ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MATHICSE

Mathematics Institute of Computational Science and Engineering

School of Basic Sciences - Section of Mathematics

MATHICSE Technical Report

Nr. 05.2015

March 2015

Riemannian optimization for high-dimensional tensor completion

Michael Steinlechner

http://mathicse.epfl.ch

Phone: +41 21 69 37648

Address:
EPFL - SB - MATHICSE (Bâtiment MA)
Station 8 - CH-1015 - Lausanne - Switzerland

Fax: +41 21 69 32545

RIEMANNIAN OPTIMIZATION FOR HIGH-DIMENSIONAL TENSOR COMPLETION

MICHAEL STEINLECHNER*

Abstract. Tensor completion aims to reconstruct a high-dimensional data set with a large fraction of missing entries. The assumption of low-rank structure in the underlying original data allows us to cast the completion problem into an optimization problem restricted to the manifold of fixed-rank tensors. Elements of this smooth embedded submanifold can be efficiently represented in the tensor train (TT) or matrix product states (MPS) format with storage complexity scaling linearly with the number of dimensions. We present a nonlinear conjugate gradient scheme within the framework of Riemannian optimization which exploits this favorable scaling. Numerical experiments and comparison to existing methods show the effectiveness of our approach for the approximation of multivariate functions. Finally, we show that our algorithm can obtain competitive reconstructions from uniform random sampling of few entries of compared to adaptive sampling techniques such as cross-approximation.

Key words. Completion, data recovery, high-dimensional problems, low-rank approximation, tensor train format, matrix product states, Riemannian optimization, curse of dimensionality

AMS subject classifications. 15A69, 65K05, 65F99, 58C05

1. Introduction. In this paper, we consider the completion of a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ in which the vast majority of entries are unknown. Depending on the source of the data set represented by $\mathbf{A}$, assuming an underlying low-rank structure is a sensible choice. In particular, this is the case for discretizations of functions that are well-approximated by a short sum of separable functions. Low-rank models have been shown to be effective for various applications, such as electronic structure calculations and approximate solutions of stochastic and parametric PDEs, see [17] for an overview. With this assumption, the tensor completion problem can be cast into the optimization problem

$$\begin{aligned} \min_{\mathbf{X}} \quad & f(\mathbf{X}) := \frac{1}{2} \| \mathrm{P}_\Omega \, \mathbf{X} - \mathrm{P}_\Omega \, \mathbf{A} \|^2 \\ \text{subject to} \quad & \mathbf{X} \in \mathcal{M}_{\mathbf{r}} := \left\{ \mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \;\middle|\; \operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) = \mathbf{r} \right\}. \end{aligned} \tag{1.1}$$

where $\operatorname{rank}_{\mathrm{TT}}(\mathbf{X})$ denotes the *tensor train rank* of the tensor $\mathbf{X}$, a certain generalization of the matrix rank to the tensor case, see Section 3 for details. We denote with P_Ω the projection onto the *sampling set* $\Omega \subset [1, n_1] \times \cdots \times [1, n_d]$ corresponding to the indices of the known entries of $\mathbf{A}$,

$$\mathrm{P}_\Omega \, \mathbf{X} := \begin{cases} \mathbf{X}(i_1, i_2, \ldots, i_d) & \text{if } (i_1, i_2, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

In the two-dimensional case $d = 2$, tensor completion (1.1) reduces to the extensively studied *matrix completion* for which convergence theory and efficient algorithms have been derived in the last years, see [27] for an overview. The number of entries in the

*MATHICSE-ANCHP, Section de Mathématiques, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland. E-mail: michael.steinlechner@epfl.ch

This work 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.* Parts of this work have been conducted during a research stay at PACM, Princeton University, with a mobility grant from the SNSF.

optimization variable $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ scales exponentially with d, the so-called *curse of dimensionality*. Hence, the naive approach of reshaping the given data $\mathbf{A}$ into a large matrix and applying existing algorithms for matrix completion is computationally infeasible for all but very small problem sizes.

Assuming low-rank structure in the different tensor formats along with their notion of tensor rank allows us to reduce the degrees of freedom significantly. For an overview of the most common choices of tensor formats, we refer to the literature survey by Grasedyck *et al.* [17]. The most popular choice, the *Tucker* tensor format, reduces the storage complexity from $O(n^d)$ to $O(dnr + r^d)$ and thus yields good results for not too high-dimensional problems, say $d \leq 5$. Algorithms developed for the Tucker format range from alternating minimization [23, 24] and convex optimization, using various generalizations of the nuclear norm [23, 24, 13, 36, 37, 38], to Riemannian optimization on the embedded submanifold of tensors with fixed Tucker rank [22].

If higher dimensional problems are considered, the Tucker format can be replaced by the *hierarchical Tucker* (HT) or alternatively the *tensor train / matrix product states* (TT/MPS) formats. In the TT/MPS format, for example, the number of unknowns in a tensor with fixed rank r is scaling like $O(dnr^2 - dr^2)$, which potentially allows for the efficient treatment of problems with a much higher number of dimensions d than the Tucker format. For the HT format, Da Silva and Herrmann [9, 10] have derived a fast Riemannian approach closely related to the algorithm presented in this paper. In the TT/MPS format Grasedyck *et al.* [15] presented an alternating direction method with similar computational complexity. We will compare these two methods to our algorithm and to a simple alternating least squares approach.

Note that for tensor completion, the index set Ω of given entries is fixed. If we relax this constraint and instead try to reconstruct the original tensor $\mathbf{A}$ from as few entries as possible, black-box approximation techniques such as *cross-approximation* are viable alternatives. In the matrix case, cross approximation was first developed for the approximation of integral operators [5, 6, 7] and as the so-called *Skeleton decomposition* [14, 40]. Extensions to high dimensional tensors in the TT/MPS format were developed by Oseledets *et al.* [32, 35] and by Ballani *et al.* [4, 3] for the HT format. In these techniques, the sampling points are chosen adaptively during the runtime of the algorithm. In this paper, we will compare them to our proposed algorithm and show in numerical experiments that choosing randomly distributed sampling points beforehand is competitive in certain applications.

This paper is a generalization of the Riemannian approach presented in [42, 22] to a higher-dimensional setting using this TT/MPS format. We obtain an algorithm that scales linear in d, linear in n and polynomial in r and show its applicability to very high-dimensional problems.

We will first state in Section 2 the final Riemannian optimization algorithm to give an overview of the necessary ingredients and quantities. Then we will briefly recapitulate the TT/MPS format and its geometric properties in Section 3, and explain each individual part of the optimization algorithm in detail in Section 4. We will focus on efficient computation of each part and give detailed analysis of the involved computational complexity to solve the tensor completion problem (1.1). Finally, we apply our algorithm to several test cases and compare its performance to other existing methods in Section 5.

2. A Riemannian nonlinear conjugate gradient scheme. To solve the tensor completion problem (1.1), we will use a nonlinear conjugate gradient scheme restricted to $\mathcal{M}_{\mathbf{r}}$, the set of TT/MPS tensors with fixed rank. As we will discuss in

Section 3, this set forms a smooth embedded submanifold of $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and we can apply a Riemannian version [1, Sec. 8.3] of the usual nonlinear CG algorithm. The main difference is the fact that we need to make sure that every iterate $\mathbf{X}_k$ of the optimization scheme stays on the Manifold $\mathcal{M}_\mathbf{r}$ and that we use the *Riemannian gradient*, which points to the direction of greatest increase within the tangent space $T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}$. Taking a step from the starting point $\mathbf{X}_0 \in \mathcal{M}_\mathbf{r}$ in the direction of the negative Riemannian gradient η_0 yields a new iterate which is, in general, not an element of $\mathcal{M}_\mathbf{r}$ anymore. Hence, we will need the concept of a *retraction* R which maps the new iterate $\mathbf{X}_0 + \alpha_0\eta_0$ back to the manifold. Furthermore, as the conjugate gradient scheme compares the current gradient $\xi_k \in T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}$ with the previous search direction $\eta_{k-1} \in T_{\mathbf{X}_{k-1}}\mathcal{M}_\mathbf{r}$, we first need to bring $\mathbf{X}_{k-1}$ into the current tangent space using the *vector transport* $\mathcal{T}_{\mathbf{X}_{k-1}\to\mathbf{X}_k}$.

Algorithm 1 Geometric Nonlinear CG for Tensor Completion, see also [22]

Input: Initial guess $\mathbf{X}_0 \in \mathcal{M}_\mathbf{r}$.

$\xi_0 \leftarrow \operatorname{grad} f(\mathbf{X}_0)$ *% compute Riemannian gradient*
$\eta_0 \leftarrow -\xi_0$ *% first step is steepest descent*
$\alpha_0 \leftarrow \operatorname{argmin}_\alpha f(\mathbf{X}_0 + \alpha\eta_0)$ *% step size by linearized line search*
$\mathbf{X}_1 \leftarrow R(\mathbf{X}_0, \alpha_0\eta_0)$ *% obtain next iterate by retraction*

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

The resulting optimization scheme is given by Algorithm 1. Note that the basic structure does not depend on the representation of $\mathbf{X}$ as a TT/MPS tensor and is exactly the same as in the Tucker case [22]. We will derive and explain each part of the algorithm in the following Section 4: the computation of the Riemannian gradient, Sec. 4.2, the line search procedure, Sec. 4.5, the retraction, Sec. 4.3, the direction updating rule and vector transport, Sec. 4.4 and the stopping criteria, Sec. 4.6.

3. The geometry of tensors in the TT/MPS format. In this section, we will give a brief overview of the TT/MPS representation and fix the notation for the rest of this paper. The notation follows the presentation in [21]. For a more detailed description of the TT/MPS format, we refer to the original paper by I. Oseledets [30]. We will review the manifold structure of TT/MPS tensors with fixed rank and present different representations of elements in the tangent space. This will then allow us to derive our optimization scheme in Section 4.

3.1. The TT/MPS representation of tensors. A d-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ can be reshaped into a matrix

$$\mathbf{X}^{<\mu>} \in \mathbb{R}^{(n_1 n_2 \cdots n_\mu) \times (n_{\mu+1} \cdots n_d)},$$

the so-called μth *unfolding* of $\mathbf{X}$, see [30] for a formal definition of the ordering of the indices. This allows us to define the *tensor train rank* $\operatorname{rank}_{\mathrm{TT}}$ as the following

$(d+1)$-tuple of ranks,

$$\operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) = \mathbf{r} = (r_0, r_1, \ldots, r_d) := \left(1, \operatorname{rank}(\mathbf{X}^{<1>}), \ldots, \operatorname{rank}(\mathbf{X}^{<d-1>}), 1\right),$$

where we define $r_0 = r_d = 1$.

