Low-Rank Tensor Completion by Riemannian Optimization[*]

Daniel Kressner[†] Michael Steinlechner[‡] Bart Vandereycken[§]

October 29, 2013

Abstract

In tensor completion, the goal is to fill in missing entries of a partially known tensor under a low-rank constraint. We propose a new algorithm that performs Riemannian optimization techniques on the manifold of tensors of fixed multilinear rank. More specifically, a variant of the nonlinear conjugate gradient method is developed. Paying particular attention to the efficient implementation, our algorithm scales linearly in the size of the tensor. Examples with synthetic data demonstrate good recovery even if the vast majority of the entries are unknown. We illustrate the use of the developed algorithm for the recovery of multidimensional images and for the approximation of multivariate functions.

1 Introduction

This paper is concerned with low-rank completion for tensors in the sense of multidimensional arrays. To be more specific, we aim to solve the tensor completion problem

$$\begin{aligned} \min_{\mathcal{X}} \quad & \frac{1}{2}\| \mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}\|^2 \\ \text{subject to} \quad & \mathcal{X} \in \mathcal{M}_{\mathbf{r}} := \big\{ \mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \,\big|\, \operatorname{rank}(\mathcal{X}) = \mathbf{r} \big\}. \end{aligned} \tag{1}$$

Here, $\operatorname{rank}(\mathcal{X})$ denotes the *multilinear rank* [15] of the tensor $\mathcal{X}$, a tuple of d integers defined via the ranks of the matricizations of $\mathcal{X}$ (see Section 2.1 for details) and $\mathrm{P}_\Omega : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a linear operator. A typical choice for P_Ω frequently encountered in applications is

$$\mathrm{P}_\Omega \mathcal{X} := \begin{cases} \mathcal{X}_{i_1 i_2 \ldots, i_d} & \text{if } \ (i_1, i_2, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise}, \end{cases}$$

[*] The work of M. Steinlechner has been supported by the SNSF research module *Riemannian optimization for solving high-dimensional problems with low-rank tensor techniques* within the SNSF ProDoc *Efficient Numerical Methods for Partial Differential Equations.*

[†] MATHICSE-ANCHP, École polytechnique fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. daniel.kressner@epfl.ch

[‡] MATHICSE-ANCHP, École polytechnique fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. michael.steinlechner@epfl.ch

[§] Department of Mathematics, Princeton University, Fine Hall, Princeton, NJ 08544, US. bartv@math.princeton.edu

where $\Omega \subset [1, n_1] \times \cdots \times [1, n_d]$ denotes the so-called sampling set. In this case, the objective function $\| \mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}\|^2/2$ measures the ability of $\mathcal{X}$ to match the entries of the partially known tensor $\mathcal{A}$ in Ω.

The tensor completion problem (1) and variants thereof have been discussed a number of times in the literature. Most of this work builds upon existing work for the special case $d = 2$, also known as matrix completion, see [18] for a comprehensive overview. One of the first approaches to tensor completion has been discussed by Liu et al. [16]. It is based on extending the notion of nuclear norm to tensors by defining $\|\mathcal{X}\|_*$ as the (weighted) sum of the nuclear norms of the matricizations of $\mathcal{X}$. This leads to the convex optimization problem

$$\min_{\mathcal{X}} \ \|\mathcal{X}\|_* \quad \text{subject to} \quad \mathrm{P}_\Omega \mathcal{X} = \mathrm{P}_\Omega \mathcal{A}, \tag{2}$$

which can be addressed by established techniques such as block coordinate descent. Allowing noise in the sampling data and thus somewhat closer to our formulation (1) of tensor completion, Signoretto et al. [25] and Gandy et al. [11] consider the unconstrained optimization problem

$$\min_{\mathcal{X}} \ \| \mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}\|^2 + \mu \|\mathcal{X}\|_*$$

and propose the use of ADMM (alternating direction method of multipliers) and other splitting methods. This approach has been shown to yield good recovery results when applied to tensors from various fields such as medical imaging, hyperspectral images and seismic data. However, nuclear norm minimization approaches are usually quite costly and involve singular value decompositions of potentially very large matrices. Liu/Shang [17] recently proposed the use of the economy sized QR decomposition, reducing the cost per iteration step considerably.

Besides the two approaches described above, a number of variations [20] and alternatives have been discussed in the literature. For example, [16] proposes a block coordinate descent method while [23] proposes an iterative hard thresholding method for fitting the factors of a Tucker decomposition. In [3], gradient optimization techniques have been proposed for fitting the factors of CP decomposition. Closely related to the approach considered in this paper, Da Silva and Herrmann [8] have recently proposed to perform tensor completion in the hierarchical Tucker format via Riemannian optimization.

The approach proposed in this paper is based on the observation that the set of tensors of fixed multilinear rank $\mathbf{r}$, denoted by $\mathcal{M}_\mathbf{r}$, forms a smooth manifold [28, 14]. Manifold structure for low-rank tensors has recently been exploited in a number of works targeting applications in numerical analysis and computational physics, see [12] for an overview. We will make use of this manifold structure by viewing (1) as an unconstrained optimization problem on $\mathcal{M}_\mathbf{r}$. This view allows for the use of Riemannian optimization techniques [1]. A similar approach has been considered in [30, 21, 19] for the matrix case, where it was shown to be competitive to other state-of-the-art approaches to matrix completion. Note that it is not entirely trivial to extend such Riemannian optimization techniques from the matrix to the tensor case, due to the lack of a simple characterization of the metric projection onto $\mathcal{M}_\mathbf{r}$ [15].

The rest of this paper is organized as follows. In Section 2, we recall differential geometric properties of tensors having fixed multilinear rank and propose a suitable retraction map for $\mathcal{M}_{\mathbf{r}}$. Section 3 proposes the use of the nonlinear CG algorithm on $\mathcal{M}_{\mathbf{r}}$ for solving (1), for which several algorithmic details as well as convergence properties are discussed. Finally, in Section 4, we investigate the effectiveness of our algorithm for various test cases, including synthetic data, hyperspectral images, and function-related tensors.

2 Differential geometry for low-rank tensor manifolds

To adapt the nonlinear CG algorithm on manifolds to the tensor completion problem (1), we derive a number of basic tools from differential geometry for tensors of low multilinear rank.

2.1 Preliminaries on tensors

Throughout this paper, we will follow the notation in the survey paper by Kolda and Bader [15]. In the following, we give a brief summary.

The *ith mode matricization*

$$X_{(i)} \in \mathbb{R}^{n_i \times \prod_{j \neq i} n_j}.$$

of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ is a rearrangement of the entries of $\mathcal{X}$ into the matrix $X_{(i)}$, such that the ith mode becomes the row index and all other $(d-1)$ modes become column indices, in lexicographical order. Similarly, the *vectorization* $\operatorname{vec}(\mathcal{X}) \in \mathbb{R}^{\prod_{i=1}^d n_i}$ stacks all entries of $\mathcal{X}$ into one long vector. The ranks of all the matricizations yield the *multilinear rank* tuple $\mathbf{r}$ of $\mathcal{X}$:

$$\operatorname{rank}(\mathcal{X}) = \big(\operatorname{rank}(X_{(1)}), \operatorname{rank}(X_{(2)}), \dots, \operatorname{rank}(X_{(d)})\big) .$$

The *ith mode product* of $\mathcal{X}$ multiplied with a matrix $M \in \mathbb{R}^{m \times n_i}$ is defined as

$$\mathcal{Y} = \mathcal{X} \times_i M \quad \Leftrightarrow \quad Y_{(i)} = M X_{(i)}, \quad \mathcal{Y} \in \mathbb{R}^{n_1 \times \dots \times n_{i-1} \times m \times n_{i+1} \times \dots \times n_d}.$$

The *inner product* of two tensors $\mathcal{X}$ and $\mathcal{Y}$ is given by

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \operatorname{vec}(\mathcal{X}), \operatorname{vec}(\mathcal{Y}) \rangle = \operatorname{vec}(\mathcal{X})^T \operatorname{vec}(\mathcal{Y}) = \operatorname{trace}(X_{(1)}^T Y_{(1)}) . \tag{3}$$

This induces the norm $\|\mathcal{X}\| := \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$.

Any tensor of multilinear rank $\mathbf{r} = (r_1, r_2, \dots, r_d)$ can be represented in the so-called *Tucker decomposition*

$$\mathcal{X} = \mathcal{C} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d = \mathcal{C} \bigtimes_{i=1}^{d} U_i, \tag{4}$$

with the *core tensor* $\mathcal{C} \in \mathbb{R}^{r_1 \times \dots \times r_d}$, and the *basis matrices* $U_i \in \mathbb{R}^{n_i \times r_i}$. Without loss of generality, all U_i are orthonormal: $U_i^T U_i = I_{r_i}$, which will be assumed for the rest of the paper.

Let us denote the truncation of a tensor $\mathcal{X}$ to multilinear rank $\mathbf{r}$ using the *higher order singular value decomposition* (HOSVD) [9] by $\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}}$. The HOSVD procedure can be described by the successive application of best rank-r_i approximations $\mathrm{P}_{r_i}^{i}$ in each mode $i = 1, \dots, d$:

$$\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathcal{M}_{\mathbf{r}}, \quad \mathcal{X} \mapsto \mathrm{P}_{r_d}^{d} \circ \cdots \circ \mathrm{P}_{r_1}^{1} \mathcal{X}.$$

Each individual projection can be computed by a truncated SVD as follows. Let U_Y contain the r_i dominant left singular vectors of the ith matricization $Y_{(i)}$ of a given tensor $\mathcal{Y}$. Then the tensor resulting from the projection $\widetilde{\mathcal{Y}} = \mathrm{P}_{r_i}^{i} \mathcal{Y}$ is given in terms of its matricization as $\widetilde{Y}_{(i)} = U_Y U_Y^T Y_{(i)}$.

