Optimization on the Hierarchical Tucker manifold - applications to tensor completion

Curt Da Silva[a,c,*], Felix J. Herrmann[b,c]

[a]*Department of Mathematics, University of British Columbia*
[b]*Department of Earth and Ocean Sciences, University of British Columbia*
[c]*Seismic Laboratory for Imaging and Modeling, University of British Columbia*
2207 Main Mall, V6T 1Z4, Vancouver, BC, Canada

Abstract

In this work, we develop an optimization framework for problems whose solutions are well-approximated by *Hierarchical Tucker* (HT) tensors, an efficient structured tensor format based on recursive subspace factorizations. By exploiting the smooth manifold structure of these tensors, we construct standard optimization algorithms such as Steepest Descent and Conjugate Gradient for completing tensors from missing entries. Our algorithmic framework is fast and scalable to large problem sizes as we do not require SVDs on the ambient tensor space, as required by other methods. Moreover, we exploit the structure of the Gramian matrices associated with the HT format to regularize our problem, reducing overfitting for high subsampling ratios. We also find that the organization of the tensor can have a major impact on completion from realistic seismic acquisition geometries. These samplings are far from idealized randomized samplings that are usually considered in the literature but are realizable in practical scenarios. Using these algorithms, we successfully interpolate large-scale seismic data sets and demonstrate the competitive computational scaling of our algorithms as the problem sizes grow.

Keywords: Hierarchical Tucker tensors, tensor completion, Riemannian manifold optimization, Gauss-Newton, differential geometry, low-rank tensor representation
2010 MSC: 15A69, 57N35, 65K10, 65Y20, 53B21, 90C90

1. Introduction

The matrix completion problem is concerned with interpolating a $m \times n$ matrix from a subset of its entries. The amount of recent successes in developing solution techniques to this problem is a result of assuming a *low-rank* model on the 2-D signal of interest and by considering subsampling schemes that *increase* the rank of the underlying matrix [8], [7], [9]. The original signal is recovered by promoting low-rank structures subject to data constraints.

Using a similar approach, we consider the problem of interpolating a $d-$dimensional tensor from samples of its entries. That is, we aim to solve,

$$\min_{\mathbf{X} \in \mathcal{H}} \frac{1}{2} \| P_\Omega \mathbf{X} - b \|_2^2, \tag{1}$$

where P_Ω is a linear operator $P_\Omega : \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d} \to \mathbb{R}^m$, $b \in \mathbb{R}^m$ is our subsampled data satisfying $b = P_\Omega \mathbf{X}^*$ for some "solution" tensor $\mathbf{X}^*$ and $\mathcal{H}$ is a *specific* class of low-rank tensors to be specified later. Under the assumption that $\mathbf{X}^*$ is well approximated by an element in $\mathcal{H}$, our goal is to recover $\mathbf{X}^*$ by solving (1). For

[*]Corresponding author
Email addresses: `curtd@math.ubc.ca` (Curt Da Silva), `fherrmann@eos.ubc.ca` (Felix J. Herrmann)

concreteness, we concern ourselves with the case when P_Ω is a restriction operator, i.e.,

$$P_\Omega \mathbf{X} = \mathbf{X}_{i_1,i_2,\ldots,i_d} \quad \text{if } (i_1, i_2, \ldots, i_d) \in \Omega,$$

and $\Omega \subset [n_1] \times [n_2] \times \cdots \times [n_d]$ is the so-called *sampling set*, where $[n] = \{1, \ldots, n\}$. In the above equation, we suppose that $|\Omega| = m \ll n_1 n_2 \ldots n_d$, so that P_Ω is a subsampling operator.

Unlike the matrix case, there is no unique notion of rank for tensors, as we shall see in Section 1.1, and there are multiple tensor formats that generalize a particular notion of *separability* from the matrix case—i.e, there is no unique extension of the SVD to tensors. Although each tensor format can lead to compressible representations of their respective class of low-rank signals, the truncation of a general signal to one of these formats requires access to the *fully* sampled tensor $\mathbf{X}$ (or at the very least *query*-based access to the tensor [4]) in order to achieve reasonable accuracy, owing to the use of truncated SVDs acting on various *matricizations* of the tensor. As in matrix completion, randomized missing entries change the behavior of the singular values and vectors of these matricizations and hence of the final approximation. Moreover, when the tensor of interest is actually a discretized continuous signal, there can be a number of constraints, physical or otherwise, that limit our ability to ideally sample it. For instance, in the seismic case, the tensor of interest is a multi-dimensional wavefield in the earth's subsurface sampled at an array of receivers located at the surface. In real-world seismic experiments, budgetary constraints or environmental obstructions can limit both the total amount of time available for data acquisition as well as the number and placement of active sources and receivers. Since seismic and other methods rely on having fully sampled data for drawing accurate inferences, tensor completion is an important technique for a variety of scientific fields that acquire multidimensional data.