Given these ranks $(r_0, \ldots r_d)$, it is possible to write every entry of the d-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ as a product of d matrices,

$$\mathbf{X}(i_1, i_2, \ldots, i_d) = U_1(i_1) U_2(i_2) \cdots U_d(i_d), \tag{3.1}$$

where each $U_\mu(i_\mu)$ is a matrix of size $r_{\mu-1} \times r_\mu$ for $i_\mu = 1, 2, \ldots, n_\mu$. The slices $U_\mu(i_\mu)$, $i_\mu = 1, 2, \ldots, n_\mu$ can be combined into third order tensors $\mathbf{U}_\mu$ of size $r_{\mu-1} \times n_\mu \times r_\mu$, the *cores* of the TT/MPS format. In terms of these core tensors, equation (3.1) can also be written as

$$\mathbf{X}(i_1, \ldots, i_d) = \sum_{k_1=1}^{r_1} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \mathbf{U}_1(1, i_1, k_1) \mathbf{U}_2(k_1, i_2, k_2) \cdots \mathbf{U}_d(k_{d-1}, i_d, 1).$$

To be able to exploit the product structure of (3.1), we now focus on ways to manipulate individual cores. We define the *left* and the *right unfoldings*,

$$\mathbf{U}_\mu^{\mathsf{L}} \in \mathbb{R}^{r_{\mu-1} n_\mu \times r_\mu}, \quad \text{and} \quad \mathbf{U}_\mu^{\mathsf{R}} \in \mathbb{R}^{r_{\mu-1} \times n_\mu r_\mu}$$

by reshaping $\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$ into matrices of size $r_{\mu-1} n_\mu \times r_\mu$ and $r_{\mu-1} \times n_\mu r_\mu$, respectively. Additionally, we can split the tensor $\mathbf{X}$ into left and right parts, the *interface matrices*

$$\mathbf{X}_{\leq \mu}(i_1, \ldots, i_\mu) = U_1(i_1) U_2(i_2) \cdots U_\mu(i_\mu) \in \mathbb{R}^{n_1 n_2 \cdots n_\mu \times r_\mu},$$

$$\mathbf{X}_{\geq \mu}(i_\mu, \ldots, i_d) = [U_\mu(i_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d)]^{\mathsf{T}} \in \mathbb{R}^{n_\mu n_{\mu+1} \cdots n_d \times r_{\mu-1}}.$$

Figure 3.1 depicts a TT/MPS tensor of order 6. The interface matrices can be constructed recursively:

$$\mathbf{X}_{\leq \mu} = (I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}) \mathbf{U}_\mu^{\mathsf{L}}, \quad \text{and} \quad \mathbf{X}_{\geq \mu}^{\mathsf{T}} = \mathbf{U}_\mu^{\mathsf{R}} (\mathbf{X}_{\geq \mu+1}^{\mathsf{T}} \otimes I_{n_\mu}), \tag{3.2}$$

where I_m denotes the $m \times m$ identity matrix and $\otimes$ the Kronecker product.

The following formula connects the μth unfolding of a tensor with its interface matrices:

$$\mathbf{X}^{<\mu>} = \mathbf{X}_{\leq \mu} \mathbf{X}_{\geq \mu+1}^{\mathsf{T}}.$$

We call $\mathbf{X}$ μ-orthogonal if

$$\begin{aligned}
&\mathbf{X}_{\leq \nu}^{\mathsf{T}} \mathbf{X}_{\leq \nu} = I_{r_\nu} && \text{and hence} \quad (\mathbf{U}_\nu^{\mathsf{L}})^{\mathsf{T}} \mathbf{U}_\nu^{\mathsf{L}} = I_{r_\nu}, && \text{for all } \nu = 1, \ldots, \mu - 1, \\
&\mathbf{X}_{\geq \nu} \mathbf{X}_{\geq \nu}^{\mathsf{T}} = I_{r_{\nu-1}}, && \text{and hence} \quad \mathbf{U}_\nu^{\mathsf{R}} (\mathbf{U}_\nu^{\mathsf{R}})^{\mathsf{T}} = I_{r_{\nu-1}}, && \text{for all } \nu = \mu + 1, \ldots, d.
\end{aligned}$$

The tensor $\mathbf{X}$ is called left-orthogonal if $\mu = d$ and right-orthogonal if $\mu = 1$. Orthogonalizing a given TT/MPS tensor can be done efficiently using recursive QR decompositions, as we have for the left-orthogonal part:

$$\mathbf{X}_{\leq \mu} = Q_{\leq \mu} R_\mu, \quad \text{with} \quad Q_{\leq \mu} = (I_{n_\mu} \otimes Q_{\leq \mu-1}) Q_\mu^{\mathsf{L}} \quad \text{and} \quad Q_{\leq 0} = 1.$$

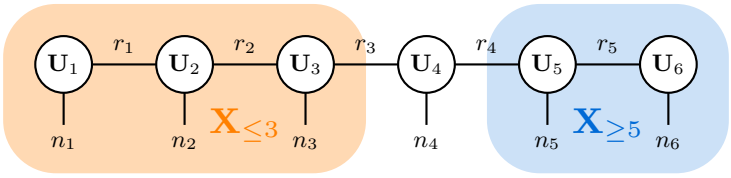


Fig. 3.1: *Representation of a TT/MPS tensor of order $d = 6$. As $r_0 = r_d = 1$, they are usually omitted. The two interface matrices $\mathbf{X}_{\leq 3}$ and $\mathbf{X}_{\geq 5}$ are shown. See also [21].*

A similar relation holds for the right-orthogonal part, see [30] for more details. Moving from a μ-orthogonal tensor to a $(\mu+1)$-orthogonal one only requires a QR decomposition of $\mathbf{U}_\mu^{\mathsf{L}}$. An important consequence that we will use in Section 3.3 is that changing the orthogonalization of a tensor only affects its internal representation and does not change the tensor itself.

As an inner product of two TT/MPS tensors we take the standard Euclidean inner product of the vectorizations $\operatorname{vec}(\mathbf{X}), \operatorname{vec}(\mathbf{Y}) \in \mathbb{R}^{n_1 n_2 \cdots n_d}$:

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \langle \operatorname{vec}(\mathbf{X}), \operatorname{vec}(\mathbf{Y}) \rangle = \operatorname{trace}(\mathbf{X}^{<\mu>} \mathbf{Y}^{<\mu>}) \quad \text{for any } \mu \in \{1, \dots d-1\}. \tag{3.3}$$

The norm of a TT/MPS tensor is induced by the inner product. Note that if $\mathbf{X}$ is μ-orthogonal, then we obtain

$$\|\mathbf{X}\| = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle} = \|\mathbf{U}_\mu\| = \|\mathbf{U}_\mu^{\mathsf{L}}\|_F = \|\operatorname{vec}(\mathbf{U}_\mu)\|,$$

so we only have to compute the norm of the μth core.

The recursive relation (3.2) allows us to extend the singular value decomposition to TT/MPS tensors, the *TT-SVD* [30] procedure. Let $\mathbf{X}$ be d-orthogonal. By taking the SVD of $\mathbf{U}_d^{\mathsf{R}} = QSV^{\mathsf{T}}$ and splitting the product to the adjacent cores:

$$\mathbf{U}_d^{\mathsf{R}} \leftarrow V^{\mathsf{T}}, \qquad \mathbf{U}_{d-1}^{\mathsf{L}} \leftarrow \mathbf{U}_{d-1}^{\mathsf{L}} QS,$$

the tensor is now $(d-1)$-orthogonalized. By keeping only s singular values, the rank r_{d-1} between $\mathbf{U}_{d-1}$ and $\mathbf{U}_d$ is reduced to s. Repeating this procedure till core $\mathbf{U}_1$ allows us to truncate the rank $\mathbf{r}$ of $\mathbf{X}$ to any prescribed value $\tilde{\mathbf{r}}$, with $\tilde{r}_\mu \leq r_\mu$ for all $\mu = 1, \dots, d$. Similar to the best-approximation property of the matrix SVD, the truncated $\widetilde{\mathbf{X}}$ fulfills the following inequality:

$$\|\mathbf{X} - \widetilde{\mathbf{X}}\| \leq \sqrt{d-1} \|\mathbf{X} - \mathbf{X}^*\|, \tag{3.4}$$

with $\mathbf{X}^*$ being the best approximation to $\mathbf{X}$ within the set of rank-$\tilde{\mathbf{r}}$ TT/MPS tensors, see [30, Cor. 2.4] for a proof.

3.2. Manifold of TT/MPS tensors of fixed rank. It was shown by Holtz *et al.* [19] that the set

$$\mathcal{M}_{\mathbf{r}} = \left\{ \mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \;\middle|\; \operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) = \mathbf{r} \right\}$$

forms a smooth embedded submanifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ of dimension

$$\dim \mathcal{M}_{\mathbf{r}} = \sum_{\mu=1}^{d} r_{\mu-1} n_\mu r_\mu - \sum_{\mu=1}^{d-1} r_\mu^2.$$

The inner product (3.3) induces an Euclidean metric on the embedding space $\mathbb{R}^{n_1 \times \cdots \times n_d}$. When equipped with this metric, $\mathcal{M}_{\mathbf{r}}$ is turned into a Riemannian manifold. An analogous result is also available for the HT format [41].

3.3. The tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$. Assume $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is represented in the TT/MPS format as

$$\mathbf{X}(i_1, \ldots, i_d) = U_1(i_1) U_2(i_2) \cdots U_d(i_d)$$

with left-orthogonal cores $\mathbf{U}_\mu$, $\left(\mathbf{U}_\mu^{\mathsf{L}}\right)^{\mathsf{T}} \mathbf{U}_\mu^{\mathsf{L}} = I_{r_\mu}$, for all $\mu = 1, \ldots, d-1$. By the product rule, any element of the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be uniquely represented in the following way:

$$\begin{aligned} \mathbf{Y}(i_1, \ldots, i_d) = {} & \delta U_1(i_1) U_2(i_2) \cdots U_d(i_d) \\ & + U_1(i_1) \delta U_2(i_2) \cdots U_d(i_d) \\ & + \cdots \\ & + U_1(i_1) U_2(i_2) \cdots \delta U_d(i_d), \end{aligned} \tag{3.5}$$

with the gauge condition $\left(\mathbf{U}_\mu^{\mathsf{L}}\right)^{\mathsf{T}} \delta \mathbf{U}_\mu^{\mathsf{L}} = 0$, for all $\mu = 1, \ldots, d-1$. There is no gauge condition for the last core. This representation has been introduced by Holtz et al. [19].

