of concern when dealing with flow simulations, it is realistic when dealing with flows obtained from measurement. We chose a signal-to-noise ratio of 2 to demonstrate the robustness of the CP decomposition to noise.

Table 3. Summary of the computational results for the noise-corrupted cylinder flow.

	Parameters	Time (s)	Speedup	Iterations	Error
CP BCD	$k = 30$	64.01	–	239	0.191
rCP BCD	$k = 30, p = 10, q = 0$	0.99	64	332	0.522
	$k = 30, p = 10, q = 1$	1.23	52	414	0.189
	$k = 30, p = 10, q = 2$	1.13	56	370	0.153
SVD	$k = 30$	0.58	–	–	0.655
	$k = 6$	0.58	–	–	0.311

Figure 10 shows again the corresponding dominant spatial modes and temporal dynamics. Both the SVD and the CP decomposition faithfully capture the temporal dynamics. However, comparing the modes of the SVD to figure 9, it is apparent that the spatial modes contain a large amount of noise. The spatial modes revealed by the CP decomposition provide a significantly better approximation. Again, it is crucial to use power iterations to achieve a good approximation quality (see table 3). By inspection, the relative reconstruction error using the SVD is poor compared to the error achieved using the CP decomposition. Here, we show the error for a rank $k = 30$ and $k = 6$ approximation. The target rank was determined using the optimal hard threshold for singular values (Gavish and Donoho 2014).

The CP decomposition overcomes this disadvantage, and is able to approximate the first $k = 30$ modes with only a slight loss of accuracy. Note that here the randomized CP decomposition performs better than the deterministic algorithm. We assume that this is due to the favorable intrinsic regularization effect of randomized methods.

5. Conclusion

The emergence of massive tensors require efficient algorithms for obtaining tensor decompositions.

To address this challenge, we have presented a randomized algorithm which substantially reduces the computational demands of the CP decomposition.

Indeed, randomized algorithms have established themselves as highly competitive methods for computing traditional matrix decompositions. A key advantage of the randomized algorithm is that modern computational architectures are fully exploited. Thus, the algorithm benefits substantially from multithreading in a multi-core processor. In contrast to previously proposed high-performance tensor algorithms which are based on computational concepts such as distributed computing, our proposed randomized algorithm provides substantial computational speedups even on standard desktop computers. Our proposed algorithm achieves these speedups by reducing the computational costs per iteration, which enables the user to decompose real world examples that typically require a large number of iterations to converge.

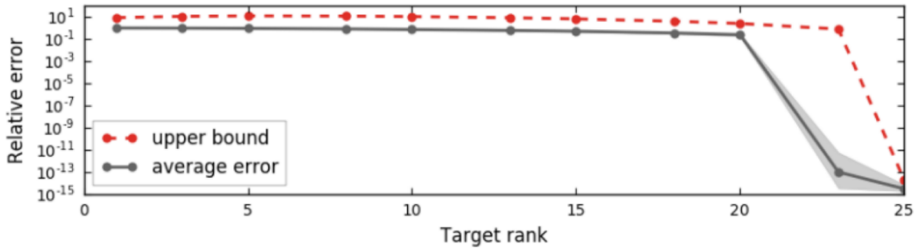
In addition to computational savings, the randomized CP decomposition demonstrates outstanding performance on several examples using artificial and real-world data, including decompositions of high-resolution flow fields that may not be tractable with deterministic methods. Moreover, our experiments show that the power iteration concept is crucial in order to achieve a robust tensor decomposition. Thus, our algorithm has a practical advantage over previous randomized tensor algorithms, at a slightly increased computational cost due to additional power iterations.