In this work, we consider the class of Hierarchical Tucker (abbreviated HT) tensors, introduced in [21, 18], as our low-rank tensors of interest. The set of all such tensors is a smooth, embedded *submanifold* of $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$, first studied in [42], which we equip with a Riemannian metric. Using this Riemannian structure, we can construct optimization algorithms in order to solve (1) for d-dimensional tensors. We will also study some of the effects of higher dimensional sampling and extend ideas from compressive sensing and matrix completion to the HT tensor case for our specific seismic examples.

1.1. Previous Work

To provide the reader with some context on tensor representations, let us briefly detail some of the available structured tensor formats, including tensor completion results, here (see [26] and [28] for a very comprehensive overview). Here we let $N = \max_{i=1\ldots d} n_i$ be the maximum individual dimension size, $N^d := \prod_{i=1}^d n_i$ denote the dimension of the ambient space $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$, and, for each tensor format discussed, K is the maximum of all of the rank parameters associated to that format.

The so-called Candecomp/Parafac (CP) decomposition is a very straightforward application of the separation of variables technique. Very much like the SVD of a matrix, one stipulates that, for a function f living on a tensor product space, one can write

$$f(x_1, x_2, \ldots, x_d) \approx \sum_{i=1}^{K} f_i^{(1)}(x_1) f_i^{(2)}(x_2) \ldots f_i^{(d)}(x_d).$$

Thus its discretization can be written as

$$\mathbf{f} \approx \sum_{i=1}^{K} f_i^{(1)} \otimes f_i^{(2)} \otimes \cdots \otimes f_i^{(d)}$$

where $\otimes$ is the Kronecker product and $f_i^{(j)} \in \mathbb{R}^{n_j}$ is the discretization of the one dimensional function $f_i^{(j)}(x_j)$. In addition to its intuitive construction, the CP decomposition of rank K only requires dNK parameters versus the N^d of the full tensor and tensor-tensor operations can be performed efficiently on the underlying factors rather than the full tensors themselves (see [3] for a comprehensive set of MATLAB tools).

Unfortunately, despite the parsimoniousness of the CP construction, the approximation of an arbitrary (full) tensor by CP tensors has both theoretical and numerical difficulties. In particular, the set of all CP tensors of rank at most K is not closed, and thus the notion of a best $K-$rank approximation is difficult to compute in many cases [13]. Despite this shortcoming, various authors have proposed iterative and non-iterative algorithms in the CP format for approximating full tensors [28] as well as interpolating tensors with missing data, such as the Alternating Least Squares approach (a block Gauss-Seidel type method) proposed alongside the CP format in [10] and [22], with convergence analysis in [41], and a nonlinear least-squares optimization scheme in [2].

The CP format is a specific case of the more general *Tucker format*, which aims to write a tensor $\mathbf{f}$ as a multilinear product

$$\mathbf{f} \approx U_1 \times_1 U_2 \times_2 \ldots U_d \times_d \mathbf{C}$$

where $\mathbf{C} \in \mathbb{R}^{k_1 \times k_2 \times \ldots \times k_d}$ is the so-called *core tensor* and the matrices $U_j \in \mathbb{R}^{n_j \times k_j}$, $j = 1, \ldots, d$ are the *factors* of the decomposition. Here we use the notation of the multilinear product, that is, $U_i \times_i \mathbf{C}$ indicates that $\mathbf{C}$ is multiplied by U_i in dimension i, e.g., see [13, 28]. We will elaborate on this construction in Section 2.2. The CP format follows from this formulation when the core tensor is *diagonal*, i.e., $\mathbf{C}_{i_1,i_2,i_3,\ldots,i_d} = \mathbf{C}_{i_1,i_1,\ldots,i_1}\delta_{i_1,i_2,\ldots,i_d}$, where $\delta_{i_1,i_2,\ldots,i_d} = 1$ when $i_1 = i_2 = \cdots = i_d$ and 0 otherwise.