For the orthogonal projection $\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ of a certain tensor $\mathbf{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, Lubich et al. [26] derived the following explicit expression for the first order variations $\delta \mathbf{U}_\mu$:

$$\delta \mathbf{U}_\mu^{\mathsf{L}} = \left(I_{n_\mu r_{\mu-1}} - \mathrm{P}_\mu\right) \left(I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}\right)^{\mathsf{T}} \mathbf{Z}^{<\mu>} \mathbf{X}_{\geq \mu+1} \left(\mathbf{X}_{\geq \mu+1}^{\mathsf{T}} \mathbf{X}_{\geq \mu+1}\right)^{-1}, \tag{3.6}$$

where P_μ denotes the orthogonal projection into the range of $\mathbf{U}_\mu^{\mathsf{L}}$. This expression is relatively straight-forward to implement using tensor contractions along the modes $1, \ldots, \mu-1, \mu+1, \ldots, d$. Note that the inverse $\left(\mathbf{X}_{\geq \mu+1}^T \mathbf{X}_{\geq \mu+1}\right)^{-1}$ potentially leads to numerical instabilities or is not even well-defined: For example, $\mathbf{X}_{\geq \mu+1}^T \mathbf{X}_{\geq \mu+1}$ is singular in case the right unfoldings $\mathbf{U}_{\mu+1}^{\mathsf{R}}, \ldots, \mathbf{U}_d^{\mathsf{R}}$ do not have full rank.

To circumvent these numerical instabilites, we propose a different parametrization of the tangent space. First, observe that in the representation (3.5), each individual summand is a TT/MPS tensor which *can be orthogonalized individually* without changing the overall tangent tensor. Hence, we can μ-orthogonalize the μth summand:

$$\begin{aligned} & U_1(i_1) \cdots U_{\mu-1}(i_{\mu-1}) \delta U_\mu(i_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d) \\ = {} & U_1(i_1) \cdots U_{\mu-1}(i_{\mu-1}) \delta V_\mu(i_\mu) V_{\mu+1}(i_{\mu+1}) \cdots V_d(i_d). \end{aligned} \tag{3.7}$$

For the rest of this article, left-orthogonal cores are always denoted by the letter U, and right-orthogonal cores by the letter V. The first order variations $\delta \mathbf{V}_\mu$ are neither left- nor right-orthogonal, but fulfill the gauge condition $(\mathbf{U}_\mu^{\mathsf{L}})^{\mathsf{T}} \delta \mathbf{V}_\mu^{\mathsf{L}} = 0, \mu = 1, \ldots, d-1$. Because of the right-orthogonalization of $\mathbf{X}_{\mu+1}$, the projection (3.6) simplifies to

$$\delta \widetilde{\mathbf{U}}_\mu^{\mathsf{L}} = \left(I_{n_\mu r_{\mu-1}} - \mathrm{P}_\mu\right) \left(\mathbf{I}_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}\right)^T Z^{\langle \mu \rangle} \mathbf{X}_{\geq \mu+1}, \tag{3.8}$$

because now $\left(\mathbf{X}_{\geq \mu+1}^{\mathsf{T}} \mathbf{X}_{\geq \mu+1}\right)^{-1}$ is the identity matrix. Hence, we are able to circumvent the explicit inverse. Due to the individual orthogonalizations of the summands, a tangent tensor now has the form

$$\begin{aligned} \mathbf{Y}(i_1, \ldots, i_d) = {} & \delta V_1(i_1) V_2(i_2) \cdots V_d(i_d) \\ & + U_1(i_1) \delta V_2(i_2) \cdots V_d(i_d) \\ & + \cdots \\ & + U_1(i_1) U_2(i_2) \cdots \delta V_d(i_d). \end{aligned} \tag{3.9}$$

Making use of the structure of the sum (3.9), we observe that a tangent tensor in $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can also be represented as a TT/MPS tensor:

PROPOSITION 3.1. *Let $\mathbf{Y} \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ be given in the representation* (3.9)*. Then, it can be identified with a TT/MPS tensor of TT-rank at most* $(1, 2r_1, 2r_2, \ldots, 2r_{d-1}, 1)$*:*

$$\mathbf{Y}(i_1, \ldots, i_d) = W_1(i_1)W_2(i_2)\cdots W_d(i_d), \tag{3.10}$$

with cores given for $\mu = 2 \ldots d-1$ by

$$W_1(i_1) = \begin{bmatrix} \delta U_1(i_1) & U_1(i_1) \end{bmatrix}, \; W_\mu(i_\mu) = \begin{bmatrix} V_\mu(i_\mu) & 0 \\ \delta V_\mu(i_\mu) & U_\mu(i_\mu) \end{bmatrix}, \; W_d(i_d) = \begin{bmatrix} V_d(i_d) \\ \delta V_d(i_d) \end{bmatrix}.$$

Proof. The identity can be easily verified by multiplying out the matrix product. The resulting cores $\mathbf{W}_\mu$ are of size $2r_{\mu-1} \times n_\mu \times 2r_\mu$ for $\mu = 2, \ldots, d-1$. ∎

This result allows us to conveniently handle elements of the tangent space in the same way as elements in the manifold, being able to directly reuse all implemented operations.

4. A Riemannian nonlinear conjugate gradient scheme. With the basic properties of elements in $\mathcal{M}_{\mathbf{r}}$ and $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ discussed, we can now describe all necessary steps of Algorithm 1 in detail. For each part, we will also mention its computational complexity. To simplify the presentation, we will state the operation count in terms of $r := \max_\mu r_\mu$ and $n := \max_\mu n_\mu$.

4.1. Evaluation of the cost function and sampling of TT/MPS tensors. In the tensor completion problem (1.1), the cost function is given by $f(\mathbf{X}) = \|\mathrm{P}_\Omega\mathbf{X} - \mathrm{P}_\Omega\mathbf{A}\|^2/2$. As the data samples $\mathrm{P}_\Omega\,\mathbf{A}$ are supplied as input, evaluating the cost function reduces to the application of the sampling operator P_Ω, see (1.2) for its definition, to the current iterate $\mathbf{X}$.

Sampling of TT/MPS tensors. We can compute the application of the sampling operator P_Ω to a TT/MPS tensor $\mathbf{X}$ by direct evaluation of the matrix product for each sampling point:

$$\mathbf{X}(i_1, i_2, \ldots, i_n) = U_1(i_1)U_2(i_2)\cdots U_d(i_d).$$

Thus, for each sampling point, we need to compute $d-1$ matrix-vector multiplications. Hence, evaluating the tensor at $|\Omega|$ sampling points involves $|\Omega|(d-1)r^2$ operations. By permuting the storage of the cores of size $r_\mu \times n_\mu \times r_{\mu+1}$ to arrays of size $r_\mu \times r_{\mu+1} \times n_\mu$ before starting the sampling procedure, the matrix blocks are properly, continuously aligned in memory for fast calls to the BLAS routine DGEMV.

Sampling of TT/MPS tangent tensors. Writing the tangent tensor in the form (3.10) we can reuse the sampling routine for TT/MPS tensors with an acceptable amount of extra cost — the rank is doubled, hence the number of operations is quadrupled.

4.2. Riemannian gradient. To obtain the descent direction in our optimization scheme, we need the Riemannian gradient of the objective function $f(\mathbf{X}) = \|\mathrm{P}_\Omega\mathbf{X} - \mathrm{P}_\Omega\mathbf{A}\|^2/2$.

PROPOSITION 4.1 ([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 $\nabla f_{\mathbf{X}}$ at point $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$. Then the Riemannian gradient of $f : \mathcal{M}_{\mathbf{r}} \to \mathbb{R}$ is given by* $\operatorname{grad} f(\mathbf{X}) = \mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}(\nabla f_{\mathbf{X}})$.

Hence, the Riemannian gradient is obtained by projecting the Euclidean gradient $\nabla f = \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}$ into the tangent space:

$$\operatorname{grad} f(\mathbf{X}) = \mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}(\mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}). \tag{4.1}$$

As the Euclidean gradient $\mathbf{Z} = \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a very large but sparse tensor, the projection into the tangent space has to be handled with care. When successively calculating the projection (3.8), the intermediate results are, in general, dense already after the first step. For example, the intermediate result $\mathbf{Z}^{<\mu>}\mathbf{X}_{\geq \mu+1}$ is a huge dense matrix of size $\mathbb{R}^{n_1 n_2 \cdots n_\mu \times r_\mu}$ whereas the resulting $\delta\mathbf{V}_\mu^{\mathsf{L}}$ is only of size $\mathbb{R}^{r_\mu n_\mu \times r_{\mu+1}}$. Hence, each entry of $\delta\mathbf{V}_\mu$ should be evaluated directly:

$$\begin{aligned}
\delta\mathbf{V}_\mu^{\mathsf{L}} &= (I_{r_\mu n_\mu} - \mathrm{P}_\mu)(I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1})^{\mathsf{T}} \mathbf{Z}^{<\mu>} \mathbf{X}_{\geq \mu+1}\,, \quad \text{for } \mu = 1, \ldots, d-1 \\
\delta\mathbf{V}_d^{\mathsf{L}} &= (I_{n_d} \otimes \mathbf{X}_{\leq d-1})^{\mathsf{T}} \mathbf{Z}^{<d>}.
\end{aligned}$$

The projection $(\mathbf{I}_{r_\mu n_\mu} - \mathrm{P}_\mu)$ is a straight-forward matrix multiplication using $\mathrm{P}_\mu = Q_\mu Q_\mu^{\mathsf{T}}$, where Q_μ is an orthogonal basis for the range of $\mathbf{U}_\mu^{\mathsf{L}}$ obtained from a QR decomposition and can be done separately at the end. Instead, we focus on $\mathbf{W}_\mu^{\mathsf{L}} := (I_\mu \otimes \mathbf{X}_{\leq \mu-1})^{\mathsf{T}} \mathbf{Z}^{<\mu>} \mathbf{X}_{\geq \mu+1}$. The i_μth slice of $\mathbf{W}_\mu$ is given by:

$$\begin{aligned}
W_\mu(i_\mu) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{\mu-1}=1}^{n_{\mu-1}} \sum_{i_{\mu+1}=1}^{n_{\mu+1}} \cdots \sum_{i_d=1}^{n_d} \Big[&\mathbf{X}_{\leq \mu-1}(i_1, \ldots, i_{\mu-1})^{\mathsf{T}} \mathbf{Z}(i_1, \ldots, i_d) \\
&\cdot \mathbf{X}_{\geq \mu+1}(i_{\mu+1}, \ldots, i_d) \Big].
\end{aligned}$$

Let $\Theta_{i_\mu} \subset \Omega$ be the set of sampling points $(j_1, \ldots, j_d)$ where the μth index j_μ coincides with i_μ:

$$\Theta_{i_\mu} = \{(j_1, \ldots j_d) \in \Omega \mid j_\mu = i_\mu\} \subset \Omega. \tag{4.2}$$

Then, the multiple sum reduces to one sum over Θ_{i_μ}, as the other entries do not contribute:

$$W(i_\mu) = \sum_{(j_1, \ldots j_d) \in \Theta_{i_\mu}} \mathbf{X}_{\leq \mu-1}(j_1, \ldots, j_{\mu-1})^{\mathsf{T}} \mathbf{Z}(j_1, \ldots, j_d) \mathbf{X}_{\geq \mu+1}(j_{\mu+1}, \ldots, j_d),$$

which can be reformulated into

$$W(i_\mu) = \sum_{(j_1, \ldots j_d) \in \Theta_{i_\mu}} \mathbf{Z}(j_1, \ldots, j_d) \big[U_1(j_1) \cdots U_{\mu-1}(j_{\mu-1})\big]^{\mathsf{T}} \big[U_{\mu+1}(j_{\mu+1}) \cdots U_d(j_d)\big].$$

The algorithmic implementation for all first order variations $\delta\mathbf{V}_\mu$ for $\mu = 1, \ldots d$ is shown in Algorithm 2, with reuse of intermediate matrix products.