In contrast to the matrix case, the HOSVD does in general *not* yield the best rank-$\mathbf{r}$ approximation. Instead, the following quasi-best approximation property [9] holds:

$$\|\mathcal{X} - \mathrm{P}_{\mathbf{r}}^{\mathrm{HO}} \mathcal{X}\| \leq \sqrt{d}\, \|\mathcal{X} - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}} \mathcal{X}\|, \tag{5}$$

where $\mathrm{P}_{\mathcal{M}_{\mathbf{r}}} \mathcal{X} \in \mathcal{M}_{\mathbf{r}}$ is any best approximation of $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in the norm $\| \cdot \|$.

The HOSVD does inherit the smoothness of low-rank matrix approximations.

Proposition 1 (Smoothness of truncated HOSVD)**.** *Let $\mathcal{X} \in \mathcal{M}_{\mathbf{r}}$. Then there exists a neighborhood $D \subset \mathbb{R}^{n_1 \times \cdots \times n_d}$ of $\mathcal{X}$ such that $\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}} : D \to \mathcal{M}_{\mathbf{r}}$ is C^∞ smooth.*

Proof. Let D_i denote the open set of tensors whose ith mode matricization has a nonzero gap between the r_ith and the $(r_i + 1)$th singular values. From standard results in matrix perturbation theory, it then follows [7] that each projector $\mathrm{P}_{r_i}^{i}$ is smooth and well-defined on D_i. Since $\mathcal{X} \in \mathcal{M}_{\mathbf{r}}$ is contained in all D_i and is a fixpoint of every $\mathrm{P}_{r_i}^{i}$, it is possible to construct an open neighborhood $D \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of $\mathcal{X}$ such that $\mathrm{P}_{r_i}^{i} \circ \cdots \circ \mathrm{P}_{r_1}^{1} D \subseteq D_i$ for all i. Hence, the chain rule yields the smoothness of the operator $\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}}$ on D. □

2.2 Manifold setting

The set $\mathcal{M}_{\mathbf{r}}$ of tensors of fixed multilinear rank $\mathbf{r} = (r_1, r_2, \dots, r_d)$ forms a smooth embedded submanifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ [28, 29]. By counting the degrees of freedom in (4), it follows that the dimension of $\mathcal{M}_{\mathbf{r}}$ is given by

$$\dim(\mathcal{M}_{\mathbf{r}}) = \prod_{j=1}^{d} r_j + \sum_{i=1}^{d} r_i n_i - r_i^2.$$

Observe that $\dim(\mathcal{M}_{\mathbf{r}})$ is much smaller than the dimension of $R^{n_1 \times \cdots \times n_d}$ when $r_i \ll n_i$. The Tucker decomposition (4) allows for the efficient representation and manipulation of tensors in $\mathcal{M}_{\mathbf{r}}$.

According to [14], the *tangent space* of $\mathcal{M}_{\mathbf{r}}$ at $\mathcal{X} = \mathcal{C} \times_1 U_1 \cdots \times_d U_d$ can be parametrized as

$$T_{\mathcal{X}} \mathcal{M}_{\mathbf{r}} = \left\{ \mathcal{G} \bigtimes_{i=1}^{d} U_i + \sum_{i=1}^{d} \mathcal{C} \times_i V_i \bigtimes_{j \neq i} U_j \;\middle|\; V_i^T U_i = 0 \right\}, \tag{6}$$

where $\mathcal{G} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ and $V_i \in \mathbb{R}^{n_i \times r_i}$ are the free parameters. Furthermore, the orthogonal projection of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ onto $T_\mathcal{X}\mathcal{M}_\mathbf{r}$ is given by

$$\mathrm{P}_{T_\mathcal{X}\mathcal{M}_\mathbf{r}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to T_\mathcal{X}\mathcal{M}_\mathbf{r},$$

$$\mathcal{A} \mapsto \left(\mathcal{A} \bigtimes_{j=1}^{d} U_j^T\right) \bigtimes_{i=1}^{d} U_i + \sum_{i=1}^{d} \mathcal{C} \times_i \left(\mathrm{P}_{U_i}^{\perp}\left[\mathcal{A} \bigtimes_{j\neq i} U_j^T\right]_{(i)} C_{(i)}^{\dagger}\right) \bigtimes_{k\neq i} U_k. \tag{7}$$

Here, $C_{(j)}^{\dagger}$ denotes the pseudo-inverse of $C_{(j)}$. Note that $C_{(j)}$ has full row rank and hence $C_{(j)}^{\dagger} = C_{(j)}^T (C_{(j)} C_{(j)}^T)^{-1}$. We use $\mathrm{P}_{U_i}^{\perp} := \underline{I_{r_i}} - U_i U_i^T$ to denote the orthogonal projection onto the orthogonal complement of $\mathrm{span}(U_i)$. little error here should be I_{d_i}

2.3 Riemannian metric and gradient

As a metric on $\mathcal{M}_\mathbf{r}$, we will use the Euclidean metric from the embedded space induced by the inner product (3). Together with this metric, $\mathcal{M}_\mathbf{r}$ becomes a Riemannian manifold. This in turn allows us to define the Riemannian gradient of an objective function, which can be obtained from the projection of the Euclidean gradient into the tangent space.

Proposition 2 ([1, Chap. 3.6])**.** *Let $f : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}$ be a cost function with Euclidean gradient $\nabla f_\mathcal{X}$ at point $\mathcal{X} \in \mathcal{M}_\mathbf{r}$. Then the Riemannian gradient of $f : \mathcal{M}_\mathbf{r} \to \mathbb{R}$ is given by $\mathrm{grad}\, f(\mathcal{X}) = \mathrm{P}_{T_\mathcal{X}\mathcal{M}_\mathbf{r}}(\nabla f_\mathcal{X})$.*

By Proposition 2, the Riemannian gradient of the objective function $f(\mathcal{X}) = \|\mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}\|^2/2$ is given by

$$\mathrm{grad}\, f(\mathcal{X}) = \mathrm{P}_{T_\mathcal{X}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}). \tag{8}$$

2.4 Retraction

Retraction maps an element from the tangent space at $\mathcal{X} \in \mathcal{M}_\mathbf{r}$ to the manifold $\mathcal{M}_\mathbf{r}$. The choice of this map is not unique. A popular theoretical choice is the so-called *exponential map*, which, however, is usually too expensive to compute. According to [1] it is often sufficient to approximate this exponential map in first order for the purpose of optimization algorithms. A graphical depiction of this concept is shown on the left of Figure 1. More specifically, a retraction fulfills the following properties.

Definition 1 (Retraction, [2, Def. 1])**.** *Let $\mathcal{M}$ be a smooth submanifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$. Let 0_x denote the zero element of $T_x\mathcal{M}$. A mapping R from the tangent bundle $T\mathcal{M}$ into $\mathcal{M}$ is said to be a* retraction *on $\mathcal{M}$* around *$x \in \mathcal{M}$ if there exists a neighborhood $\mathcal{U}$ of $(x, 0_x)$ in $T\mathcal{M}$ such that the following properties hold:*

(a) We have $\mathcal{U} \subseteq \mathrm{dom}(R)$ and the restriction $R : \mathcal{U} \to \mathcal{M}$ is smooth.

(b) $R(y, 0_y) = y$ for all $(y, 0_y) \in \mathcal{U}$.

(c) With the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$, R satisfies the local rigidity condition*:*

$$DR(x, \cdot)(0_x) = \mathrm{id}_{T_x\mathcal{M}} \text{ for all } (x, 0_x) \in \mathcal{U},$$

where $\mathrm{id}_{T_x\mathcal{M}}$ *denotes the identity mapping on* $T_x\mathcal{M}$.

If $\mathcal{M}$ is an embedded submanifold then the orthogonal projection

$$\mathrm{P}_{\mathcal{M}}(x+\xi) = \operatorname*{argmin}_{y\in\mathcal{M}} \|x+\xi-y\| \tag{9}$$

induces the so called *projective retraction*

$$R : \mathcal{U} \to \mathcal{M}, \quad (x,\xi) \mapsto \mathrm{P}_{\mathcal{M}}(x+\xi),$$

which satisfies the properties of Definition 1 [2, Prop. 5].

Since it only satisfies the quasi-best approximation property (5), the HOSVD procedure does *not* yield a projective retraction. Nevertheless, it still possesses all necessary properties of a retraction in the sense of Definition 1.

Proposition 3 (HOSVD as Retraction)**.** *The map*

$$R : T\mathcal{M}_{\mathbf{r}} \to \mathcal{M}_{\mathbf{r}}, \quad (\mathcal{X},\xi) \mapsto \mathrm{P}^{\mathrm{HO}}_{\mathbf{r}}(\mathcal{X}+\xi) \tag{10}$$

is a retraction on $\mathcal{M}_{\mathbf{r}}$ *around* $\mathcal{X}$.

Proof. The map R defined in (10) can be written as the composition

$$R : T\mathcal{M}_{\mathbf{r}} \to \mathcal{M}_{\mathbf{r}}, \quad (\mathcal{X},\xi) \mapsto \mathrm{P}^{\mathrm{HO}}_{\mathbf{r}} \circ F(\mathcal{X},\xi),$$

where the smooth map $F : T\mathcal{M}_{\mathbf{r}} \to \mathbb{R}^{n_1\times\cdots\times n_d}$ is defined as $F(\mathcal{X},\xi) := \mathcal{X}+\xi$. By Proposition 1, $\mathrm{P}^{\mathrm{HO}}_{\mathbf{r}}$ is smooth for all $\mathcal{X}\in\mathcal{M}_{\mathbf{r}}$ and sufficiently small ξ. Hence, R defines a locally smooth map in a neighborhood $\mathcal{U}\subset T\mathcal{M}_{\mathbf{r}}$ around $(\mathcal{X}, 0_{\mathcal{X}})$.

Definition 1 (b) follows from the fact that the application of the HOSVD to elements in $\mathcal{M}_{\mathbf{r}}$ leaves them unchanged.