The Tucker format enjoys many benefits in terms of approximation properties over its CP counterpart. Namely, the set of all Tucker tensors of at most multilinear rank $\mathbf{k} = (k_1, k_2, \ldots, k_d)$ is closed and as a result every tensor $\mathbf{f}$ has a best at most multilinear rank-$\mathbf{k}$ Tucker approximation. A near-optimal approximation can be computed efficiently by means of the Higher Order SVD [12]. For the tensor completion problem, the authors in [17] consider the problem of recovering a Tucker tensor with missing entries using the Douglas-Rachford splitting technique, which decouples interpolation and regularization by nuclear norm penalization of different matricizations of the tensor into subproblems that are then solved via a particular proximal mapping. An application of this approach to seismic data is detailed in [29] for the interpolation problem and [30] for denoising. Depending on the size and ranks of the tensor to be recovered, there are theoretical and numerical indications that this approach is no better than penalizing the nuclear norm in a single matricization (see [36] for a theoretical justification in the Gaussian measurement case, as well as [39] for an experimental demonstration of this effect). Some preliminary results on theoretical guarantees for recovering low-rank Tucker tensors from subsampled measurements are given in [25] for pointwise measurements and a suitable, tensor-based incoherence condition and [35], which considers a nuclear norm penalty of the matricization of the first $d/2$ modes of $\mathbf{X}$ as opposed to a sum of nuclear norms of each of its d modes, as is typically considered.

Aside from convex relaxations of the tensor rank minimization problem, the authors in [31] develop an alternative manifold-based approach to Tucker Tensor optimization similar to our considerations for the Hierarchical Tucker case and subsequently complete such tensors with missing entries. Each evaluation of the objective and Riemannian gradient requires $O(d(N + |\Omega|)K^d + dK^{d+1})$ operations, whereas our method only requires $O(dNK^2 + d|\Omega|K^3 + dK^4)$ operations. As a result of using the Hierarchical Tucker format instead of the Tucker format, our method scales much better as d, N, and K grow.

Previous work in completing tensors in the Tensor Train format, which is the Hierarchical Tucker format with a specific, degenerate binary dimension tree, includes [19, 23], wherein the authors use an alternating least-squares approach for the tensor completion problem. The derivations of the smooth manifold structure of the set of TT tensors can be found in [24]. This work is a precursor for the manifold structure of Hierarchical Tucker tensors studied in [42], upon which we expand in this article. For a comprehensive review of various tensor formats, we refer the reader to [27, 20].

Owing to its extremely efficient storage requirements (which are *linear* in the dimension d as opposed to exponential in d), the Hierarchical Tucker format has enjoyed a recent surge in popularity for parametrizing high-dimensional problems. The hTucker toolbox [32] contains a suite of MATLAB tools for working with tensors in the HT format, including efficient vector space operations, matrix-tensor and tensor-tensor products, and truncations of full arrays to HT format. This truncation, the so-called Hierarchical SVD developed in [18], allows one to approximate a full tensor in HT format with a near-optimal approximation error. Even though the authors in [4] develop a HT truncation method that does not need access to every entry of the tensor in order to form the HT approximation, their approach requires algorithm-driven access to the entries, which does not apply for the seismic examples we consider below. A HT approach for solving dynamical systems is outlined in [33], which considers similar manifold structure as in this article applied in a different context.

1.2. Contributions and Organization

In this paper, we extend the primarily theoretical results of [42] to practical algorithms for solving optimization algorithms on the HT manifold. In Section 3.1, we introduce the Hierarchical Tucker format. We restate some of the results of [42] in Section 3.1 to provide context for the Riemannian metric we introduce on the quotient manifold in Section 4. Equipped with this metric, we can now develop optimization methods on the HT manifold in Section 5 that are fast and SVD-free. For large-scale, high-dimensional problems, the computational costs of SVDs are prohibitive and affect the scalability of tensor completion methods such as [17]. Since we are using the HT manifold rather than the Tucker manifold, we avoid an exponential dependence on the internal rank parameters as in [31]. In Section 5.4, we exploit the structure of HT tensors to regularize different matricizations of the tensor *without* having to compute SVDs of these matricizations, lessening the effects of overfitting when there are very few samples available. To the best of our knowledge, our approach is the first instance of exploiting the manifold structure of HT tensors for solving the tensor completion problem. We conclude by demonstrating the effectiveness of our techniques on interpolating various seismic data volumes with missing data points in all dimensions as well as missing receivers, which is more realistic. Our numerical results are similar to those presented previously in [11], but much more extensive and include our regularization and Gauss-Newton based methods. In this paper, we also compare our method to a reference implementation of [31] and achieve very reasonable results for our seismic data volumes.