4.3. Retraction. Taking one step from the current iterate $\mathbf{X}_k \in \mathcal{M}_{\mathbf{r}}$ in the search direction $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ yields a new point in the tangent space which is, in general, not an element of $\mathcal{M}_{\mathbf{r}}$ anymore. In Riemannian geometry, the well-known exponential map provides a local diffeomorphism between the neighborhood of $\mathbf{X}_k$ and the neighborhood around the origin of $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$. Assuming not-too-large step sizes, the exponential map thus allows us to map the new point $\mathbf{X}_k + \xi$ back onto the manifold. While this map is given in a simple form for some manifolds, such as the sphere, it

Algorithm 2 Computing the Riemannian gradient

Input: Left-orth. cores $\mathbf{U}_\mu$ and right-orth. cores $\mathbf{V}_\mu$ of $\mathbf{X} \in \mathcal{M}_\mathbf{r}$, Euclid. gradient $\mathbf{Z}$
Output: First order variations $\delta\mathbf{V}_\mu$ for $\mu = 1, \dots, d$.

```
{% Initialize cores}
for μ = 1, ..., d do
    δV_μ ← zeros(r_μ, n_μ, r_{μ+1})
end for

for (j_1, j_2, ..., j_d) ∈ Ω do

    {% Precompute left matrix product}
    U_L{1} ← U_1(j_1)
    for μ = 2, ..., d − 1 do
        U_L{μ} ← U_L{μ − 1} U_2(j_2)
    end for

    V_R ← V_d(j_d)

    {% Calculate the cores beginning from the right}
    δV_d(j_d) ← δV_d(j_d) + Z(j_1, j_2, ..., j_d) · U_L{d − 1}^T
    for μ = d − 1, ..., 2 do
        δV_μ(j_μ) ← δV_μ(j_μ) + Z(j_1, j_2, ..., j_d) · U_L{μ − 1}^T V_R^T
        V_R ← V_μ(j_μ) V_R^T
    end for
    δU_1(j_1) ← δU_1(j_1) + Z(j_1, j_2, ..., j_d) · U_R^T
end for
{% Project into orth. complement of the range of U_μ^L}
for μ = 1, ..., d − 1 do
    [Q_μ, R_μ] = qr(U_μ^L)
    δV_μ^L ← δV_μ^L − Q_μ(Q_μ^T) δV_μ^L
end for
```

is computationally infeasible for our manifold $\mathcal{M}_\mathbf{r}$. However, as proposed in [1], a first order approximation of this map, a so-called *retraction*, is sufficient to ensure convergence of gradient-based optimization schemes.

For matrices, multiple variants of retractions into the low-rank manifold have been discussed in a recent survey paper by Absil and Oseledets [2].

For $\mathcal{M}_\mathbf{r}$, a retraction maps the new point to a rank-$\mathbf{r}$ tensor. A computationally efficient method is given by the TT-SVD procedure which satisfies the quasi-best approximation property (3.4). This is very similar to the Tucker case discussed in [22], where it was shown that the HOSVD (which posesses an analogous quasi-best approximation property) fulfills indeed all necessary properties of a retraction map. Due to the strong similarity, we refer to this paper for more details. The required smoothness of the TT-SVD follows directly, as it basically only involves unfoldings and sequential applications of smooth SVDs, moving from one core to the next.

To efficiently apply the TT-SVD procedure, we exploit the representation (3.9) and notice that the new point, obtained by a step $\mathbf{Y} \in T_\mathbf{X}\mathcal{M}_\mathbf{r}$ of length α from the

point $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$, can be written as

$$\mathbf{X}(i_1, \dots, i_d) + \alpha \mathbf{Y}(i_1, \dots, i_d) = \begin{bmatrix} \alpha \mathbf{V}_1(i_1) & \mathbf{U}_1(i_1) \end{bmatrix} \begin{bmatrix} \mathbf{V}_2(i_2) & 0 \\ \alpha\delta\mathbf{V}_2(i_2) & \mathbf{U}_2(i_2) \end{bmatrix} \cdots \\ \cdots \begin{bmatrix} \mathbf{V}_{d-1}(i_{d-1}) & 0 \\ \alpha\delta\mathbf{V}_{d-1}(i_{d-1}) & \mathbf{U}_{d-1}(i_{d-1}) \end{bmatrix} \begin{bmatrix} \mathbf{V}_d(i_d) \\ \mathbf{U}_d(i_d) + \alpha\delta\mathbf{V}_d(i_d) \end{bmatrix}.$$

Hence, retracting back to the rank-$\mathbf{r}$ manifold $\mathcal{M}$ is equivalent to truncating a rank-$2\mathbf{r}$ TT/MPS tensor to rank $\mathbf{r}$, for a total cost of approximately $O(dnr^3)$. We refer to [30, Section 3] for a detailed description on the TT-SVD procedure and its computational complexity. Furthermore, the orthogonality relations between $\mathbf{U}_\mu$ and $\delta\mathbf{V}_\mu$ can be used to save some work when reorthogonalizing.

4.4. Search direction update and vector transport. In the nonlinear conjugate gradient scheme, the new search direction η_k is computed as a linear combination of the current gradient ξ_k and the previous direction η_{k-1} scaled by a factor β_k,

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

where $\mathcal{T}_{\mathbf{X}_{k-1} \to \mathbf{X}_k}$ is the *vector transport* which maps elements from $T_{\mathbf{X}_{k-1}}\mathcal{M}_{\mathbf{r}}$ to $T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}$, see [1] for a formal definition. The orthogonal projection into the tangent space, $\mathrm{P}_{T_{\mathbf{X}+1}\mathcal{M}_{\mathbf{r}}}$, yields a vector transport, as $\mathcal{M}_{\mathbf{r}}$ is an embedded submanifold of $\mathbb{R}^{n_1 \times \cdots \times n_d}$, see [1, Sec. 8.1.3].

To compute the projection of a certain tangent tensor $\mathbf{Y}$ into $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$, we make use of the identity (3.10), reducing it to the projection of a TT/MPS tensor of rank $2\mathbf{r}$ into $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$. Let $\mathbf{Y}$ thus be represented as

$$\mathbf{Y}(i_1, \dots, i_d) = W_1(i_1) W_2(i_2) \cdots W_d(i_d).$$

Then, we have to compute the first order variations

$$\begin{aligned} \delta\mathbf{V}_\mu^{\mathsf{L}} &= \left(I_{n_\mu r_{\mu-1}} - \mathrm{P}_\mu\right) \left(I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}\right)^{\mathsf{T}} \mathbf{Y}^{\langle \mu \rangle} \mathbf{X}_{\geq \mu+1} \\ &= \left(I_{n_\mu r_{\mu-1}} - \mathrm{P}_\mu\right) \left(I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}\right)^{\mathsf{T}} \mathbf{Y}_{\leq \mu} \mathbf{Y}_{\geq \mu+1}^{\mathsf{T}} \mathbf{X}_{\geq \mu+1}, \end{aligned}$$

which can be efficiently evaluated as a tensor contraction along dimensions $1, \dots d$ except mode μ. This can be checked using the recursive relations (3.2),

$$\begin{aligned} \mathbf{Y}_{\geq \mu+1}^{\mathsf{T}} \mathbf{X}_{\geq \mu+1} &= \mathbf{V}_{\mu+1}^{\mathsf{R}} (\mathbf{Y}_{\geq \mu}^{\mathsf{T}} \otimes I_{n_{\mu+1}}) (\mathbf{X}_{\geq \mu} \otimes I_{n_{\mu+1}}) (\mathbf{U}_{\mu+1}^{\mathsf{R}})^{\mathsf{T}} \\ &= \mathbf{V}_{\mu+1}^{\mathsf{R}} (\mathbf{Y}_{\geq \mu}^{\mathsf{T}} \mathbf{X}_{\geq \mu} \otimes I_{n_{\mu+1}}) (\mathbf{U}_{\mu+1}^{\mathsf{R}})^{\mathsf{T}} \end{aligned}$$

and $\mathbf{Y}_{\geq d}^{\mathsf{T}} \mathbf{X}_{\geq d} = \mathbf{V}_d^{\mathsf{R}} (\mathbf{U}_d^{\mathsf{R}})^{\mathsf{T}}$. An analogous relation holds for $\mathbf{X}_{\leq \mu-1}^{\mathsf{T}} \mathbf{Y}_{\leq \mu-1}$.

As these quantities are shared between all $\delta\mathbf{V}_\mu$, we first precompute $\mathbf{Y}_{\geq \mu}^{\mathsf{T}} \mathbf{X}_{\geq \mu}$ for all $\mu = 2, \dots, d$ and $\mathbf{X}_{\leq \mu}^{\mathsf{T}} \mathbf{Y}_{\leq \mu}$ for $\mu = 1, \dots, d-1$, using $O(dr^3 n)$ operations. Combining this with the orthogonal projection $\left(I_{n_\mu r_{\mu-1}} - \mathrm{P}_\mu\right)$ yields a total amount of $O(dr^3 n)$ flops.

Many options exist for the choice of β_k within the framework of nonlinear conjugate gradient. Here, we choose the Fletcher-Reeves update,

$$\beta_k = \frac{\|\xi_k\|}{\|\xi_{k-1}\|},$$

as it needs only about $2dnr^2$ operations. This follows from representation (3.9), as there, the inner product of two tangent tensors in the same tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ reduces to the inner product of the $\delta\mathbf{V}_\mu$ cores only.

In our experiments, we observed that choosing a different scheme, such as Polak-Ribière+, has almost no influence on the convergence of Algorithm 1.

4.5. Line search procedure. Determining a good step size α in Algorithm 1 is crucial to ensure fast convergence of the method. The optimal choice would be the minimizer of the objective function along the geodesic determined by the current search direction η_k: $\alpha_k = \operatorname{argmin}_\alpha f(R(\mathbf{X}_k + \alpha\eta_k))$. As described in [42, 22], this nonlinear optimization problem can be linearized and then solved explicitly by dropping the retraction R and thereby only minimizing within the tangent space,

$$\alpha_k \approx \operatorname*{argmin}_\alpha f(\mathbf{X}_k + \alpha\eta_k) = \frac{\langle \mathrm{P}_\Omega\, \eta_k, \mathrm{P}_\Omega(\mathbf{A} - \mathbf{X}_k)\rangle}{\langle \mathrm{P}_\Omega\, \eta_k, \mathrm{P}_\Omega\, \eta_k\rangle}.$$

This estimate can be calculated in $O(|\Omega|(d-1)r^2)$ operations. The accuracy of this approximation depends on the curvature of the manifold at the current iterate, but in all our experiments, we have never observed a situation where this estimate was not good enough. To make sure that the iterates fulfill the Wolfe conditions, a simple Armijo backtracking scheme can be added, see e.g. [29].