It remains to check Definition 1 (c), the local rigidity condition. Because the tangent space $T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}$ is a first order approximation of $\mathcal{M}_{\mathbf{r}}$ around ξ, we have that $\|(\mathcal{X}+t\xi) - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathcal{X}+t\xi)\| = O(t^2)$ for $t\to 0$. Thus, using (5):

$$\|(\mathcal{X}+t\xi) - R(\mathcal{X},t\xi)\| \le \sqrt{d}\|(X+t\xi) - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathcal{X}+t\xi)\| = O(t^2).$$

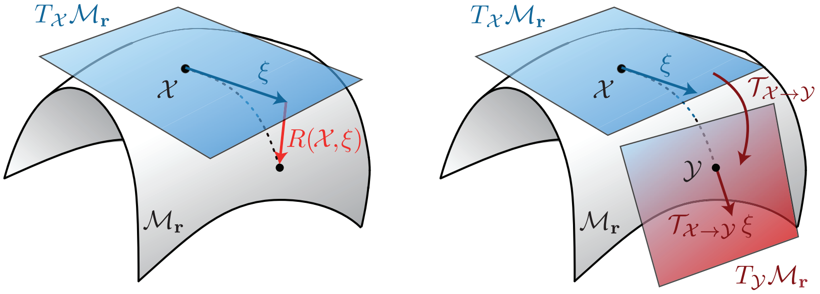
Hence, $R(\mathcal{X},t\xi) = (\mathcal{X}+t\xi) + O(t^2)$, which gives $\frac{d}{dt}R(\mathcal{X},t\xi)\Big|_{t=0} = \xi$. In other words, $DR(\mathcal{X},\cdot)(0_{\mathcal{X}}) = \mathrm{id}_{T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}}$, which completes the proof. □

2.5 Vector transport

A vector transport $\mathcal{T}_{\mathcal{X}\to\mathcal{Y}}$, as introduced in [1], allows us to map tangent vectors from $T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}$ to $T_{\mathcal{Y}}\mathcal{M}_{\mathbf{r}}$. Because $\mathcal{M}_{\mathbf{r}}$ is an embedded submanifold of $\mathbb{R}^{n_1\times\cdots\times n_d}$, the orthogonal projection $\mathrm{P}_{T_{\mathcal{Y}}\mathcal{M}_{\mathbf{r}}}$ constitutes a vector transport, see [1, Sec. 8.1.3]:

$$\mathcal{T}_{\mathcal{X}\to\mathcal{Y}} : T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}} \to T_{\mathcal{Y}}\mathcal{M}_{\mathbf{r}}, \quad \xi \mapsto \mathrm{P}_{T_{\mathcal{Y}}\mathcal{M}_{\mathbf{r}}}(\xi).$$

A visualization of the concept of vector transport is shown on the right of Figure 1.


$T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}$
$\mathcal{X}$
ξ
$R(\mathcal{X},\xi)$
$\mathcal{M}_{\mathbf{r}}$
$T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}$
$\mathcal{X}$
ξ
$\mathcal{T}_{\mathcal{X}\to\mathcal{Y}}$
$\mathcal{Y}$
$\mathcal{M}_{\mathbf{r}}$
$\mathcal{T}_{\mathcal{X}\to\mathcal{Y}}\,\xi$
$T_{\mathcal{Y}}\mathcal{M}_{\mathbf{r}}$

(a) Retraction

(b) Vector transport

Figure 1: *Graphical representation of the concept of retraction and vector transport within the framework of Riemannian optimization techniques.*

3 Nonlinear Riemannian CG

With the concepts introduced in Section 2, we have all the necessary geometric ingredients for performing Riemannian optimization on the manifold $\mathcal{M}_{\mathbf{r}}$ of low-rank tensors. In particular, the nonlinear CG algorithm discussed in [1, Sec 8.3], yields Algorithm 1. This can be seen as an extension of the standard nonlinear CG algorithm [22], with the Euclidean gradient replaced by the Riemannian gradient. Applying retraction after each optimization step ensures that we stay on the manifold. Finally, the use of vector transport allows us to calculate conjugate directions using the Polak-Ribière+ (PR+) update rule. If the search directions become insufficiently gradient-related during the iteration, the algorithm should revert to steepest descent, see [5]. A standard Armijo backtracking scheme is added to control the step sizes, using the result of a linearized line search procedure as an initial guess.

Algorithm 1 Geometric nonlinear CG for Tensor Completion

```
Input: Initial guess $\mathcal{X}_0 \in \mathcal{M}_{\mathbf{r}}$.
  $\eta_0 \leftarrow -\operatorname{grad} f(\mathcal{X}_0)$                      % first step is steepest descent
  $\alpha_0 \leftarrow \operatorname{argmin}_\alpha f(\mathcal{X}_0 + \alpha\eta_0)$               % step size by linearized line search
  $\mathcal{X}_1 \leftarrow R(\mathcal{X}_0, \alpha_0\eta_0)$

  for $k = 1, 2, \ldots$ do
    $\xi_k \leftarrow \operatorname{grad} f(\mathcal{X}_k)$                     % compute Riemannian gradient
    $\eta_k \leftarrow -\xi_k + \beta_k \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}\eta_{k-1}$         % conjugate direction by updating rule
    $\alpha_k \leftarrow \operatorname{argmin}_\alpha f(\mathcal{X}_k + \alpha\eta_k)$             % step size by linearized line search
    Find smallest integer $m \geq 0$ such that  % Armijo backtracking for sufficient decrease
        $f(\mathcal{X}_k) - f(R(\mathcal{X}_k, 2^{-m}\alpha_k\eta_k)) \geq -10^{-4} \cdot \langle \xi_k, 2^{-m}\alpha_k\eta_k \rangle$
    $\mathcal{X}_{k+1} \leftarrow R(\mathcal{X}_k, 2^{-m}\alpha_k\eta_k)$               % obtain next iterate by retraction
  end for
```

In the following sections, we will provide algorithmic details on the individual steps of Algorithm 1 and discuss their computational complexity. To simplify the expressions for the complexity, we assume that $n := n_1 = \ldots = n_d$ and $r := r_1 = \ldots = r_d$.

3.1 Calculation of the gradient

The calculation of the Riemannian gradient (8) requires the explicit computation of individual entries of a tensor $\mathcal{X}$ from its Tucker decomposition:

$$\mathcal{X}_{i_1 i_2 \dots i_d} = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \cdots \sum_{j_d=1}^{r_d} \mathcal{C}_{j_1 j_2 \dots j_d} (U_1)_{i_1 j_1} (U_2)_{i_2 j_2} \cdots (U_d)_{i_d j_d}.$$

In total, this requires $|\Omega|(d+1)r^d$ operations for computing $\mathrm{P}_\Omega \mathcal{X}$.

The projection of $\mathcal{E} := \mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A}$ onto the tangent space gives the gradient ξ, which will be stored in factorized form as

$$\xi = \mathcal{G} \bigtimes_{j=1}^{d} U_j + \sum_{i=1}^{d} \mathcal{C} \times_i V_i \bigtimes_{j \neq i} U_j \,, \tag{11}$$

where

$$\mathcal{G} := \mathcal{E} \bigtimes_{j=1}^{d} U_j^T, \qquad V_i := \mathrm{P}_{U_i}^{\perp} \left[\mathcal{E} \bigtimes_{j \neq i} U_j^T \right]_{(i)} C_{(i)}^{\dagger},$$

see (7). By exploiting the sparsity of $\mathcal{E}$, the computation of $\mathcal{G}$ and V_i, $i = 1, \dots, d$, requires $O\big(r^d(|\Omega| + n) + r^{d+1}\big)$ operations. This makes the calculation of the gradient the most time consuming part of our optimization scheme.

3.2 Vector transport and new search direction

To calculate the new search direction, we use the Polak-Ribière+ update formula adapted to Riemannian optimization, see [1, 22]:

$$\beta_k = \max \left\{ 0, \frac{\langle \operatorname{grad} f(\mathcal{X}_k), \operatorname{grad} f(\mathcal{X}_k) - \mathcal{T}_{\mathcal{X}_{k-1} \to \mathcal{X}_k} \operatorname{grad} f(\mathcal{X}_{k-1}) \rangle}{\| \operatorname{grad} f(\mathcal{X}_{k-1}) \|^2} \right\}. \tag{12}$$

The calculation of β_k requires the evaluation of the vector transport

$$\mathcal{T}_{\mathcal{X}_{k-1} \to \mathcal{X}_k} \xi = \mathrm{P}_{T_{\mathcal{X}_k} \mathcal{M}_{\mathbf{r}}} (\xi_{k-1}),$$

where $\xi_{k-1} = \operatorname{grad} f(\mathcal{X}_{k-1})$ is assumed to be in the factorized form (11). Moreover, $\mathcal{X}_k \in \mathcal{M}_{\mathbf{r}}$ is given in terms of a Tucker decomposition $\mathcal{X}_k = \widetilde{\mathcal{C}} \bigtimes_{i=1}^{d} \widetilde{U}_i$. As in the previous section, we obtain

$$\mathrm{P}_{T_{\mathcal{X}_k} \mathcal{M}_{\mathbf{r}}} (\xi_{k-1}) = \widetilde{\mathcal{G}} \bigtimes_{j=1}^{d} \widetilde{U}_j + \sum_{i=1}^{d} \widetilde{\mathcal{C}} \times_i \widetilde{V}_i \bigtimes_{j \neq i} \widetilde{U}_j \,, \tag{13}$$

where

$$\widetilde{\mathcal{G}} := \xi_{k-1} \bigtimes_{j=1}^{d} \widetilde{U}_j^T, \qquad \widetilde{V}_i := \mathrm{P}_{\widetilde{U}_i}^{\perp} \left[\xi_{k-1} \bigtimes_{j \neq i} \widetilde{U}_j^T \right]_{(i)} \widetilde{C}_{(i)}^{\dagger}.$$

To compute and $\widetilde{\mathcal{G}}$ and $\widetilde{V}_i$, we make use of the linearity in ξ_{k-1} and process each summand in the representation (11) of ξ_{k-1} separately. By exploiting the tensor product structure of each summand, we then arrive at a total cost of $O(nr^d)$ operations.