We note that the algorithmic results here generalize readily to complex tensor completion $\mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ and more general subsampling operators P_Ω.

2. Notation

In this paper, we denote vectors by lower case letters $x, y, z, \ldots$, matrices by upper case, plain letters $A, B, C, \ldots, X, YZ$, and tensors by upper case, bold letters $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$.

2.1. Matricization

We let the *matricization* of a $n_1 \times n_2 \times \cdots \times n_d$ tensor $\mathbf{X}$ along the modes $t = (t_1, t_2, \ldots, t_k) \subset \{1, \ldots, d\}$ be the matrix $X^{(t)}$ such that the indices in t are vectorized along the rows and the indices in t^c are vectorized along the columns, i.e., if we set $s = t^C$, then

$$X^{(t)} \in \mathbb{R}^{(n_{t_1} n_{t_2} \ldots n_{t_k}) \times (n_{s_1} n_{s_2} \ldots n_{s_{d-k}})}$$
$$(X^{(t)})_{(i_{t_1}, \ldots, i_{t_k}), (i_{s_1}, \ldots, i_{s_{d-k}})} := \mathbf{X}_{i_1, \ldots, i_d}.$$

We also use the notation $(\cdot)_{(t)}$ for the *dematricization* operation, i.e., $(X^{(t)})_{(t)} = \mathbf{X}$, which reshapes the matricized version of $\mathbf{X}$ along modes t back to its full tensor form.

2.2. Multilinear product

A natural operation to consider on tensors is that of the *multilinear product* [42, 18, 28].

Definition 1. Given a $d-$tensor $\mathbf{X}$ of size $n_1 \times n_2 \times \ldots n_d$ and matrices $A_i \in \mathbb{R}^{m_i \times n_i}$, the multilinear product of $\{A_i\}_{i=1}^d$ with $\mathbf{X}$, is the $m_1 \times m_2 \times \ldots \times m_d$ tensor $\mathbf{Y} = A_1 \times_1 A_2 \times_2 \ldots A_d \times_d \mathbf{X}$, is defined in terms of the matricizations of $\mathbf{Y}$ as

$$Y^{(i)} = A_i X^{(i)} A_d^T \otimes A_{d-1}^T \otimes \ldots A_{i+1}^T \otimes A_{i-1}^T \cdots \otimes A_1^T, \quad i = 1, 2, \ldots, d.$$

Conceptually, we are applying operator each operator A_i to dimension i of the tensor $\mathbf{X}$, keeping all other coordinates fixed. For example, when A, X, B are matrices of appropriate sizes, the quantity AXB^T can be written as $AXB^T = A \times_1 B \times_2 X$.

The standard Euclidean inner product between two $d-$dimensional tensors X and Y can be defined in terms of the standard Euclidean product for vectors, by letting

$$\langle \mathbf{X}, \mathbf{Y} \rangle := \operatorname{vec}(\mathbf{X})^T \operatorname{vec}(\mathbf{Y})$$

where $\operatorname{vec}(\mathbf{X}) := X^{(1,2,\ldots,d)}$ is the usual vectorization operator. This inner product induces a norm $\|\mathbf{X}\|_2$ on the set of all $d-$dimensional tensors in the usual way, $\|\mathbf{X}\|_2 = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle}$.

Here we state several properties of the multilinear product, which are straightforward to prove.