4.6. Stopping criteria. During the course of the optimization procedure, we monitor the current relative error on the sampling set,

$$\varepsilon_\Omega(\mathbf{X}_k) = \frac{\|\,\mathrm{P}_\Omega\,\mathbf{A} - \mathrm{P}_\Omega\,\mathbf{X}_k\|}{\|\,\mathrm{P}_\Omega\,\mathbf{A}\|}.$$

This requires almost no extra computations, as the Euclidean gradient $\nabla f = \mathrm{P}_\Omega\,\mathbf{A} - \mathrm{P}_\Omega\,\mathbf{X}_k$ is already calculated in each step.

When the sampling rate is very low or the estimated rank $\mathbf{r}$ is chosen larger than the true rank of the underlying data, one can often observe *overfitting*. In these cases, the residual error does converge to zero as expected, but the obtained reconstruction is a bad approximation of the original data. This is to be expected as the degrees of freedom in the model exceed the information available. While this intrinsic problem cannot be completely avoided, it is helpful to be able to detect these cases of overfitting. Then, we can act accordingly and reduce the degrees of freedom of the model (by choosing a lower estimated rank $\mathbf{r}$) or by increasing the number of sampling points, if possible. Hence, we also measure the relative error on a test set Γ, $\Gamma \cap \Omega = \emptyset$:

$$\varepsilon_\Gamma(\mathbf{X}_k) = \frac{\|\,\mathrm{P}_\Gamma\,\mathbf{A} - \mathrm{P}_\Gamma\,\mathbf{X}_k\|}{\|\,\mathrm{P}_\Gamma\,\mathbf{A}\|}.$$

This test set is either given or can be obtained by random subsampling of the given data. Numerical experiments show that it usually suffices to choose only few test samples, say, $|\Gamma| = 100$. Algorithm 1 is run until either the prescribed tolerance is attained for $\varepsilon_\Omega(\mathbf{X}_k)$ and $\varepsilon_\Gamma(\mathbf{X}_k)$ or the maximum number of iterations is reached. Furthermore, to detect stagnation in the optimization procedure, we check if the relative change in ε_Ω and ε_Γ between two iterates is below a certain threshold δ:

$$\frac{|\varepsilon_\Omega(\mathbf{X}_k) - \varepsilon_\Omega(\mathbf{X}_{k+1})|}{|\varepsilon_\Omega(\mathbf{X}_k)|} < \delta, \quad \frac{|\varepsilon_\Gamma(\mathbf{X}_k) - \varepsilon_\Gamma(\mathbf{X}_{k+1})|}{|\varepsilon_\Gamma(\mathbf{X}_k)|} < \delta. \tag{4.3}$$

In practice, we choose $\delta \approx 10^{-4}$, but adjustments can be necessary depending on the underlying data.

4.7. Scaling of the algorithm: Computational complexity. If we sum up the computational complexity needed for one iteration of Algorithm 1, we arrive at a total cost of

$$O(d(n + |\Omega|)r^3) \tag{4.4}$$

operations. As almost all tensor operations for elements in $\mathcal{M}_\mathbf{r}$ have this asymptotic complexity, this result is expected. While suggesting that most parts of the algorithm are equally expensive, the majority of the wall-time is spent on the calculation of the gradient, see Section 4.2, as it involves $|\Omega|$ operations on single indices. Inevitably, this leads to unaligned memory accesses which are a bottleneck on today's computers. This can be partly avoided by embarrassingly parallel distribution of the loop over all indices in the sampling set Ω, but is out of the scope of this article.

4.8. Convergence of the algorithm. The convergence analysis of Algorithm 1 is a direct consequence of the analysis employed in the matrix [42] and the Tucker tensor case [22].

PROPOSITION 4.2 (c.f. [1, Theorem 4.3.1]). *Let $(\mathbf{X}_k)_{k\in\mathbb{N}}$ be an infinite sequence of iterates generated by Algorithm 1. Then, every accumulation point $\mathbf{X}_*$ of $(\mathbf{X}_k)$ is a critical point of the cost function f and hence satisfies $\mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathbf{X}_*) = \mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathbf{A})$.*

This result shows us that in the tangent space, reconstruction is achieved for all limit points of Algorithm 1. Unfortunately, $\mathbf{X}_*$ is not necessarily an element of $\mathcal{M}_\mathbf{r}$ anymore — and hence not a valid solution of problem (1.1). To enforce this, we regularize the cost function in such a way that the iterates $\mathbf{X}_k$ stay inside a compact subset of $\mathcal{M}_\mathbf{r}$. Analogous to [22], we define the modified cost function g as

$$g : \mathcal{M}_\mathbf{r} \to \mathbb{R}, \quad \mathbf{X} \mapsto f(\mathbf{X}) + \lambda^2 \sum_{\mu=1}^{d} \left(\|\mathbf{X}^{<\mu>}\|_F^2 + \|(\mathbf{X}^{<\mu>})^\dagger\|_F^2 \right), \quad \lambda > 0, \tag{4.5}$$

where $(\mathbf{X}^{<\mu>})^\dagger$ denotes the pseudo-inverse of $\mathbf{X}^{<\mu>}$ and λ is the regularization parameter. With this modification regularizing the singular values of the matricizations of $\mathbf{X}$, we obtain the following result:

PROPOSITION 4.3. *Let $(\mathbf{X}_k)_{k\in\mathbb{N}}$ be an infinite sequence of iterates generated by Algorithm 1 but with the modified cost function g defined in* (4.5)*. Then*

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

The proof of Proposition 4.3 is completely analogous to the proof in [22, Prop. 3.2] but with the modified cost function (4.5), and is omitted here.

Note that the regularization parameter λ can be chosen arbitrarily small. If the core unfoldings are always of full rank during the optimization procedure even when $\lambda \to 0$, then the accumulation points $\mathbf{X}_*$ are guaranteed to stay inside $\mathcal{M}_\mathbf{r}$ and $\operatorname{grad} f(\mathbf{X}_*) = \mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathbf{X}_* - \mathrm{P}_\Omega \mathbf{A}) \to 0$ as $\lambda \to 0$. In this case, optimizing the modified cost function (4.5) is equivalent to the original cost function.

4.9. Adaptive rank adjustment. There is an obvious disadvantage in the way the tensor completion problem (1.1) is defined: The rank of the underlying problem has to be known *a priori*. For high-dimensional TT/MPS tensors, this problem is much more severe than in the matrix case, as we not only have one unknown rank, but a whole rank tuple $\mathbf{r} = (r_0, r_1, \ldots, r_d)$. For real-world data, this information is

usually not available and the ranks can be very different from each other, see e.g. [30, Table 3.2] for a common distribution of the ranks.

Instead, we can require that the solution should have rank smaller than a certain prescribed maximal rank. This maximal rank $\mathbf{r}_{\max}$ is determined by the computational resources available and not necessarily by the underlying data. As the rank has a strong influence on the complexity of the algorithm, see Section 4.7, this motivates the following rank-adaptive procedure shown in Algorithm 3. First, we run Algorithm 1 with $\mathbf{r} = (1, 1, \dots, 1)$. Then, we use the obtained result as a starting guess for another run of Algorithm 1, but with $\mathbf{r} = (1, 2, 1, \dots, 1)$. Then, the rank r_2 between the second and third core is increased, and so on and so forth. This procedure is repeated until either the prescribed residual tolerance is fulfilled or $\mathbf{r} = \mathbf{r}_{\max}$ is reached.

To reuse the previous solution as a warmstart for the higher rank, we augment the μth and $(\mu+1)$th core by zeros:

$$\mathbf{U}_\mu^{\mathsf{L}} \leftarrow \begin{bmatrix} \mathbf{U}_\mu^{\mathsf{L}} & 0 \end{bmatrix}, \qquad \mathbf{U}_{\mu+1}^{\mathsf{R}} \leftarrow \begin{bmatrix} \mathbf{U}_{\mu+1}^{\mathsf{R}} & 0 \end{bmatrix}. \tag{4.6}$$

As a result, the augmented cores $\mathbf{U}_\mu$ and $\mathbf{U}_{\mu+1}$ are rank-deficient. Due to our special choice of orthogonalization of tangent tensors (3.9), the projection (3.8) does not include any inverses and therefore, this rank-deficiency does not pose any problems.

As the main goal of the first steps is to steer the iterates into the right direction, only few steps of Algorithm 1 are required at each level. We make use of the stagnation criterion (4.3) with a crude tolerance $\delta = 0.01$. With this choice, usually less than 10 steps of Algorithm 1 are performed at each level.

In the worst-cast scenario, this leads to $(d-1)r_{\max}$ executions of Algorithm 1. To limit the occurances of unnecessary rank increases, we reset and lock the rank in the current mode if the increase did not sufficiently improve on ε_Γ.

Algorithm 3 Adaptive rank adjustment

Input: Sampled data $\mathrm{P}_\Omega \mathbf{A}$, maximal rank $r_{\max}$
Output: Completed tensor $\mathbf{X}$ with $\mathrm{rank}_{\mathrm{TT}}(\mathbf{X}) < r_{\max}$

```
X random tensor, r := rank_TT(X) = (1, 1, ..., 1)^T
X ← Result of Algorithm 1 using X as starting guess.
locked = zeros(1, d − 1)

for k = 2, ..., r_max do
  for μ = 1, ..., d − 1 do
    if locked(μ) then
      Rank μ is locked. Skip.
    else
      X_new ← Increase μth rank of X to (1, r_1, ..., r_μ + 1, ... r_{d−1}, 1) using (4.6)
      X_new ← Result of Algorithm 1 using X_new as starting guess.

      if |ε_Γ(X) − ε_Γ(X_new)|/|ε_Γ(X)| < 10^{−4} then
        locked(μ) = 1          % do not increase rank any further and revert step
      else
        X ← X_new              % accept step
      end if
    end if
  end for
end for
```

5. Numerical experiments. In the following section we investigate the performance of the proposed tensor completion algorithm and compare it to existing methods, namely HTOpt [9, 10], ADF [15] and a simple alternating optimization scheme, see Section 5.2.

As mentioned in the introduction, cross approximation techniques can be used if the sampling set Ω is not prescribed, but can be choosen freely. This is often the case when dealing with the approximation of high-dimensional functions and solutions of parameter-dependent PDEs. To show the effectiveness of tensor completion in these applications, we compare Algorithm 1 to state-of-the-art cross-approximation algorithms based on the TT/MPS [12] and HT format [3].