Further, the evaluation of (12) requires the inner product between the tensor $\mathrm{P}_{T_{\mathcal{X}_i}\mathcal{M}_{\mathbf{r}}}(\xi_{k-1})$ in (13) and $\xi_k = \operatorname{grad} f(\mathcal{X}_i)$ also given in factorized form:

$$\xi_k = \widehat{\mathcal{G}} \bigtimes_{j=1}^{d} \widetilde{U}_j + \sum_{i=1}^{d} \widetilde{\mathcal{C}} \times_i \widehat{V}_i \bigtimes_{j \neq i} \widetilde{U}_j .$$

Utilizing the orthogonality of $\widetilde{U}_i$ and the uniqueness condition $\widetilde{U}_i^T \widetilde{V}_i = \widetilde{U}_i^T \widehat{V}_i = 0$ for the tangent space, see (6), we obtain

$$\big\langle\, \xi_k \,,\, \mathrm{P}_{T_{\mathcal{X}_k}\mathcal{M}_{\mathbf{r}}}(\xi_{k-1}) \,\big\rangle = \langle\, \widehat{\mathcal{G}} \,,\, \widetilde{\mathcal{G}} \,\rangle + \sum_{i=1}^{d} \langle\, \widetilde{\mathcal{C}} \,,\, \widetilde{\mathcal{C}} \times_i \widehat{V}_i^T \widetilde{V}_i \,\rangle .$$

The evaluation of the (smaller) inner products requires $O(nr^2 + r^{d+1})$ operations. The norm of ξ_{k-1} appearing in the denominator of (12) is computed analogously. Hence, the total cost for computing β_k is given by $O(nr^d)$ operations.

Once β_k has been determined, the new conjugate direction is computed by

$$\eta_k = -\xi_k + \beta_k \mathcal{T}_{\mathcal{X}_{k-1} \to \mathcal{X}_k} \eta_{k-1} .$$

where $\eta_{k-1} \in T_{\mathcal{X}_{k-1}}\mathcal{M}_{\mathbf{r}}$ is the previous conjugate direction. The vector transport is performed exactly in the same way as above. The obtained tensor in $T_{\mathcal{X}_k}\mathcal{M}_{\mathbf{r}}$ is multiplied by β_k and added to $-\xi_k \in T_{\mathcal{X}_k}\mathcal{M}_{\mathbf{r}}$. Due to linearity, the addition of two tensors in the same tangent space is performed by simply adding the corresponding coefficients $\mathcal{G}$ and V_i.

3.3 Calculation of the retraction

To obtain the next iterate, Algorithm 1 retracts the updated tensor $\mathcal{X} + \alpha\eta$ back to the manifold by means of the HOSVD. When performing this retraction, we will exploit the fact that $\mathcal{X} \in \mathcal{M}_{\mathbf{r}}$ is in Tucker decomposition and $\eta \in T_{\mathcal{X}}\mathcal{M}_{\mathbf{r}}$ is represented in the factorized form (11):

$$\begin{aligned}
\mathcal{X} + \alpha\eta &= \mathcal{C} \bigtimes_{i=1}^{d} U_i + \alpha \left(\mathcal{G} \bigtimes_{i=1}^{d} U_i + \sum_{i=1}^{d} \mathcal{C} \times_i V_i \bigtimes_{j \neq i} U_j \right) \\
&= (\mathcal{C} + \alpha\mathcal{G}) \bigtimes_{i=1}^{d} U_i + \alpha \sum_{i=1}^{d} \mathcal{C} \times_i V_i \bigtimes_{j \neq i} U_j \\
&= \mathcal{S} \bigtimes_{i=1}^{d} [U_i, V_i]
\end{aligned}$$

where $\mathcal{S} \in \mathbb{R}^{2r_1 \times \cdots \times 2r_d}$ has the special structure depicted in Figure 2. After orthogonalizing the combined basis matrices $[U_i, V_i]$ and a corresponding update of $\mathcal{S}$, we can then restrict the application of the HOSVD to the smaller tensor $\mathcal{S}$, which requires only $O\big(r^{d+1}\big)$ operations. The retraction is completed by multiplying the basis matrices obtained from the HOSVD of $\mathcal{S}$ to the combined basis factors. In total, the retraction requires $O(nr^2 + r^{d+1})$ operations.

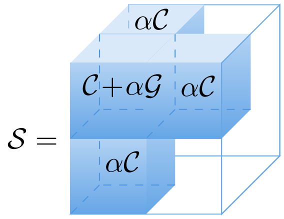


Figure 2: *Structure of the combined core tensor $\mathcal{S}$ for the case $d = 3$.*

3.4 Line Search

Following [30], we obtain an initial guess for the step size α in Algorithm 1 by performing exact line search along the tangent space. This leads to the optimization problem

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \| \mathrm{P}_\Omega(\mathcal{X} + \alpha\xi) - \mathrm{P}_\Omega\, \mathcal{A}\|^2,$$

which can be solved analytically:

$$\alpha^* = \frac{\langle \mathrm{P}_\Omega\, \xi, \mathrm{P}_\Omega(\mathcal{A} - \mathcal{X})\rangle}{\langle \mathrm{P}_\Omega\, \xi, \mathrm{P}_\Omega\, \xi\rangle}. \tag{14}$$

The computation of the nonzero entries of $\mathrm{P}_\Omega\, \xi$ requires $2|\Omega|(d+1)r^d$ operations (see Section 3.1), while each of the inner products requires $2|\Omega|$ operations.

This linearized exact line search procedure is combined with an Armijo backtracking scheme. In our numerical experiments, we have observed that this was almost never necessary; α^* is often very close to the optimal step size.

3.5 Summary of computational complexity

By exploiting the low-rank structures of the iterates, we arrive at a total cost of

$$O\big(r^d(n + |\Omega|) + r^{d+1}\big) \tag{15}$$

operations. In particular, Algorithm 1 scales linearly with n, when keeping r fixed. This makes the algorithm suitable for large sizes n and moderate values of d, say $d = 3, 4$.

3.6 Convergence of Algorithm 1

To investigate the convergence of our proposed algorithm, we proceed similarly to the matrix case [30, Sec. 4.1] by applying the general convergence theory for Riemannian optimization.

In particular [1, Theorem 4.3.1] yields the following proposition, which shows that any limit point of Algorithm 1 coincides with the prescribed entries $\mathrm{P}_\Omega\,\mathcal{A}$ *within the tangent space.*

Proposition 4. *Let $\mathcal{X}_k$ be an infinite sequence of iterates generated by Algorithm 1. Then, every accumulation point $\mathcal{X}_*$ of $\mathcal{X}_k$ satisfies $\mathrm{P}_{T_{\mathcal{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega\,\mathcal{X}_*) = \mathrm{P}_{T_{\mathcal{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega\,\mathcal{A})$.*

A more detailed convergence analysis is complicated by the fact that $\mathcal{M}_\mathbf{r}$ is not closed; for example a sequence of tensors in $\mathcal{M}_\mathbf{r}$ may approach a tensor for which the ith matricization has rank less than r_i. To avoid this effect, we first discuss the convergence for a modification of the original cost function $f(\mathcal{X}) = \|\mathrm{P}_\Omega\mathcal{X} - \mathrm{P}_\Omega\mathcal{A}\|^2/2$:

$$g : \mathcal{M}_\mathbf{r} \to \mathbb{R}, \quad \mathcal{X} \mapsto f(\mathcal{X}) + \mu^2 \sum_{i=1}^{d} \left(\|X_{(i)}\|^2 + \|X_{(i)}^\dagger\|^2 \right), \quad \mu > 0, \tag{16}$$

where $\|\cdot\|$ denotes the Frobenius norm for matrices.

Proposition 5. *Let $\mathcal{X}_k$ be an infinite sequence of iterates generated by Algorithm 1 but with the modified cost function g defined in* (16)*. Then*

$$\lim_{k\to\infty} \|\operatorname{grad} g(\mathcal{X}_k)\| = 0.$$

Proof. By construction of the line search, all iterates $\mathcal{X}_k$ fulfill $g(\mathcal{X}_k) \leq g(\mathcal{X}_0)$ and therefore

$$\frac{1}{2}\|\,\mathrm{P}_\Omega\,\mathcal{X}_k - \mathrm{P}_\Omega\,\mathcal{A}\|^2 + \mu^2 \sum_{i=1}^{d} \left(\|X_{k,(i)}\|^2 + \|X_{k,(i)}^\dagger\|^2 \right) \leq g(\mathcal{X}_0) =: C_0^2,$$

where $X_{k,(i)}$ denotes the ith matricization of $\mathcal{X}_k$. In particular,

$$\mu^2 \sum_{i=1}^{d} \|X_{k,(i)}\|^2 \leq C_0^2, \qquad \mu^2 \sum_{i=1}^{d} \|X_{k,(i)}^\dagger\|^2 \leq C_0^2,$$

yielding upper and lower bounds for the largest and smallest singular values, respectively:

$$\sigma_{\max}(X_{k,(i)}) \leq \|X_{k,(i)}\| \leq C_0/\mu, \qquad \sigma_{\min}^{-1}(X_{k,(i)}) \leq \|X_{k,(i)}^\dagger\| \leq C_0/\mu.$$

Hence, all iterates $\mathcal{X}_k$ stay inside the compact set

$$B := \left\{ \mathcal{X} \in \mathcal{M}_\mathbf{r} \;\middle|\; \sigma_{\max}(X_{(i)}) \leq C_0/\mu,\ \sigma_{\min}(X_{(i)}) \geq \mu/C_0 \text{ for } i = 1,\ldots,d \right\}.$$