Proposition 1. *Let $\{A_i\}_{i=1}^d$, $\{B_i\}_{i=1}^d$ be collections of linear operators and $\mathbf{X}, \mathbf{Y}$ be tensors, all of appropriate sizes, so that the multilinear products below are well-defined. Then we have the following:*

1. $(A_1 \times_1 \ldots, A_d \times_d) \circ (B_1 \times_1 \ldots, B_d \times_d \mathbf{X}) = (A_1 B_1) \times_1 \ldots (A_d B_d) \times_d \mathbf{X}$ *[13]*
2. $\langle A_1 \times \ldots A_d \times_d \mathbf{X}, B_1 \times_1 \ldots B_d \times_d \mathbf{Y} \rangle = \langle (B_1^T A_1) \times_1 \ldots (B_d^T A_d) \times_d \mathbf{X}, \mathbf{Y} \rangle$

2.3. Tensor-tensor contraction

Another natural operation to consider between two tensors is *tensor-tensor contraction*, a generalization of matrix-matrix multiplication. We define tensor-tensor contraction in terms of tensors of the same dimension for ease of presentation [16].

Definition 2. Given a $d-$tensor $\mathbf{X}$ of size $n_1 \times \cdots \times n_d$ and a $d-$tensor $\mathbf{Y}$ of size $m_1 \times \cdots \times m_d$, select $s, t \subset \{1, \ldots, d\}$ such that $|s| = |t|$ and $n_{s_i} = m_{t_i}$ for $i = 1, \ldots, |s|$. The *tensor-tensor contraction* of $\mathbf{X}$ and $\mathbf{Y}$ along modes s, t, denoted $\langle \mathbf{X}, \mathbf{Y} \rangle_{(s,t)}$, is defined as $(2d - (|s| + |t|))-$tensor Z of size (n_{s^c}, m_{t^c}), satisfying

$$\mathbf{Z} = \langle \mathbf{X}, \mathbf{Y} \rangle_{(s,t)} = (X^{(s^c)} Y^{(t)})_{(s^c),(t^c)}.$$

Tensor tensor contraction over modes s and t merely sums over the dimensions specified by s, t in $\mathbf{X}$ and $\mathbf{Y}$ respectively, leaving the dimensions s^c and t^c free.

The inner product $\langle \mathbf{X}, \mathbf{Y} \rangle$ is a special case of the tensor product when $s = t = \{1, \ldots, d\}$.

We also make use of the fact that when the index sets s, t are $s, t = [d] \setminus i$ with $\mathbf{X}$, $\mathbf{Y}$, and A_i are appropriately sized for $i = 1, \ldots, d$, then

$$\langle A_1 \times_1 A_2 \times_2 \ldots A_d \times_d \mathbf{X}, \mathbf{Y} \rangle_{[d]\setminus i,[d]\setminus i} = A_i \langle A_1 \times_1 A_2 \times_2 \ldots A_{i-1} \times_{i-1} A_{i+1} \times_{i+1} \ldots A_d \times_d \mathbf{X}, \mathbf{Y} \rangle_{[d]\setminus i,[d]\setminus i} \tag{2}$$

i.e., applying A_i to dimension i commutes with contracting tensors over every dimension except the ith one.

3. Smooth Manifold Geometry of the Hierarchical Tucker Format

In this section, we review the definition of the Hierarchical Tucker format (Section 3.1) as well as previous results [42] in the smooth manifold geometry of this format (Section 3.2). We extend these results in the next section by introducing a Riemannian metric on the space of HT parameters and subsequently derive the associated Riemannian gradient with respect to this metric. A reader familiar with the results in [42] can glance over this section quickly for a few instances of notation and move on to Section 4.

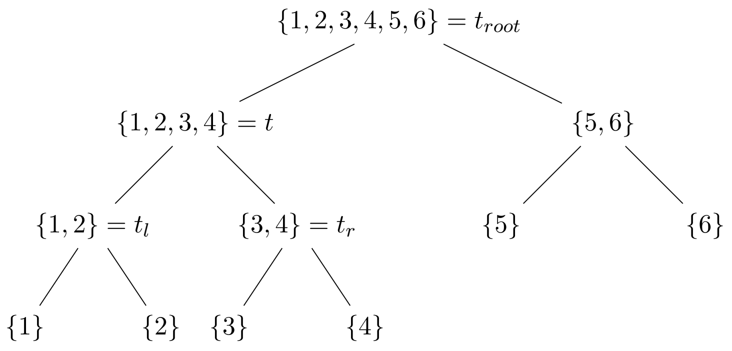


Figure 1: Complete dimension tree for $\{1,2,3,4,5,6\}$.