We have implemented Algorithm 1 in MATLAB based on the TTeMPS toolbox, see `http://anchp.epfl.ch/TTeMPS`. As shown in Section 4, the calculation of the cost function, the gradient and the linearized linesearch involve operations on sparse tensors. To obtain an efficient algorithm we have implemented these crucial steps in C using the MEX-function capabilities of MATLAB with direct calls to BLAS routines wherever possible.

All timings have been conducted on an Intel Xeon E31225, 3.10GHz quad-core processor with 8 GB of memory, running MATLAB version 2012b.

5.1. Scaling of the algorithm. As a first experiment, we check the computational complexity of Algorithm 1 derived in Section 4.7. In Figure 5.1, we show three plots corresponding to the scaling with respect to (left) the tensor size n, (middle) the number of dimensions d and (right) the tensor rank r. In all cases, the original data is a TT/MPS tensor of known rank r whose cores consist of entries uniformly, randomly chosen from $[0, 1]$. In the first case (left), we have chosen $d = 10$, $\mathbf{r} = (1, 10, \ldots, 10, 1)$, and mode size $n \in \{20, 40, \ldots, 200\}$. In (middle), we have set the mode size to $n = 100$, $\mathbf{r} = (1, 10, \ldots, 10, 1)$, and dimensions ranging from $d = 3$ to $d = 20$. Finally, for (right) we set $d = 10$, $n = 100$ and $\mathbf{r} = (1, r, \ldots, r, 1)$ with $r \in \{2, \ldots, 25\}$. For each configuration, we run 10 steps of Algorithm 1 with 5 repetitions. The red lines are linear fits for n and d and a cubic fit for tensor rank r. We observe that Algorithm 1 indeed scales linearly with n and d and cubically in r in accordance with (4.4).

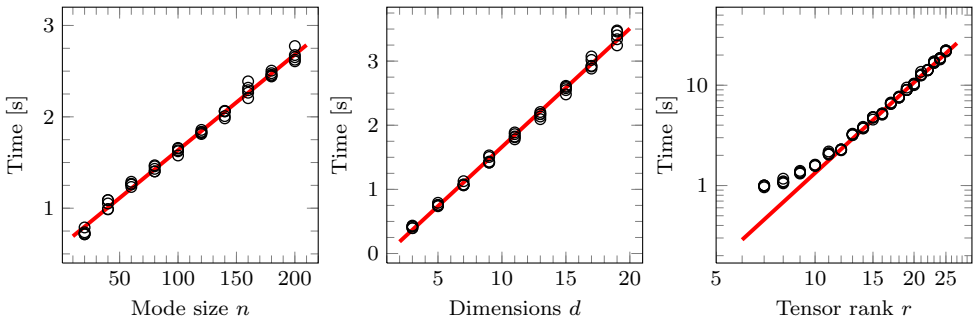


Fig. 5.1: *Scaling of Algorithm 1 with respect to tensor size n, number of dimensions d and tensor rank r. In the rightmost figure, in log-log scale, the red line corresponds to a cubic dependence on the tensor rank. See Section 5.1 for details.*

5.2. Comparison to an alternating linear scheme. To evaluate the performance of our proposed algorithm, we have implemented a simple *alternating linear*

scheme (ALS) algorithm to solve the tensor completion problem (1.1). This approach was suggested by Grasedyck, Kluge and Krämer [16]. In the TT/MPS format, algorithms based on ALS-type optimization are a simple but surprisingly effective way to solve an optimization scheme. In each step, all cores but one are kept fixed and the optimization problem is reduced to a small optimization problem of a single core. The core is replaced by its locally optimal solution and fixed, while the optimization scheme moves to the neighboring core. This approach has been successfully used to solve linear systems and eigenvalue problems in the TT/MPS format, see e.g. [11, 18, 34].

For the tensor completion problem (1.1), one single core optimization step of the ALS scheme is given by

$$\begin{aligned}\widetilde{\mathbf{U}}_\mu &= \operatorname*{argmin}_{\mathbf{V}} \frac{1}{2} \sum_{\mathbf{i}\in\Omega} \big[\mathbf{A}(i_1,\dots,i_d) - U_1(i_1)\cdots U_{\mu-1}(i_{\mu-1})V(i_\mu)U_{\mu+1}(i_{\mu+1})\cdots U_d(i_d)\big]^2 \\ &= \operatorname*{argmin}_{\mathbf{V}} \frac{1}{2} \sum_{\mathbf{i}\in\Omega} \big[\mathbf{A}(i_1,\dots,i_d) - \mathbf{X}_{\le\mu-1}(i_1,\dots i_{\mu-1})V(i_\mu)\mathbf{X}_{\ge\mu+1}(i_{\mu+1},\dots,i_d)\big]^2.\end{aligned}$$

Cycling through all cores U_μ for $\mu = 1,\dots,d$ multiple times until convergence yields the full ALS completion scheme. To solve one core optimization step efficiently, we look at the i_μth slice of the new core $\widetilde{\mathbf{U}}_\mu$, making use of notation (4.2):

$$\begin{aligned}\widetilde{\mathbf{U}}_\mu(i_\mu) = \operatorname*{argmin}_{\mathbf{V}} \frac{1}{2} \sum_{\mathbf{j}\in\Theta_{i_\mu}} &\big[\mathbf{A}(j_1,\dots,j_d) \\ &- \mathbf{X}_{\le\mu-1}(j_1,\dots j_{\mu-1})V(j_\mu)\mathbf{X}_{\ge\mu+1}(i_{\mu+1},\dots,i_d)\big]^2 \\ = \operatorname*{argmin}_{\mathbf{V}} \frac{1}{2} \sum_{\mathbf{j}\in\Theta_{j_\mu}} &\Big[\mathbf{A}(j_1,\dots,j_d) \\ &- \big(\mathbf{X}_{\ge\mu+1}(j_{\mu+1},\dots,j_d)^{\mathsf{T}} \otimes \mathbf{X}_{\le\mu-1}(j_1,\dots j_{\mu-1})\big)\operatorname{vec}\left(V(j_\mu)\right)\Big]^2\end{aligned}$$

which is a linear least squares problem for $\operatorname{vec}(V(i_\mu))$ with a system matrix of size $|\Theta_{i_\mu}| \times r_\mu r_{\mu+1}$. For a uniform random sampling, we have that $n_\mu|\Theta_{i_\mu}| \approx |\Omega|$. Hence, solving the n_μ least squares problems to obtain the new core $\widetilde{\mathbf{U}}_\mu$ can be performed in $O(|\Omega|r^4)$ operations. Updating each core $\mu = 1,\dots,d$ once (one so-called half-sweep of the ALS procedure) then results in a total cost of $O(d|\Omega|r^4)$ operations. Compared to the complexity of Algorithm 1, this ALS procedure involves the fourth power of the rank instead of the third and is therefore less competitive as the rank of the underlying data increases. To obtain comparable performance to Algorithm 1, the setup of the least squares system matrices is performed in optimized MEX-functions, written in C.

5.3. Reconstruction of random tensors. In Figure 5.2 a phase plot is depicted, showing the ability of Algorithm 1 and the ALS procedure to recover a tensor as a function of the number of known values of the original tensor. We run both algorithms 5 times for each combination of tensor size n and sampling set size $|\Omega|$. The brightness represents how many times the iteration converged, where white means 5 out of 5 converged and black corresponds to no convergence. For cases directly at the phase transition, where convergence may be extremely slow, we measure the the convergence speed

$$\rho = \left(\frac{\|\operatorname{P}_\Gamma \mathbf{A} - \operatorname{P}_\Gamma \mathbf{X}_{\text{end}}\|}{\|\operatorname{P}_\Gamma \mathbf{A} - \operatorname{P}_\Gamma \mathbf{X}_{\text{end-5}}\|}\right)^{\frac{1}{5}},$$

determining the progress during the last 5 iterations of the algorithm. If after 250 iterations $\rho < 0.95$, then we consider the iteration convergent. Note that in almost all convergent cases, the algorithms have reached the prescribed accuracy within less than 100 steps.

The original data $\mathbf{A}$ is constructed as a TT/MPS tensor of dimension $d = 5$ and rank $\mathbf{r} = (1, 3, \ldots, 3, 1)$ with the entries of each core being uniform random samples from $[0, 1]$. The question of how many samples are required to recover the underlying data has been a very active topic in the last years. For the matrix case ($d = 2$) it has been shown that around $O(nr \log(n))$ samples are necessary [8, 20], where n is the matrix size and r the rank of the underlying data. Up to this date, these results could not be extended to the tensor case, see e.g. [33] for a discussion. Some approaches, such as [28] were able to prove that $O(n^{\frac{d}{2}})$ samples suffice, but numerical experiments, such as in [21] for $d = 3$, clearly suggest a much better scaling behaviour similar to the matrix case. We have included a white dashed curve corresponding to $20n \log(n)$ in Figure 5.2.

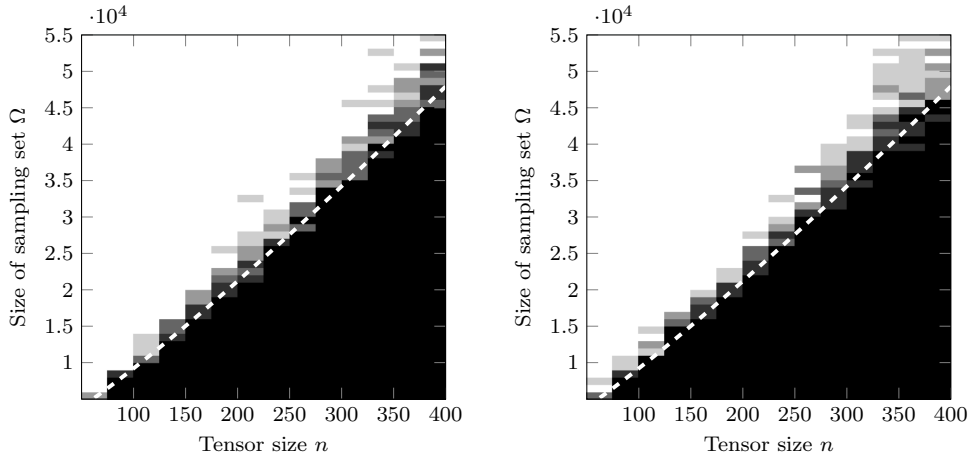


Fig. 5.2: *Phase plots for* ***(left)*** *Algorithm 1 and* ***(right)*** *ALS, showing the ability to recover the original data as a function of the mode size n and the size of the sampling set Ω. The underlying tensor is a random TT/MPS tensor of known rank $\mathbf{r} = (1, 3, \ldots, 3, 1$. White means convergent in all 5 runs, black means convergent in no runs. The white dashed curve corresponds to $20n \log(n)$.*

5.4. Interpolation of high-dimensional functions. In the next section, we will investigate the application of tensor completion to discretizations of high dimensional functions. The random tensors considered in the previous Section 5.3 are constructed to be of a certain well-defined rank. For discretizations of function data, we can, in general, only hope for a sufficiently fast decay of the singular values in each matricization, resulting in good approximability by a low-rank tensor. As the rank is not known *a priori*, we will employ the rank-increasing strategy presented in Section 4.9, Algorithm 3, for all following experiments, both for Algorithm 1 and the ALS.