Acknowledgments

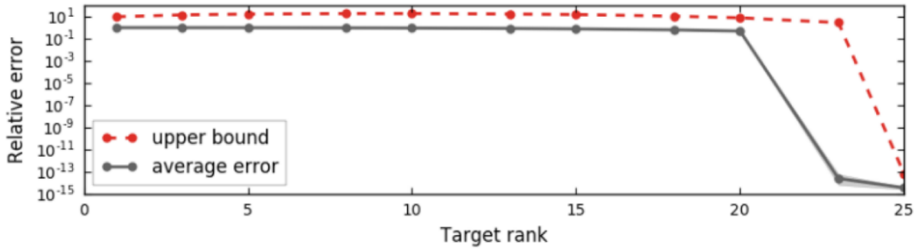
We would like to thank Alex Williams, and Michael W Mahoney for insightful discussion on tensor decompositions and randomized numerical linear algebra. Further, we would like to express our gratitude to the two anonymous reviewers for their valuable feedback, which helped us greatly improve the manuscript. NBE would like to acknowledge DARPA and NSF for providing partial support of this work. KM acknowledges support from NSF MSPRF Award 1803663. JNK acknowledges support from the Air Force Office of Scientific Research (AFOSR) grant FA9550-17-1-0329. SLB acknowledges funding support from the Air Force Office of Scientific Research (AFOSR) grant FA9550-18-1-0200.

Data availability statement

The data that support the findings of this study are available upon request.



(A) Tensor of dimension $50 \times 50 \times 50$.



(B) Tensor of dimension $50 \times 50 \times 50 \times 50$.

Figure A1. Empirical evaluation of the theoretical upper bound.

Appendix A. Proof of theorem 1

In the following, we sketch a proof for Theorem 1, which yields an upper bound for the approximate basis for the range of a tensor.

To assess the quality of the basis matrices $\{\mathbf{Q}_n\}_{n=1}^N$, we first show that the problem can be expressed as a sum of subproblems. Defining the residual error

$$\|\mathcal{E}\|_F = \|\mathcal{X} - \hat{\mathcal{X}}\| = \|\mathcal{X} - \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top\|_F. \tag{A1}$$

Note that the Frobenius norm of a tensor and its matricized forms are equivalent. Defining the orthogonal projector $\mathbf{P}_n \equiv \mathbf{Q}_n\mathbf{Q}_n^\top$, we can reformulate (A1) compactly as

$$\|\mathcal{E}\|_F = \|\mathcal{X} - \mathcal{X} \times_1 \mathbf{P}_1 \times_2 \cdots \times_N \mathbf{P}_N\|_F. \tag{A2}$$

Proof. Assuming that $\mathbf{P}_n$ yields an exact projection onto the column space of the matrix $\mathbf{Q}_n$, we need to show first that the error can be expressed as a sum of the errors of the n projections

$$\|\mathcal{E}\|_F = \sum_{n=1}^N \|\mathcal{X} - \mathcal{X} \times_n \mathbf{P}_n\|_F = \sum_{n=1}^N \|\mathcal{X} \times_n (\mathbf{I} - \mathbf{P}_n)\|_F, \tag{A3}$$

where $\mathbf{I}$ denotes the identity matrix. Following Drineas and Mahoney (2007), let us add and subtract the term $\mathcal{X} \times_N \mathbf{P}_N$ in equation (A2) so that we obtain

$$\|\mathcal{E}\|_F = \|\mathcal{X} - \mathcal{X} \times_N \mathbf{P}_N + \mathcal{X} \times_N \mathbf{P}_N - \mathcal{X} \times_1 \mathbf{P}_1 \times_1 \cdots \times_N \mathbf{P}_N\|_F \tag{A4}$$

$$\leq \|\mathcal{X} - \mathcal{X} \times_N \mathbf{P}_N\|_F + \|\mathcal{X} \times_N \mathbf{P}_N - \mathcal{X} \times_1 \mathbf{P}_1 \times_1 \cdots \times_N \mathbf{P}_N\|_F \tag{A5}$$

$$= \|\mathcal{X} - \mathcal{X} \times_N \mathbf{P}_N\|_F + \|(\mathcal{X} - \mathcal{X} \times_1 \mathbf{P}_1 \times_1 \cdots \times_{N-1} \mathbf{P}_{N-1}) \times_N \mathbf{P}_N\|_F \tag{A6}$$

$$\leq \|\mathcal{X} - \mathcal{X} \times_N \mathbf{P}_N\|_F + \|\mathcal{X} - \mathcal{X} \times_1 \mathbf{P}_1 \times_1 \cdots \times_{N-1} \mathbf{P}_{N-1}\|_F. \tag{A7}$$