3.1. Hierarchical Tucker Format

The standard definition of the Hierarchical Tucker format relies on the notion of a *dimension tree*, chosen apriori, which specifies the format [18]. Intuitively, the dimension tree specifies which groups of dimensions are "separated" from other groups of dimensions, where "separation" is used in a similar sense to the SVD in two dimensions.

Definition 3. A *dimension tree* T is a non-trivial binary tree such that

- the root, $\mathrm{t_{root}}$, has the label $\mathrm{t_{root}} = \{1, 2, \ldots, d\}$
- for every $t \notin L$, where L is the set of leaves of T, the labels of its left and right children, t_l, t_r, form a partition of the label for t, i.e., $t_l \cup t_r = t$ and $t_l \cap t_r = \emptyset$.

An example of a dimension tree when $d = 6$ is given in Figure 1.

Remark 1. For the following derivations, we take the point of view of each quantity with a subscript $(\cdot)_t$ is associated to the node $t \in T$. By Definition 3, for each $t \in T$, there is a corresponding subset of $\{1, \ldots, d\}$ associated to t. If our HT tensor has dimensions $n_1 \times n_2 \times \ldots \times n_d$, we let $n_t = \prod_{i \in t} n_i$ and, when $t \in T \setminus L$, n_t satisfies $n_t = n_{t_l} n_{t_r}$.

Definition 4. Given a dimension tree T and a vector of *hierarchical ranks* $(k_t)_{t \in T}$ with $k_t \in \mathbb{Z}^+$, a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ can be written in the *Hierarchical Tucker format* if there exist parameters $x = ((U_t)_{t \in L}, (\mathbf{B}_t)_{t \in T \setminus L})$ such that $\phi(x) = \mathbf{X}$, where

$$\begin{aligned} \mathrm{vec}(\phi(x)) &= U_{t_l} \times_1 U_{t_r} \times_2 B_{\mathrm{t_{root}}} && t = \mathrm{t_{root}} \\ U_t &= (U_{t_l} \times_1 U_{t_r} \times_2 \mathbf{B}_t)^{(1,2)} && t \notin L \cup \mathrm{t_{root}} \end{aligned} \tag{3}$$

where $U_t \in \mathbb{R}_*^{n_t \times k_t}$, the set of full-rank $n_t \times k_t$ matrices, for $t \in L$ and $\mathbf{B}_t \in \mathbb{R}_*^{k_{t_l} \times k_{t_r} \times k_t}$, the set of 3-tensors of full *multilinear* rank, i.e.,

$$\mathrm{rank}(B_t^{(1)}) = k_{t_l}, \quad \mathrm{rank}(B_t^{(2)}) = k_{t_r}, \quad \mathrm{rank}(B_t^{(3)}) = k_t.$$

We say the parameters $x = (U_t, \mathbf{B}_t)$ are in *Orthogonal Hierarchical Tucker* (OHT) format if, in addition to the above construction, we also have

$$\begin{aligned} U_t^T U_t &= I_{k_t} \quad \text{for } t \in L \\ (B_t^{(1,2)})^T B_t^{(1,2)} &= I_{k_t} \quad \text{for } t \notin L \cup \mathrm{t_{root}} \end{aligned} \tag{4}$$

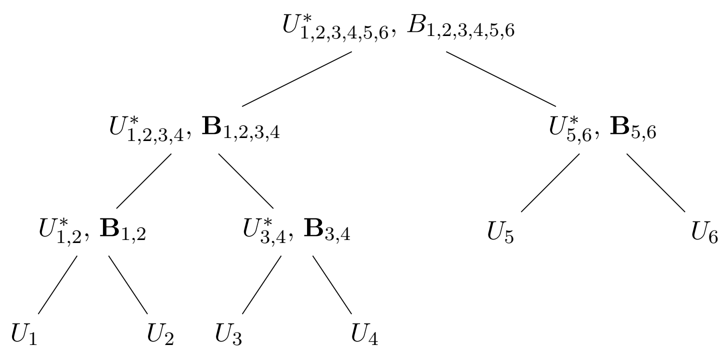

$U^*_{1,2,3,4,5,6}$, $B_{1,2,3,4,5,6}$
$U^*_{1,2,3,4}$, $\mathbf{B}_{1,2,3,4}$
$U^*_{5,6}$, $\mathbf{B}_{5,6}$
$U^*_{1,2}$, $\mathbf{B}_{1,2}$
$U^*_{3,4}$, $\mathbf{B}_{3,4}$
U_5
U_6
U_1
U_2
U_3
U_4

Figure 2: Dimension tree representation of (3) with dimensions $\{1,2,3,4,5,6\}$. Starred quantities are computed recursively.