Now suppose, conversely to the statement of the proposition, that $\|\operatorname{grad} g(\mathcal{X}_k)\|$ does not converge to zero. Then there is $\delta > 0$ and a subsequence of $\{\mathcal{X}_k\}$ such that $\|\operatorname{grad} g(\mathcal{X}_k)\| > \delta$ for all elements of the subsequence. Since $\mathcal{X}_k \in B$, it follows that this subsequence has an accumulation point $\mathcal{X}_*$ for which also $\|\operatorname{grad} g(\mathcal{X}_*)\| > \delta$. However, this contradicts [1, Theorem 4.3.1], which states that every accumulation point is a critical point of g. □

It is instructive to compare the gradient of g with the gradient of f at $\mathcal{X} \in \mathcal{M}_\mathbf{r}$. For this purpose, we use the fact that $X_{(i)}$ has full row-rank and thus the derivative of its pseudo-inverse $X_{(i)}^\dagger = X_{(i)}^T(X_{(i)}X_{(i)}^T)^{-1}$ can be written as

$$\partial X_{(i)}^\dagger = -X_{(i)}^\dagger(\partial X_{(i)})X_{(i)}^\dagger + (I - X_{(i)}^\dagger X_{(i)})(\partial X_{(i)})^T \left(X_{(i)}X_{(i)}^T \right)^{-1}.$$

Thus, the Euclidean gradient of g at $\mathcal{X}$ is given by

$$\nabla g(\mathcal{X}) = \nabla f(\mathcal{X}) + 2\mu^2 \sum_{i=1}^{d} \left[U_i \left(\Sigma_i - (\Sigma_i^\dagger)^3 \right) V_i^T \right]^{(i)}, \tag{17}$$

in terms of the singular value decomposition $X_{(i)} = U_i \Sigma_i V_i^T$. The operation $[\,\cdot\,]^{(i)}$ reverses matricization, that is, $[X_{(i)}]^{(i)} = \mathcal{X}$.

The statement of Proposition 5 holds for arbitrarily small μ. If the smallest singular values of the matricizations stay bounded from below as $\mu \to 0$, that is, the accumulation points $\mathcal{X}_*$ of $\{\mathcal{X}_k\}$ do not approach the boundary of $\mathcal{M}_{\mathbf{r}}$ as $\mu \to 0$, then (17) shows that $\operatorname{grad} f(\mathcal{X}_*) \to 0$ as $\mu \to 0$. Thus, the regularization term becomes negligible in such a situation. For more details, we refer to the discussion in [30, Sec. 4.1].

4 Numerical Experiments

Algorithm 1 (*geomCG*) was implemented in Matlab version 2012a, using the Tensor Toolbox version 2.5 [4] for handling some of the tensor operations. However, to attain reasonable performance, it was important to implement operations with sparse tensors in C and call them via `mex` interfaces. In particular, this was done for the evaluation of the objective function (1), the computation of the Euclidean gradient and its projection onto the tangent space (11), as well as for the linearized line search (14). For simplicity, we restricted the implementation to the case $d = 3$. The source code is freely available under a BSD license and can be downloaded from `http://anchp.epfl.ch`.

To measure the convergence during the iteration, Algorithm 1 computes the relative residual

$$\frac{\| \mathrm{P}_\Omega \mathcal{X} - \mathrm{P}_\Omega \mathcal{A} \|}{\| \mathrm{P}_\Omega \mathcal{A} \|}.$$

However, to investigate the reconstruction quality of the algorithm, measuring the relative residual on the *sampling set* Ω is not sufficient. For this purpose, we also measure the relative error $\| \mathrm{P}_\Gamma \mathcal{X} - \mathrm{P}_\Gamma \mathcal{A} \| / \| \mathrm{P}_\Gamma \mathcal{A} \|$ on a random *test set* Γ of the same cardinality as Ω.

Unless stated otherwise, we assume that the tensor has equal size in all modes, $n := n_1 = n_2 = n_3$ and similarly for the ranks, $r := r_1 = r_2 = r_3$. All tests were performed on a quad-core Intel Xeon E31225, 3.10GHz, with 8GB of RAM running 64-Bit Debian 7.0 Linux. Stated calculation times are wall-clock times, excluding the set-up time of the problem.

4.1 Computational Complexity for Synthetic Data Sets

A synthetic data tensor $\mathcal{A}$ of exact multilinear rank $\mathbf{r}$ is created by choosing the entries of the core tensor $\mathcal{C}$ and the basis matrices U_1, U_2, U_3 as pseudo-random numbers from a uniform distribution on $[0, 1]$.

As a first test, we check that the implementation of Algorithm 1 exhibits the same scaling behaviour per iteration as predicted by the theoretical discussion in Subsection 3.5. To measure the scaling with regard to the tensor size n, we fix the multilinear rank to $\mathbf{r} = (10, 10, 10)$ and scale the size of the sampling set linearly with the tensor size, $|\Omega| = 10n$. We perform 10 iterations of our algorithm and repeat the process 10 times for different randomly chosen datasets. Analogously, we measure the dependence on the tensor rank by setting the tensor size to $n = 300$ and fixing the sampling set to 0.1% of the full tensor.

The results are shown in Figure 3. We observe that our algorithm scales indeed linearly in the tensor size over a large interval $n \in [100, 3000]$. Even for such large tensors, the time per iteration step is very low. Plotting the results for the scaling with regard to the tensor rank, we observe an $O(r^3)$-dependence, in agreement with (15).

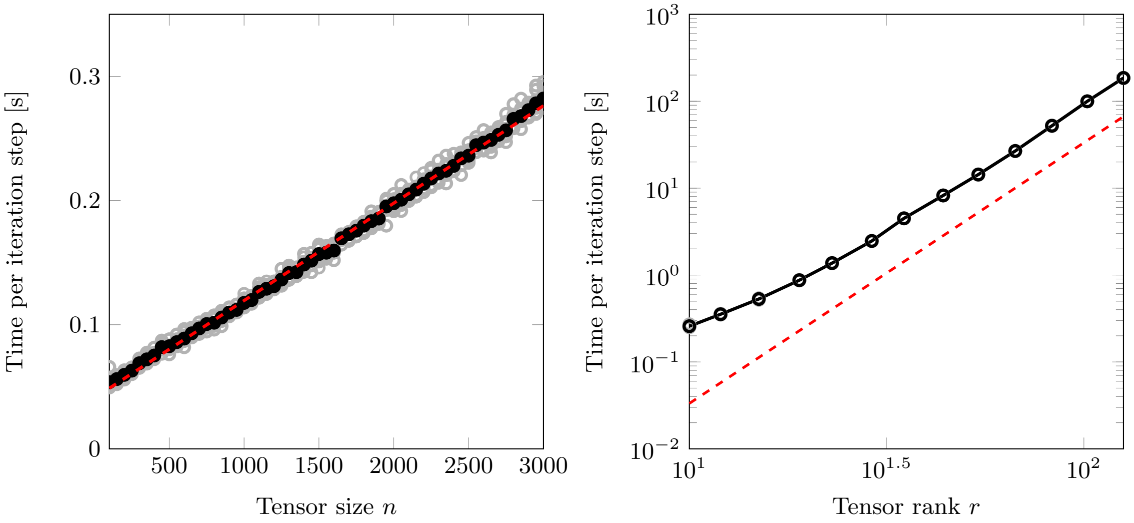


Figure 3: *Time needed per iteration step for various problem sizes.* ***Left:*** *Runtime with fixed rank* $\mathbf{r} = (10, 10, 10)$ *and varying tensor size* $n = n_1 = n_2 = n_3 \in \{100, 150, \ldots, 3000\}$. *The size of the sampling set scales linearly with* n, $|\Omega| = 10n$. ***Right:*** *Runtime with fixed tensor size* $n = (300, 300, 300)$ *and varying tensor rank* $r = r_1 = r_2 = r_3 \in \{20, 25, \ldots, 100\}$. *Size of sampling set: 0.1% of the full tensor. The red dashed line shows a scaling behaviour* $O(r^3)$.

4.2 Reconstruction of Synthetic Data Sets

We compare the reconstruction performance of our algorithm with the *hard completion* algorithm by Signoretto et al. [26, Alg. 3], based on the so called *inexact splitting method* applied to (2). We selected this algorithm because it is publicly available on the authors' website. The algorithm depends on certain parameters discussed in [26]. We have chosen the proximity parameter $\tau = 10$ and the nuclear norm weights $\lambda_1 = \lambda_2 = \lambda_3 = 1$, which corresponds to the settings used in the supplied test routine.

In Figure 4 we present the convergence behaviour of the algorithms for varying sizes of the sampling set, in terms of the error on the test set Γ. The sampling set sizes are denoted by a percentage p of the full tensor, $|\Omega| = pN^3$. We use a relative residual of 10^{-12} and a maximum number of 300 iterations as stopping criterions. Both algorithm need more iterations if the number of missing entries increases, but the effect is more strongly pronounced for hard completion. Our algorithm performs better both when measuring the performance with respect to time or the number of iterations.