5.4.1. Comparison to ADF. We compare the accuracy and timings of Algorithm 1 and the ALS procedure to an *alternating direction fitting* by Grasedyck *et al.* [15] for different values of the maximum rank. We reconstruct the setup in [15, Section

	ADF		Algorithm 1		ALS	
r	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time
2	$1.94 \cdot 10^{-2}$	14 s	$8.39 \cdot 10^{-3}$	6.40 s	$1.77 \cdot 10^{-2}$	5.73 s
3	$2.05 \cdot 10^{-3}$	1.3 min	$7.87 \cdot 10^{-3}$	21.8 s	$6.97 \cdot 10^{-3}$	19.2 s
4	$1.72 \cdot 10^{-3}$	24 min	$2.14 \cdot 10^{-3}$	43.7 s	$6.24 \cdot 10^{-3}$	46.8 s
5	$1.49 \cdot 10^{-3}$	39 min	$8.09 \cdot 10^{-4}$	1.3 min	$1.67 \cdot 10^{-3}$	1.6 min
6	$2.25 \cdot 10^{-3}$	1.7 h	$1.09 \cdot 10^{-3}$	2.3 min	$3.46 \cdot 10^{-3}$	3.4 min
7	$8.53 \cdot 10^{-4}$	2.6 h	$3.60 \cdot 10^{-4}$	3.5 min	$1.20 \cdot 10^{-3}$	6.3 min

Table 5.1: *Comparison of the reconstruction accuracy of Algorithm 1 to an ADF completion algorithm [15] for different values of the maximal rank. The underlying data is a discretized function resulting in a tensor of dimension $d = 8$ and mode size $n = 20$, see Section 5.4.1. The results and timings for ADF are taken from [15].*

4.2] for Algorithm 1 and the ALS scheme to compare against the therein stated results. The original data tensor of dimension $d = 8$ and mode sizes $n = 20$ is constructed by

$$\mathbf{A}(i_1, i_2, \dots, i_d) = \frac{1}{\sqrt{\sum_{\mu=1}^{n} i_\mu^2}},$$

The results are shown in Table 5.1 for different values of the maximal rank $r := \max_\mu r_\mu$. Both sampling set Ω and test set Γ are chosen to be of size $|\Omega| = |\Gamma| = 10dnr^2$.

The results are shown in Table 5.1 for different values of the maximal rank $r := \max_\mu r_\mu$. We can see that the reconstruction quality is similar for all algorithms, but the ADF scheme suffers from exceedingly long computation times. A part of these timing differences can be attributed to the fact that it does not use optimized MEX routines for the sampling. For the ALS procedure, the quartic instead of cubic scaling of the complexity with respect to the rank r becomes noticable.

5.4.2. Comparison to HTOpt. We now compare the reconstruction performance to a Riemannian optimization scheme on manifold of tensors in the Hierarchical Tucker format, *HTOpt* [9, 10].

This algorithm is conceptually very similar to Algorithm 1 and the Riemannian tensor completion algorithm in the Tucker format *GeomCG* [22], but uses a different tensor format and a Gauss-Newton instead of a conjugate gradient scheme. We use the most recent HTOpt version 1.1 which is available from the authors. This implementation is optimized for very high sampling rates, where up to 50% of the original data is given. Motivated by this setting, the involved matrices and tensors cannot be considered sparse anymore and are treated by dense linear algebra. On the other hand, this approach limits the applicability of the algorithm to relatively low-dimensional tensors, say $d = 4, 5$. To make a fair comparison, we have adapted the included example `synthetic.m` with the parameters left at their default value.

As this code works with a different tensor format, prescribed tensor ranks are in general not directly comparable. To circumvent this, we have chosen $d = 4$, as in this case, the TT-rank $\mathbf{r} = (1, r_1, r_2, r_3, 1)$ directly corresponds to a HT dimension tree with internal rank r_2 and leaf ranks r_1 and r_3. We discretize the function

$$f : [0, 1]^4 \to \mathbb{R}, \quad f(\mathbf{x}) = \exp(-\|x\|)$$

using $n = 20$ equally spaced discretization points on $[0, 1]$ in each mode. The maximum rank is set to $\mathbf{r} = (1, 5, 5, 5, 1)$.

$\lvert\Omega\rvert/n^d$	HTOpt $\varepsilon_\Gamma(\mathbf{X}_{\text{end}})$	time	ADF $\varepsilon_\Gamma(\mathbf{X}_{\text{end}})$	time	Algorithm 1 $\varepsilon_\Gamma(\mathbf{X}_{\text{end}})$	time	ALS $\varepsilon_\Gamma(\mathbf{X}_{\text{end}})$	time
0.001	$1.15 \cdot 10^{0}$	28.4 s	$6.74 \cdot 10^{-2}$	53.0 s	$8.95 \cdot 10^{-2}$	4.33 s	—	—
0.005	$1.67 \cdot 10^{0}$	28.8 s	$6.07 \cdot 10^{-3}$	3.2 m	$9.70 \cdot 10^{-3}$	6.19 s	$4.99 \cdot 10^{-1}$	5.67 s
0.01	$1.99 \cdot 10^{0}$	29.7 s	$3.08 \cdot 10^{-2}$	6.4 m	$4.40 \cdot 10^{-3}$	6.81 s	$6.61 \cdot 10^{-3}$	6.50 s
0.05	$3.68 \cdot 10^{-3}$	29.6 s	$1.95 \cdot 10^{-5}$	18 m	$2.02 \cdot 10^{-5}$	7.09 s	$2.00 \cdot 10^{-5}$	9.64 s
0.1	$1.99 \cdot 10^{-3}$	29.3 s	$6.50 \cdot 10^{-5}$	27 m	$4.18 \cdot 10^{-5}$	8.08 s	$4.07 \cdot 10^{-5}$	13.3 s

Table 5.2: *Comparison of the reconstruction accuracy of Algorithm 1, ALS and ADF to a Riemannian optimization scheme using the Hierarchical Tucker format (HTOpt) for different sizes of the sampling set. The underlying data is a discretized function resulting in a tensor of dimension $d = 4$ and mode size $n = 20$, see Section 5.4.2.*

In Table 5.2 we present the results for different sizes of the sampling set $|\Omega|$ and the corresponding relative error on the test set Γ, $|\Gamma| = 100$. Sampling and test sets are chosen identically for all algorithms. Since this example is not covered in [15], we now use the reference implementation of ADF provided by the authors. We observe that the proposed Algorithm 1 is the fastest and yields better reconstruction than HTOpt. For sampling ratios 0.001, 0.005, 0.01, HTOpt fails to recover the original data within the specified number of maximum iterations, 250. For the smallest sampling ratio 0.001, the ALS procedure does not have enough samples available to solve the least squares problems and hence does not converge. Altogether, Algorithm 1, ALS and ADF perform very similar when comparing the reconstruction quality. We assume that the reason for the worse performance of *HTOpt* is due to the missing rank-adaptivity. For high sampling ratios, the use of a rank adaptation scheme such as the one described in Section 4.9 is less important, as overfitting is very unlikely to occur.

5.4.3. Parameter-dependent PDE: Cookie Problem. As an example of a parameter-dependent PDE, we investigate the cookie problem [39, 3]: Consider the heat equation on the unit square $D = [0, 1]^2$ with $d = m^2$ disks $D_{s,t}$ of radius $\rho = \frac{1}{4m+2}$ and midpoints $(\rho(4s-1), \rho(4t-1))$ for $s, t = 1, \dots, m$. A graphical depiction of this setup is shown in Figure 5.3 for the cases $m = 3$ and $m = 4$. The heat conductivities of the different disks $D_{s,t}$ are described by the coefficient vector $p = (p_1, \dots, p_d)$, yielding the piecewise constant diffusion coefficient

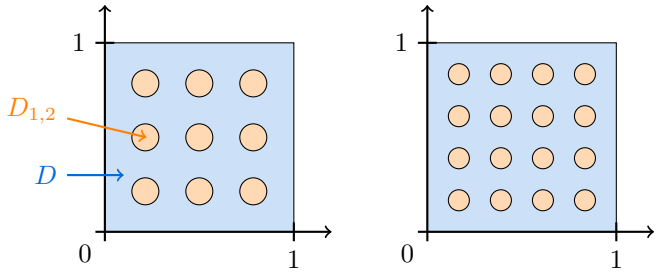
$$a(x, p) := \begin{cases} p_\mu, & \text{if } x \in D_{s,t},\ \mu = m(t-1) + s, \\ 1, & \text{otherwise.} \end{cases}$$

with each $p_\mu \in [\frac{1}{2}, 2]$. The temperature $u(x, p)$ at position $x \in D$ for a certain choice of parameters $p \in [\frac{1}{2}, 2]^d$ is then described by the diffusion equation

$$\begin{aligned} -\operatorname{div}(a(x, p)\nabla u(x, p) &= 1, \qquad x \in D, \\ u(x, p) &= 0, \qquad x \in \partial D. \end{aligned} \tag{5.1}$$

Assume now that we are interested in the average temperature $\overline{u}(p) : [\frac{1}{2}, 2]^d \to \mathbb{R}$ over the domain D,

$$\overline{u}(p) := \int_{[0,1]^2} u(x, p)\, \mathrm{d}x.$$



(a) 9 cookies ($m = 3$) (b) 16 cookies ($m = 4$)

Fig. 5.3: *Graphical depiction of the cookie problem for values $m = 3$ and $m = 4$ resulting in 9 and 16 cookies, respectively.*

Following the setup of [3], we discretize the parameter space $[\frac{1}{2}, 2]^d$ by spectral collocation using $n = 10$ Chebyshev points for each mode $\mu = 1, \dots d$. Hence, calculating the average of the solution of (5.1) for each collocation point in the parameter space would result in a solution tensor $\mathbf{A}$ of size 10^d. As this represents an infeasible amount of work, we instead try to approximate the solution tensor by a low-rank approximation $\mathbf{X}$.

A common tool to tackle such problems are cross-approximation techniques. In Table 5.3 we compare the results obtained by a recent algorithm in the hierarchical Tucker format [3], denoted by *HT-Cross*, to the most recent cross-approximation technique in the TT/MPS format [35, 12], denoted by *TT-Cross*, the routine `amen_cross` from the TT-Toolbox [31]. The results for *HT-Cross* are taken from the corresponding paper [3]. The specified tolerance is shown along with the relative reconstruction error on a test set Γ of size 100 and the number of sampling points. Both algorithms are able to construct a low-rank approximation $\mathbf{X}$ up to the prescribed accuracy, while *TT-Cross* needs about 10 times more sampling points. This could be due to an additional search along the modes of the solution tensor $\mathbf{A}$, as we have a mode size of $n = 10$ in our example. At each sampling point, the PDE (5.1) is solved for the corresponding parameter set $p = (p_1, \dots, p_d)$ using the FEniCS [25] finite element package, version 1.4.0.