The bound (A5) follows from the triangular inequality for a norm. Next, the common term $\mathbf{P}_N$ is factored out in equation (A6). Then, the bound (A7) follows from the properties of orthogonal projectors. This is because the $range(\boldsymbol{\mathcal{X}} \times_1 \mathbf{P}_1 \times_1 \cdots \times_{N-1} \mathbf{P}_{N-1}) \subset range(\boldsymbol{\mathcal{X}} \times_1 \mathbf{P}_1 \times_1 \cdots \times_N \mathbf{P}_N)$, and then it holds that $\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}} \times_1 \mathbf{P}_1 \times_1 \cdots \times_N \mathbf{P}_N\|_F \leq \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}} \times_1 \mathbf{P}_1 \times_1 \cdots \times_{N-1} \mathbf{P}_{N-1}\|_F$. See Proposition 8.5 by Halko *et al* (2011) for a proof using matrices. Subsequently the residual error $\boldsymbol{\mathcal{E}}_{N-1}$ can be bounded

$$\|\boldsymbol{\mathcal{E}}_{N-1}\| \quad \leq \quad \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}} \times_{N-1} \mathbf{P}_{N-1}\|_F + \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}} \times_1 \mathbf{P}_1 \times_1 \cdots \times_{N-2} \mathbf{P}_{N-2})\|_F. \tag{A8}$$

From this inequality, equation (A3) follows. We take the expectation of equation (A3)

$$E\|\boldsymbol{\mathcal{E}}\|_F = E\left[\sum_{n=1}^{N} \|\boldsymbol{\mathcal{X}} \times_n (\mathbf{I} - \mathbf{P}_n)\|_F\right]. \tag{A9}$$

Recalling that Theorem 10.5 formulated by Halko *et al* (2011) states the following expected approximation error (formulated here using tensor notation)

$$E\|\boldsymbol{\mathcal{X}} \times_n (\mathbf{I} - \mathbf{P}_n)\|_F \leq \sqrt{1 + \frac{k}{p-1}} \cdot \sqrt{\sum_{j>k} \sigma_j^2}, \tag{A10}$$

assuming that the sketch in equation (7) is constructed using a standard Gaussian matrix . Here σ_j denotes the singular values of the matricized tensor $\boldsymbol{\mathcal{X}}_{(n)}$ greater then the chosen target rank k. Combining Equations (A9) and (A10) then yields the results of the theorem (11).

□

Figures A1 evaluates the theoretical upper bound over 100 runs for both a third and fourth order random low-rank $R = 25$ tensor. Here, we use a fixed oversampling parameter $p = 2$. The results show that the empirical error is faithfully bounded by the theoretical upper bound for varying target ranks.

ORCID iD

N Benjamin Erichson https://orcid.org/0000-0003-0667-3516

References

Achlioptas D and McSherry F 2007 Fast computation of low-rank matrix approximations *J. ACM* **54** 9
Bader B W and Kolda T G 2015 MATLAB Tensor Toolbox version 2.6 (www.sandia.gov/ tgkolda/TensorToolbox/)
Battaglino C, Ballard G and Kolda T G 2018 A practical randomized CP tensor decomposition *SIAM J. Matrix Anal. Appl.* **39** 876–901
Brunton S L and Noack B R 2015 Closed-loop turbulence control: Progress and challenges *Appl. Mech. Rev.* **67** 1–60
Carroll J D and Chang J-J 1970 Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition *Psychometrika* **35** 283–319
Cichocki A and Phan A H 2009 Fast local algorithms for large scale nonnegative matrix and tensor factorizations *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **92** 708–21
Cichocki A, Zdunek R, Phan A and Amari S 2009 *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation* (New York: Wiley)
Colonius T and Taira K 2008 A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions *Comput. Methods Appl. Mech. Eng.* **197** 2131–46
Comon P, Luciani X and DeAlmeida A 2009 Tensor decompositions, alternating least squares and other tales *J. Chemomet.* **23** 393–405
Drineas P and Mahoney M W 2007 A randomized algorithm for a tensor-based generalization of the singular value decomposition *Linear Algebr. Appl.* **420** 553–571
Drineas P and Mahoney M W 2016 RandNLA: Randomized numerical linear algebra *Commun. ACM* **59** 80–90
Erichson N B, Voronin S, Brunton S L, and Kutz J N 2019 Randomized matrix decompositions using R *J. Stat. Softw.* **89** 1–48
Frieze A, Kannan R and Vempala S 2004 Fast Monte-Carlo algorithms for finding low-rank approximations *J. ACM* **51** 1025–41
Gavish M and Donoho D L 2014 The optimal hard threshold for singular values is $4/\sqrt{3}$ *IEEE Trans. Inf. Theory* **60** 5040–53
Gu M 2015 Subspace iteration randomization and singular value problems *SIAM J. Sci. Comput.* **37** 1139–73
Halko N, Martinsson P G and Tropp J A 2011 Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions *SIAM Rev.* **53** 217–88
Harshman R A 1970 Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis *Technical Report* No. 16, Working Papers in Phonetics, UCLA
Hitchcock F L 1927 The expression of a tensor or a polyadic as a sum of products *J. Math. Phys.* **6** 164–89
Hong D, Kolda T G and Duersch J A 2020 Generalized canonical polyadic tensor decomposition *SIAM Rev.* **62** 133–63
Jones E, Oliphant T and Peterson P 2001 SciPy: Open source scientific tools for Python (https://www.scipy.org/)
Kim J, He Y and Park H 2014 Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework *J. Global Opt.* **58** 285–319
Kolda T G and Bader B W 2009 Tensor decompositions and applications *SIAM Rev.* **51** 455–500