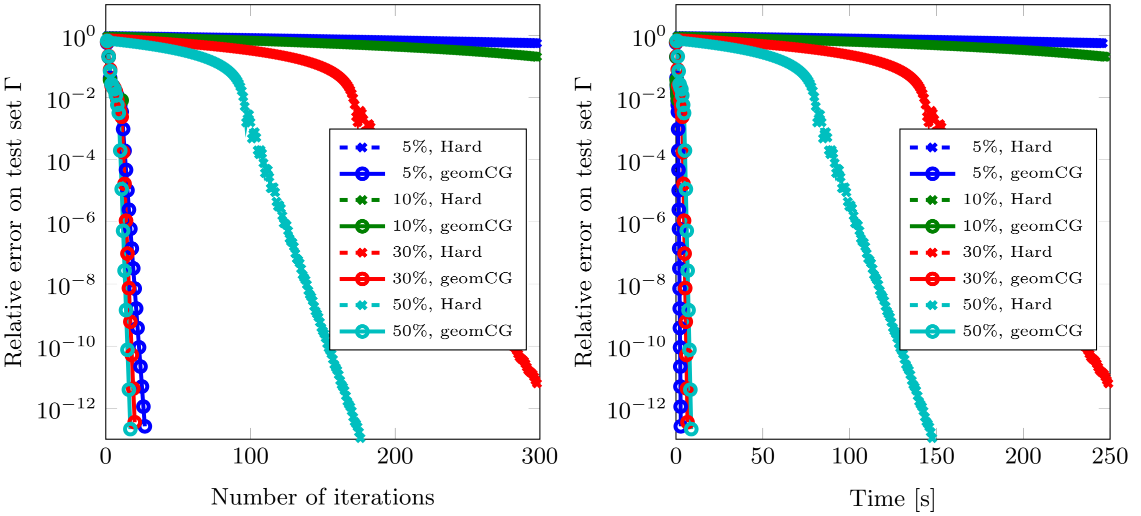


Figure 4: *Convergence curves for different sampling set sizes as functions of iterations and time for our proposed algorithm (geomCG) and the hard completion algorithm by Signoretto et al. [26]. Tensor size and multilinear rank fixed to* $n = 100$ *and* $\mathbf{r} = (5, 5, 5)$, *respectively.*

4.2.1 Reconstruction of Noisy Data

In this part, we investigate the convergence properties of Algorithm 1 in the presence of noise. The known entries of $\mathcal{A}$ are perturbed by rescaled Gaussian noise $\mathcal{E}$, such that $\| \mathrm{P}_\Omega \mathcal{E} \| = \epsilon_0 \| \mathrm{P}_\Omega \mathcal{A} \|$ for a prescribed noise level ϵ_0. Ideally, Algorithm 1 should return an approximation $\mathcal{X}^*$ at the level of ϵ_0, that is,

$$\| \mathrm{P}_\Omega \mathcal{X}^* - \mathrm{P}_\Omega(\mathcal{A} + \mathcal{E}) \| / \| \mathrm{P}_\Omega \mathcal{A} \| \approx \| \mathrm{P}_\Omega \mathcal{A} - \mathrm{P}_\Omega(\mathcal{A} + \mathcal{E}) \| / \| \mathrm{P}_\Omega \mathcal{A} \| = \epsilon_0.$$

To test that the noise does not lead to a misidentification of the rank of the underlying problem, we compare the case where we take the initial guess on the correct manifold to an uninformed rank-$(1, 1, 1)$ guess. There, we employ a heuristic rank adaptation strategy discussed in Section 4.3. We show in Figure 5 that in both cases we can indeed recover the original data up to the given noise level.

4.2.2 Size of the Sampling Set

It is well known that in the matrix case, the number of random samples needed to exactly recover the original matrix is at least $O(nr \log n)$ under standard assumptions; see e.g. [6, 13]. In the left plot of Figure 6, we present numerical experiments suggesting that a

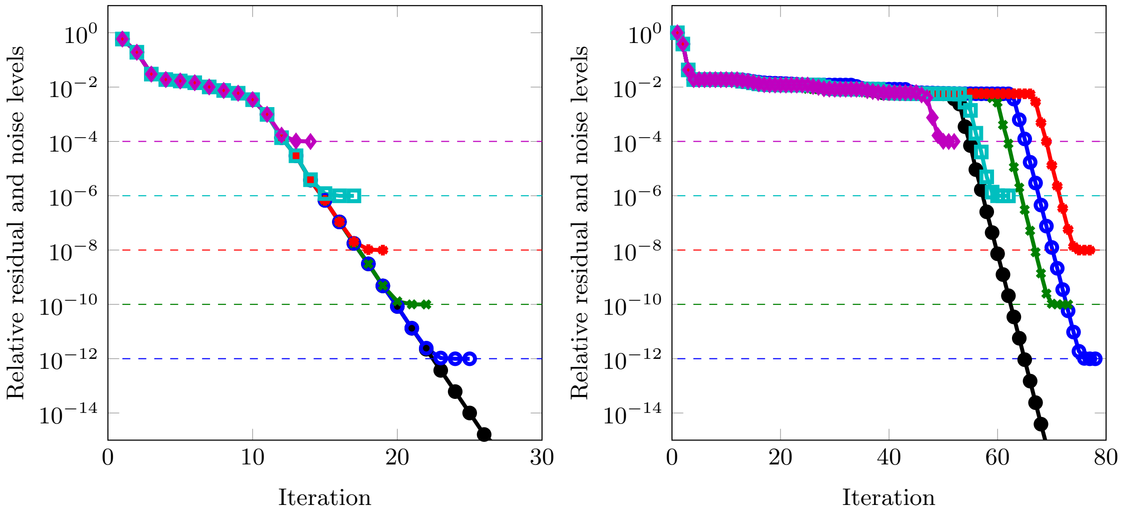


Figure 5: *Tensor completion from noisy measurements with $n = 100$, $\mathbf{r} = (6, 6, 6)$. The relative size of the sampling set was fixed to 10%. The black line corresponds to the noise-free case. The different colors correspond to the noise levels $\epsilon_0 \in \{10^{-4}, 10^{-6}, \ldots, 10^{-12}\}$.* ***Left:*** *Results when the underlying rank $\mathbf{r}$ is known.* ***Right:*** *Results for the case of unknown rank of the underlying problem. Due to the rank adaptation procedure, more iterations are necessary.*

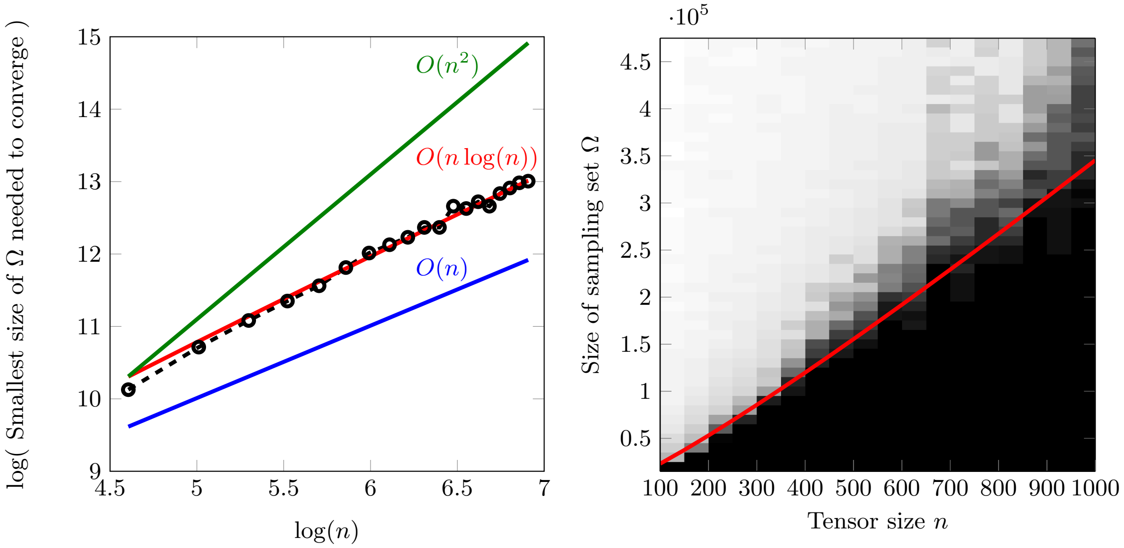


Figure 6: *Scaling of the sampling set size needed to reconstruct the original tensor of fixed multilinear rank $(10, 10, 10)$.* ***Left:*** *Minimum size of sampling set needed to attain convergence vs. tensor size $n = n_1 = n_2 = n_3$.* ***Right:*** *Phase transition of the convergence speed* (18)*. White means fast convergence, black means no convergence. The red line corresponds to $O(n\log(n))$.*

similar statement may hold for the three-dimensional tensor case. The algorithm is declared converged (and hence yields perfect reconstruction) if the relative residual drops below 10^{-6} within 100 iterations.

The right plot of Figure 6 displays a phase transition of the measured convergence speed of Algorithm 1, computed from

$$\rho = \left(\frac{\| \mathrm{P}_\Gamma \mathcal{X}_{k_{\text{end}}} - \mathrm{P}_\Gamma \mathcal{A} \|}{\| \mathrm{P}_\Gamma \mathcal{X}_{k_{\text{end}}-10} - \mathrm{P}_\Gamma \mathcal{A} \|} \right)^{\frac{1}{10}} \in [0,1], \tag{18}$$

where k_{end} is the final iteration.