To compare the performance of these two cross-approximation algorithms to our tensor completion approach, we now try to complete $\mathbf{A}$ by using the same amount of sampling points as the *HT-Cross* algorithm, but uniformly distributed. The test set Γ is the same as for *TT-Cross*. Algorithm 1 is able to recover $\mathbf{A}$ with similar accuracy.

We observe that for this problem, an adaptive choice of sampling points does not seem to perform better than a random sampling approach using tensor completion. The tensor completion approach has the added advantage that all sampling points are determined *a priori*. Thus, the expensive evaluation of the sampling points can be easily distributed to multiple computers in an embarrassingly parallel way before starting the completion procedure. In the adaptive cross-approximation, the necessary sampling points are only determined at run-time, preventing an effective parallelization.

6. Conclusion. In this paper, we have derived a tensor completion algorithm using a Riemannian optimization scheme on the manifold of TT/MPS tensors of fixed rank. We have shown that our algorithm is very competitive and significantly faster when compared to other existing tensor completion algorithms. For function-related

Tol.	TT-Cross $\varepsilon_\Gamma(\mathbf{X})$	TT-Cross Eval.	HT-Cross *(from [3])* $\varepsilon_\Gamma(\mathbf{X})$	HT-Cross *(from [3])* Eval.	Algorithm 1 $\varepsilon_\Gamma(\mathbf{X})$	Algorithm 1 $\lvert\Omega\rvert$
10^{-3}	$8.35 \cdot 10^{-4}$	11681	$3.27 \cdot 10^{-4}$	1548	$9.93 \cdot 10^{-5}$	1548
10^{-4}	$2.21 \cdot 10^{-5}$	14631	$1.03 \cdot 10^{-4}$	2784	$8.30 \cdot 10^{-6}$	2784
10^{-5}	$1.05 \cdot 10^{-5}$	36291	$1.48 \cdot 10^{-5}$	3224	$6.26 \cdot 10^{-6}$	3224
10^{-6}	$1.00 \cdot 10^{-6}$	42561	$2.74 \cdot 10^{-6}$	5338	$6.50 \cdot 10^{-7}$	5338
10^{-7}	$1.31 \cdot 10^{-7}$	77731	$1.88 \cdot 10^{-7}$	9475	$1.64 \cdot 10^{-7}$	9475

(a) 9 cookies ($m = 3$)

Tol.	TT-Cross $\varepsilon_\Gamma(\mathbf{X})$	TT-Cross Eval.	HT-Cross *(from [3])* $\varepsilon_\Gamma(\mathbf{X})$	HT-Cross *(from [3])* Eval.	Algorithm 1 $\varepsilon_\Gamma(\mathbf{X})$	Algorithm 1 $\lvert\Omega\rvert$
10^{-3}	$8.17 \cdot 10^{-4}$	22951	$3.98 \cdot 10^{-4}$	2959	$2.84 \cdot 10^{-4}$	2959
10^{-4}	$3.93 \cdot 10^{-5}$	68121	$2.81 \cdot 10^{-4}$	5261	$2.10 \cdot 10^{-5}$	5261
10^{-5}	$9.97 \cdot 10^{-6}$	79961	$1.27 \cdot 10^{-5}$	8320	$1.07 \cdot 10^{-5}$	8320
10^{-6}	$1.89 \cdot 10^{-6}$	216391	$3.75 \cdot 10^{-6}$	12736	$1.89 \cdot 10^{-6}$	12736
10^{-7}	—	—	$3.12 \cdot 10^{-7}$	26010	$7.12 \cdot 10^{-7}$	26010

(b) 16 cookies ($m = 3$)

Table 5.3: *Reconstruction results for the cookie problem using cross-approximation and tensor completion. The last entry in Table (b) for TT-Cross has been omitted due to an exceedingly high amount of function evaluations. The results for HT-Cross have been taken from the corresponding paper [3].*

tensors, a rank increasing scheme was shown to be essential to be able to recover the original data from very few samples. In our test case, tensor completion with randomly chosen samples yielded similar reconstruction errors as an adaptive cross-approximation approach with the same number of samples.

As a useful tool not limited to completion problems, we have introduced a new parametrization of elements of the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ based on a certain reorthogonalization. This helps to avoid numerical instabilites and increases the computational efficiency of operations involving tangent tensors, such as the inner product.

Theoretical lower bounds for the recoverability of low-rank tensors from few samples remain a challenging open question.

Acknowledgments. The author would like to thank Jonas Ballani, Daniel Kressner and Bart Vandereycken for helpful discussions on this paper and Melanie Kluge for providing an implementation of the ADF algorithm.

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 I. Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 2014.

[3] J. Ballani and L. Grasedyck. Hierarchical tensor approximation of output quantities of parameter-dependent PDEs. Technical report, ANCHP, EPF Lausanne, Switzerland, March 2014.

[4] J. Ballani, L. Grasedyck, and M. Kluge. Black box approximation of tensors in hierarchical Tucker format. *Linear Algebra Appl.*, 438(2):639–657, 2013.

[5] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86(4):565–589, 2000.

[6] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003.
[7] S. Börm and L. Grasedyck. Hybrid cross approximation of integral operators. *Numer. Math.*, 101(2):221–249, 2005.
[8] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory*, 56(5):2053–2080, 2009.
[9] C. Da Silva and F. J. Herrmann. Hierarchical tucker tensor optimization – applications to tensor completion. SAMPTA, 2013.
[10] C. Da Silva and F. J. Herrmann. Hierarchical tucker tensor optimization – applications to tensor completion. arxiv:1405.2096, 2014.
[11] S. V. Dolgov and I. V. Oseledets. Solution of linear systems and matrix inversion in the TT-format. *SIAM J. Sci. Comput.*, 34(5):A2718–A2739, 2012.
[12] S. V. Dolgov and D. V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.
[13] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.
[14] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra Appl.*, 261:1–21, 1997.
[15] L. Grasedyck, M. Kluge, and S. Krämer. Alternating directions fitting (ADF) of hierarchical low rank tensors. DFG SPP 1324 Preprint 149, 2013.
[16] L. Grasedyck, M. Kluge, and S. Krämer. Personal Communication, 2015.
[17] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitt.*, 36(1):53–78, 2013.
[18] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012.
[19] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed TT-rank. *Numer. Math.*, 120(4):701–731, 2012.
[20] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *JMLR*, 11:2057–2078, 2010.
[21] D. Kressner, M. Steinlechner, and A. Uschmajew. Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. *SIAM J. Sci. Comput.*, 36(5):A2346–A2368, 2014.
[22] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.
[23] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2114–2121, 2009.
[24] Y. Liu and F. Shang. An efficient matrix factorization method for tensor completion. *IEEE Signal Processing Letters*, 20(4):307–310, April 2013.
[25] A. Logg, K.-A. Mardal, and G. N. Wells. *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2012.
[26] C. Lubich, I. Oseledets, and B. Vandereycken. Time integration of tensor trains. arXiv preprint 1407.2042, 2014.
[27] 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.
[28] C. Mu, B. Huang, J. Wright, and D. Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 73–81, 2014.
[29] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edition, 2006.
[30] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.
[31] I. V. Oseledets, S. Dolgov, D. Savostyanov, and V. Kazeev. TT-Toolbox Version 2.3, 2014. Available at `https://github.com/oseledets/TT-Toolbox`.
[32] I. V. Oseledets and E. E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.
[33] H. Rauhut, R. Schneider, and Z. Stojanac. Tensor completion in hierarchical tensor representations. arXiv preprint 1404.3905, 2014.
[34] T. Rohwedder and A. Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM J. Numer. Anal.*, 51(2):1134–1162,

2013.

[35] D. V. Savostyanov and I. V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *Proceedings of 7th International Workshop on Multidimensional Systems (nDS)*. IEEE, 2011.

[36] 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.

[37] 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. *Machine Learning*, 94(3):303–351, 2014.

[38] 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.

[39] C. Tobler. *Low-rank Tensor Methods for Linear Systems and Eigenvalue Problems*. PhD thesis, ETH Zurich, Switzerland, 2012.

[40] E. E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64(4):367–380, 2000. International GAMM-Workshop on Multigrid Methods (Bonn, 1998).

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

[42] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214—1236, 2013.

Recent publications:

MATHEMATICS INSTITUTE OF COMPUTATIONAL SCIENCE AND ENGINEERING
Section of Mathematics
Ecole Polytechnique Fédérale
CH-1015 Lausanne

41.2014 FABIO NOBILE, LORENZO TAMELLINI, RAÚL TEMPONE:
Comparison of Clenshaw-Curtis and Leja quasi-optimal sparse grids for the approximation of random PDEs

42.2014 ASSYR ABDULLE, PATRICK HENNING:
A reduced basis localized orthogonal decomposition

43.2014 PAOLA F. ANTONIETTI, ILARIO MAZZIERI, ALFIO QUARTERONI:
Improving seismic risk protection through mathemactical modeling

44.2014 LUCA DEDÈ, ALFIO QUARTERONI, SEHNGFENG ZHU:
Isogeometric analysis and proper orthogonal decomposition for parabolic problems

45.2014 ZVONIMIR BUJANOVIC, DANIEL KRESSNER:
A block algorithm for computing antitriangular factorizations of symmetric matrices

46.2014 ASSYR ABDULLE:
The role of numerical integration in numerical in numerical homogenization

47.2014 ANDREA MANZONI, STEFANO PAGANI, TONI LASSILA:
Accurate solution of Bayesian inverse uncertainty quantification problems using model and error reduction methods

48.2014 MARCO PICASSO:
From the free surface flow of a viscoelastic fluid towards the elastic deformation of a solid

49.2014 FABIO NOBILE, FRANCESCO TESEI:
A multi level Monte Carlo method with control variate for elliptic PDEs with log-normal coefficients

***

01.2015 PENG CHEN, ALFIO QUARTERONI, GIANLUIGI ROZZA:
Reduced order methods for uncertainty quantification problems

02.2015 FEDERICO NEGRI, ANDREA MANZONI, DAVID AMSALLEM:
Efficient model reduction of parametrized systems by matrix discrete empirical interpolation

03.2015 GIOVANNI MIGLIORATI, FABIO NOBILE, RAÚL TEMPONE:
Convergence estimate in probability and in expectation for discrete least squares with noisy evaluations at random points

04.2015 FABIO NOBILE, LORENZO TAMELLINI, FRANCESCO TESEI, RAÚL TEMPONE:
An adaptive sparse grid alorithm for elliptic PDEs with lognormal diffusion coefficent

05.2015 MICHAEL STEINLECHNER:
Riemannian optimization for high-dimensional tensor completion