Li N, Kindermann S and Navasca C 2013 Some convergence results on the regularized alternating least-squares method for tensor decomposition *Linear Algebr. Appl.* **438** 796–812
Liberty E, Woolfe F, Martinsson P-G, Rokhlin V and Tygert M 2007 Randomized algorithms for the low-rank approximation of matrices *Proc. Natl. Acad. Sci.* 104 20167–72
Mahoney M W 2011 Randomized algorithms for matrices and data *Found. Trends Mach. Learn.* **3** 123–224
Martinsson P-G 2016 Randomized methods for matrix computations and analysis of high dimensional data. arXiv preprint arXiv:1607.01649
Martinsson P-G, Rokhlin V and Tygert M 2011 A randomized algorithm for the decomposition of matrices *Appl. Comput. Harmon. Anal.* **30** 47–68
Meneveau C and Katz J 2000 Scale-invariance and turbulence models for large-eddy simulation *Ann. Rev. Fluid Mech.* **32** 1–32
Phan A and Cichocki A 2011 PARAFAC algorithms for large-scale problems *Neurocomputing* **74** 1970–84
Rokhlin V, Szlam A and Tygert M 2009 A randomized algorithm for principal component analysis *SIAM J. Matrix Anal. Appl.* **31** 1100–24
Sharma A and McKeon B 2013 On coherent structure in wall turbulence *J. Fluid Mech.* **728** 196–238
Sidiropoulos N, Papalexakis E and Faloutsos C 2014 Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition *IEEE Signal Process. Mag.* **31** 57–70
Szlam A, Kluger Y, and Tygert M 2017 An implementation of a randomized algorithm for principal component analysis *ACM Trans. Math. Softw. (TOMS)* **43** 1–14
Tsourakakis C E 2010 MACH: Fast randomized tensor decompositions *Proc. 2010 SIAM Int. Conf. Data Mining* pp 689–700
Uschmajew A 2012 Local convergence of the alternating least squares algorithm for canonical tensor approximation *SIAM J. Matrix Anal. Appl.* **33** 639–52
Vervliet N and Lathauwer L D 2016 A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors *IEEE J. Sel. Top. Signal Process.* **10** 284–95
Vervliet N, Debals O, Sorber L and Lathauwer L D 2014 Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis *IEEE Signal Process. Mag.* **31** 71–9
Voronin S and Martinsson P-G 2015 RSVDPACK: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures. arXiv preprint arXiv:1502.05366
Wang L and Chu M T 2014 On the global convergence of the alternating least squares method for rank-one approximation to generic tensors *SIAM J. Matrix Anal. Appl.* **35** 1058–72
Witten R and Candes E 2015 Randomized algorithms for low-rank matrix factorizations: sharp performance bounds *Algorithmica* **72** 264–81
Woolfe F, Liberty E, Rokhlin V and Tygert M 2008 A fast randomized algorithm for the approximation of matrices *J. Appl. Computat. Harmonic Anal.* **25** 335–66
Xu Y and Yin W 2013 A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion *SIAM J. Imaging Sci.* **6** 1758–89
Zhou G, Cichocki A and Xie S 2014 Decomposition of big tensors with low multilinear rank. arXiv preprint arXiv:1412.1885