4.3 Applications

In the following, we assess the performance of our algorithm on tensors derived from applications. In contrast to synthetic data sets, tensors from applications usually do not posses a clear, well-defined multilinear rank. Often, they exhibit a rather smooth decay of the singular values in each matricization. In such a setting, Algorithm 1 requires a good initial guess, as directly applying it with a (large) fixed rank $\mathbf{r}$ usually results in severe overfitting. We propose the following heuristic to address this problem: Starting from a multilinear rank-$(1,1,1)$-approximation, we iteratively increase the multilinear rank in each mode and rerun our algorithm with the previous result as initial guess. This procedure is repeated until the prescribed final multilinear rank $\mathbf{r}$ is reached. We increase the multilinear rank every time the current relative change in the square root of the cost function is smaller than a tolerance δ:

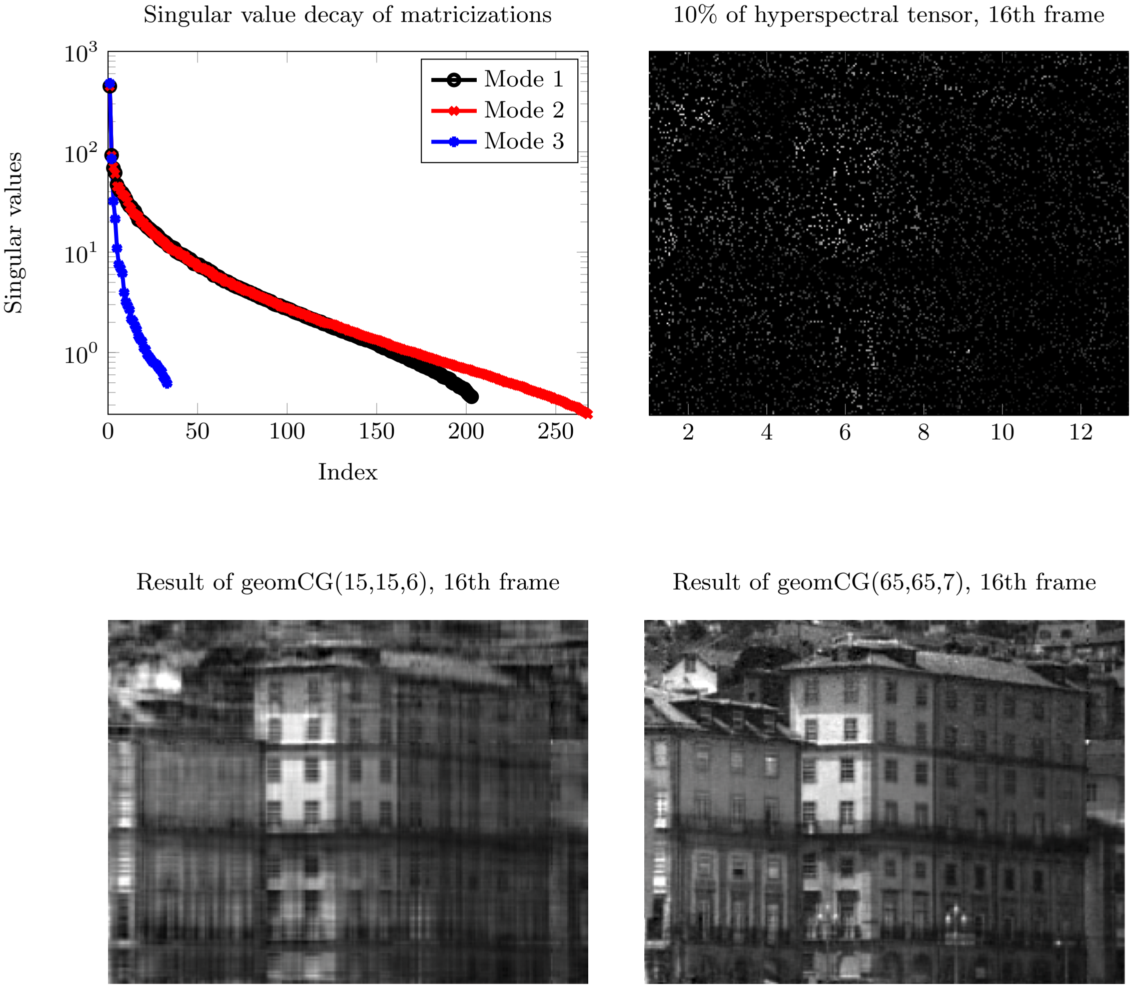
$$\left| \sqrt{f(\mathcal{X}_{i-1})} - \sqrt{f(\mathcal{X}_i)} \right| < \delta \sqrt{f(\mathcal{X}_i)}. \tag{19}$$

Initially, we use a large value for δ, say $\delta = 1$. We observed this approach to be effective at steering the algorithm into the direction of the optimal solution. Once we arrive at the final rank $\mathbf{r}$, we can also use (19) as a stopping criterion with a much smaller value for δ, say $\delta = 0.001$. In cases of convergence problems, the initial value for δ should be chosen smaller, at the cost of additional iterations. In the following numerical experiments, we always include this initialization procedure in the reported computation times.

4.3.1 Hyperspectral Image

As a first application of our algorithm to real-world data, we consider the hyperspectral image "Ribeira" [10] discussed in [27]. This results in a tensor of size $1017 \times 1340 \times 33$, where each slice corresponds to an image of the same scene measured at a different wavelength. To provide a faithful comparison, we proceed in the same way as in [27] and resize the tensor to $203 \times 268 \times 33$ by bilinear interpolation before applying our algorithm. The results are shown in Table 1. The reconstruction quality is assessed in terms of the *normalized root mean squared error*:

$$\mathrm{NRMSE}(\mathcal{X}, \mathcal{A}) := \frac{\| \mathrm{P}_{\Omega^c} \mathcal{A} - \mathrm{P}_{\Omega^c} \mathcal{X} \|}{(\max(\mathrm{P}_{\Omega^c} \mathcal{A}) - \min(\mathrm{P}_{\Omega^c} \mathcal{A}))\sqrt{|\Omega^c|}},$$



Singular value decay of matricizations

10% of hyperspectral tensor, 16th frame

Result of geomCG(15,15,6), 16th frame

Result of geomCG(65,65,7), 16th frame

Figure 7: *Full hyperspectral image data set "Ribeira" scaled to size* $(203, 268, 33)$. ***Top left:*** *Singular value decay of each matricization.* ***Top right:*** *The sampled tensor* $\mathrm{P}_\Omega \mathcal{A}$ *with 10% known entries. Unknown entries are marked in black.* ***Bottom left:*** *Result of our algorithm with iterative increase of ranks up to a final rank of* $\mathbf{r} = (15, 15, 6)$, *corresponding to entry geomCG*$(15, 15, 6)$ *in Table 1.* ***Bottom right:*** *Result of our algorithm with iterative increase of ranks up to a final rank of* $\mathbf{r} = (65, 65, 7)$, *corresponding to entry geomCG*$(65, 65, 7)$ *in Table 1.*

	Full Ribeira data set — sampling percentage					
	10%		30%		50%	
	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]
frame [27]	0.092	3.78	0.061	3.72	0.046	2.30
mode-3 [27]	0.068	0.27	0.018	0.31	**0.012**	0.33
tensor [27]	0.072	26.3	0.031	25.8	0.020	42.0
geomCG$(15, 15, 6)$	0.047	0.06	0.046	0.11	0.046	0.19
geomCG$(65, 65, 7)$	**0.025**	1.67	**0.017**	4.33	0.017	6.86
	First 5 frames of Ribeira data set — sampling percentage					
	10%		30%		50%	
	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]
frame [27]	0.071	0.15	0.046	0.14	0.034	0.14
mode-3 [27]	0.191	0.02	0.119	0.02	0.070	0.02
tensor [27]	0.067	3.14	0.034	4.48	0.023	4.06
geomCG$(15, 15, 5)$	**0.058**	0.01	0.033	0.02	0.032	0.03
geomCG$(55, 55, 5)$	0.075*	0.15	**0.026**	0.36	**0.016**	0.42

Table 1: *Reconstruction results for "Ribeira" hyperspectral image. The results for* frame, mode-3 *and* tensor *are taken from [27]. geomCG(r_1, r_2, r_3) denotes the result of Algorithm 1 using a prescribed final multilinear rank* (r_1, r_2, r_3).

where Ω^c is the complement of the sampling set, that is, the unknown entries. We compare with the results reported in [27] for the tensor completion algorithm *tensor*, the frame-wise matrix completion approach *frame*, and matrix completion applied to the *mode-3* matricization only. As shown in Figure 7, the singular values of the matricizations decay at a different rate. We take this into account in our algorithm, by choosing the final mode-1 and mode-2 ranks of the approximation significantly larger than the mode-3 rank. It can be observed that our algorithm (*geomCG*) yields very competitive results, especially in the case where the sampling set is small. There is one case of overfitting for geomCG$(55, 55, 5)$, marked by a star.

4.3.2 Reconstruction of Function Data

To investigate the applicability of our algorithm to compress tensors related to functions with singularities, we consider

$$f : [-1, 1]^3 \to \mathbb{R}, \quad x \mapsto \mathrm{e}^{-\|x\|_2} \tag{20}$$

discretized on a uniform tensor grid with mesh width $h = 1/100$. The function values are collected in a tensor $\mathcal{A} \in \mathbb{R}^{201\times 201\times 201}$. In this setting, we assume that the location of the singularity is known a priori. As f has a cusp at the origin, the information in $\mathcal{A}$ is strongly localized at this point and tensor completion applied naively to $\mathcal{A}$ would not lead to reasonable compression. To avoid this effect, we therefore cut out a small hypercube $[-0.1, 0.1]^3$, corresponding to the $21 \times 21 \times 21$ central part of the discretized tensor. The idea is to not include this region in the sampling set Ω. The entries corresponding to this region are stored separately and reconstructed exactly after performing low-rank tensor completion on the remaining region. We therefore do also not include the central part in the test set Γ when verifying the accuracy of the completed tensor. The obtained results are shown in Figure 8. Already sampling 5% of the entries gives an accuracy of 10^{-5}. This would yield a compression ratio of 5.1% if we stored the involved entries. However, storing the rank-$(5, 5, 5)$ approximation along with the central part yields the significantly lower compression ratio of 0.15%.

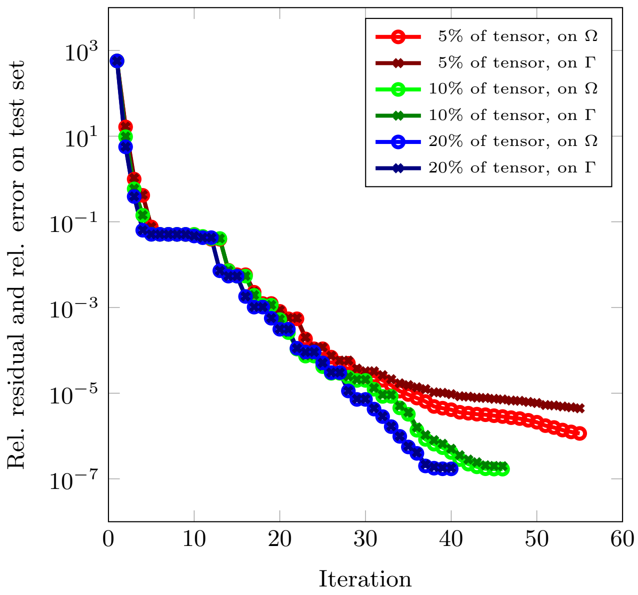


Figure 8: *Convergence of the algorithm for the discretization of a function with cusp* (20). *The final rank of the approximation is* $\mathbf{r} = (10, 10, 10)$. *The part corresponding to* $[-0.1, 0.1]^3$ *is excluded from the sampling set* Ω *and the test set* Γ.

4.3.3 Stochastic Elliptic PDE with Karhunen-Loève Expansion

Finally, we consider an elliptic PDE with stochastic coefficients:

$$\begin{aligned} -\nabla\left(a(x,y)\nabla u(x,y)\right) &= f(x), & (x,y) &\in D \times \Theta, \\ u(x,y) &= 0 & (x,y) &\in \partial D \times \Theta, \end{aligned}$$

where $y \in \Theta$ is a random variable and $D = [-1, 1]$ is the computational domain. We represent the stochastic variable y by an infinite number of parameters $\alpha \in [-1, 1]^\infty$ and write $a(x, \alpha)$ in terms of its Karhunen-Loève expansion

$$a(x,\alpha) = a_0 + \sum_{\mu=1}^{\infty} \sqrt{\lambda_\mu} a_\mu(x) \alpha_\mu,$$

where $a_\mu(x)$, $\mu = 1, 2, \dots$ are normalized $L^2(D)$-functions and the so called Karhunen-Loève eigenvalues $\lambda_\mu \geq 0$ decrease monotonically. We truncate the Karhunen-Loève expansion after $\mu = 3$ and employ a standard piecewise linear finite element (FE) discretization. This yields a parameter-dependent linear system of equations,

$$(A_0 + \alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3)x = f, \tag{21}$$

where each $A_\mu \in \mathbb{R}^{m \times m}$ is the FE stiffness matrix corresponding to the coefficient a_μ. We refer to, e.g., [24] for a detailed discussion of this procedure. In our examples, we choose

$$a_0(x) = 1, \quad a_\mu(x) = \sin(\mu x).$$

The parameters α are then sampled uniformly on a tensor grid on $[-1,1] \times [-1,1] \times [-1,1]$. Assuming that we are only interested in the mean of the solution for a specific set of parameters, this results in the solution tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$, where each entry of this tensor requires the solution of a discretized PDE (21) for one combination of the discretized $(\alpha_1, \alpha_2, \alpha_3)$. Hence, evaluating the full tensor is fairly expensive. Using tensor completion, we sample $\mathcal{X}$ at (few) randomly choosen points and try to approximate the missing entries.

In Figure 9 we show the results of this approach for $m = 50, n = 100$, and two different choices of α_μ. We used the Karhunen-Loève eigenvalues $\sqrt{\lambda_\mu} = 5\exp(-2\mu)$ and $\sqrt{\lambda_\mu} = (1+\mu)^{-2}$, respectively. As the second choice results in slower singular value decays, our algorithm requires more iterations to attain the same accuracy. Using 5% of the tensor as a sampling set is in both cases sufficient to recover the original tensor to good precision. As the sampling set gets smaller, overfitting of the sampling data is more likely to occur, especially for the second choice of λ_μ.

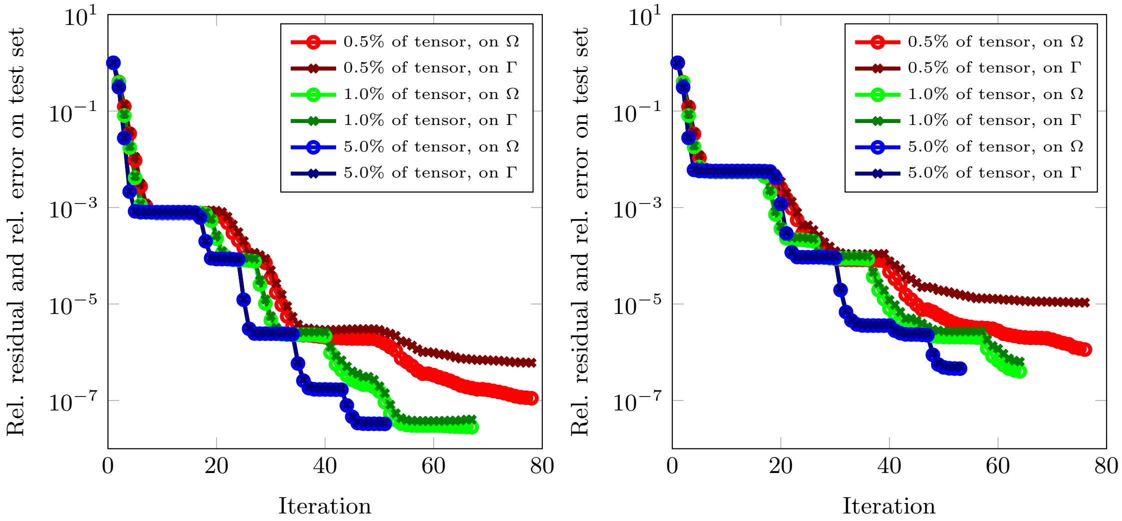


Figure 9: *Convergence for the solution tensor of size $n = 100$ of a parametrized linear system obtained from a discretized stochastic PDE.* ***Left:*** *Result for Karhunen-Loève eigenvalues $\sqrt{\lambda_\mu} = 5\exp(-2\mu)$.* ***Right:*** *Result for Karhunen-Loève eigenvalues $\sqrt{\lambda_\mu} = (1+\mu)^{-2}$. The final rank of the solution is $\mathbf{r} = (4,4,4)$.*

5 Conclusions

We have shown that the framework of Riemannian optimization yields a very effective nonlinear CG method for performing tensor completion. Such a method has also been suggested in [8]. One of the main contributions in this paper consists of a careful discussion of the algorithmic and implementation details, showing that the method scales well for large data sets and is competitive to existing methods for tensor completion. On the theoretical side, we have proven that HOSVD satisfies the properties of a retraction and discussed the convergence properties of the nonlinear CG method.

The numerical experiments indicate the usefulness of tensor completion not only for data-related but also for function-related tensors. We feel that this aspect merits further exploration. To handle high-dimensional applications, the approach considered in this paper needs to be extended to other SVD-based low-rank tensor formats, such as the tensor train and the hierarchical Tucker formats, see also [8].

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds.* Princeton University Press, Princeton, NJ, 2008.

[2] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM J. Control Optim.*, 22(1):135–158, 2012.

[3] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometr. Intell. Lab.*, 106:41–56, 2011.

[4] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. Available from `http://www.sandia.gov/\textasciitilde{}tgkolda/TensorToolbox/`, January 2012.

[5] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[6] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory*, 56(5):2053–2080, 2009.

[7] J.-L. Chern and L. Dieci. Smoothness and periodicity of some matrix decompositions. *SIAM J. Matrix Anal. Appl.*, 22(3):772–792, 2000.

[8] C. Da Silva and F. J. Herrmann. Hierarchical Tucker tensor optimization – Applications to tensor completion. In *Proc. 10th International Conference on Sampling Theory and Applications*, 2013.

[9] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[10] D. H. Foster, S. M. C. Nascimento, and K. Amano. Information limits on neural identification of colored surfaces in natural scenes. *Visual Neurosci.*, 21:331–336, 2004.

[11] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[12] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitt.*, 36(1):53–78, 2013.

[13] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *J. Mach. Learn. Res.*, 11:2057–2078, 2010.

[14] O. Koch and Ch. Lubich. Dynamical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 31(5):2360–2375, 2010.

[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[16] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. In *Proc. IEEE 12th International Conference on Computer Vision*, pages 2114–2121, 2009.

[17] Y. Liu and F. Shang. An efficient matrix factorization method for tensor completion. *IEEE Signal Processing Letters*, 20(4):307–310, April 2013.

[18] Y. Ma, J. Wright, A. Ganesh, Z. Zhou, K. Min, S. Rao, Z. Lin, Y. Peng, M. Chen, L. Wu, E. Candès, and X. Li. Low-rank matrix recovery and completion via convex optimization. Survey website, `http://perception.csl.illinois.edu/matrix-rank/`. Accessed: 22. April 2013.

[19] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. arXiv:1209.0430, 2012.

[20] C. Mu, B. Huang, J. Wright, and D. Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. arXiv:1307.5870, 2013.

[21] T. Ngo and Y. Saad. Scaled gradients on Grassmann manifolds for matrix completion. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1421–1429. 2012.

[22] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edition, 2006.

[23] H. Rauhut, R. Schneider, and Z. Stojanac. Low rank tensor recovery via iterative hard thresholding. In *Proc. 10th International Conference on Sampling Theory and Applications*, 2013.

[24] C. Schwab and C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs. *Acta Numerica*, 20:291–467, 2011.

[25] M. Signoretto, L. De Lathauwer, and J. A. K. Suykens. Nuclear norms for tensors and their use for convex multilinear estimation. Technical Report 10-186, K. U. Leuven, 2010.

[26] M. Signoretto, Q. Tran Dinh, L. De Lathauwer, and J. A. K. Suykens. Learning with tensors: a framework based on convex optimization and spectral regularization. Technical Report 11-129, K. U. Leuven, 2011.

[27] M. Signoretto, R. Van de Plas, B. De Moor, and J. A K Suykens. Tensor versus matrix completion: A comparison with application to spectral data. *IEEE Signal Processing Letters*, 18(7):403–406, 2011.

[28] A. Uschmajew. *Zur Theorie der Niedrigrangapproximation in Tensorprodukten von Hilberträumen.* PhD thesis, Technische Universität Berlin, 2013.

[29] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra Appl.*, 439(1):133–166, 2013.

[30] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.*, 23(2):1214—1236, 2013.