We have made a slight modification of the definition of the HT format compared to [42] for ease of presentation. When $d = 2$, our construction is the same as the subspace decomposition introduced in [34] for low-rank matrices, but our approach is not limited to this case.

Owing to the recursive construction (4), the intermediate matrices U_t for $t \in T \setminus L$ do not need to be stored. Instead, specifying U_t for $t \in L$ and $\mathbf{B}_t$ for $t \in T \setminus L$ determines $\mathbf{X} = \phi(x)$ completely. Therefore, the overall number of parameters $x = ((U_t)_{t\in L}, (\mathbf{B}_t)_{t\in T\setminus L})$ is bounded above by $dNK + (d-2)K^3 + K^2$, where $N = \max_{i=1,\dots,d} n_i$ and $K = \max_{t\in T} k_t$. When $d \geq 4$ and $K \ll N$, this quantity is much less than the N^d parameters typically needed to represent $\mathbf{X}$.

Definition 5. The *hierarchical rank* of a tensor $\mathbf{X} \in \mathbb{R}^{n_1\times n_2\times\dots\times n_d}$ corresponding to a dimension tree T is the vector $\mathbf{k} = (k_t)_{t\in T}$ where

$$k_t = \operatorname{rank}(X^{(t)}).$$

We consider the set of *Hierarchical Tucker tensors of fixed rank* $\mathbf{k} = (k_t)_{t\in T}$, that is,

$$\mathcal{H} = \{\mathbf{X} \in \mathbb{R}^{n_1\times n_2\times\dots\times n_d} \,|\, \operatorname{rank}(X^{(t)}) = k_t \quad \text{for}\, t \in T \setminus \mathrm{t_{root}}\}.$$

We restrict ourselves to parameters x that are strictly orthogonalized, as in (4). In addition to significantly simplifying the resulting notation, this restriction allows us to avoid cumbersome and unnecessary matrix inversions, in particular for the resulting subspace projections in future sections. Moreover, using only orthogonalized parameters avoids the problem of algorithms converging to points with possibly lower than prescribed HT rank, see [42, Remark 4.1]. This restriction does not reduce the expressibility of the HT format, however, since for any non orthogonalized parameters x such that $\mathbf{X} = \phi(x)$, there exists orthogonalized parameters x' with $\mathbf{X} = \phi(x')$ [18, Alg. 3].

We use the grouping $x = (U_t, \mathbf{B}_t)$ to denote $((U_t)_{t\in L}, (\mathbf{B}_t)_{t\in T\setminus L})$, as these are our "independent variables" of interest in this case. In order to avoid cumbersome notation, we also suppress the dependence on $(T, \mathbf{k})$ in the following, and presume a fixed dimension tree T and hierarchical ranks $\mathbf{k}$.

3.2. Quotient Manifold Geometry

The results in this section are adapted from those in [42] to the orthogonalized parameter case and we include them here in the interest of keeping this article self contained.
Below, let $S_{k_{t_l},k_{t_r},k_t}$ be the closed submanifold of $\mathbb{R}_*^{k_{t_l}\times k_{t_r}\times k_t}$, the set of 3−tensors with full multilinear rank, such that $\mathbf{S} \in S_{k_{t_l},k_{t_r},k_t}$ is orthonormal along modes 1 and 2, i.e., $(S^{(1,2)})^T S^{(1,2)} = I_{k_t}$ and let $\mathrm{St}(n_t, k_t)$ be

the $n_t \times k_t$ Stiefel manifold of $n \times k_t$ matrices with orthonormal columns.
Our orthogonal parameter space $\mathcal{M}$ is then

$$\mathcal{M} = \bigtimes_{t \in L} \mathrm{St}(n_t, k_t) \times \bigtimes_{t \notin L \cup \mathrm{t_{root}}} S_{k_{t_l}, k_{t_r}, k_t} \times \mathbb{R}_*^{k_{(\mathrm{t_{root}})_l} \times k_{(\mathrm{t_{root}})_r}}$$

with corresponding tangent space at $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$

$$\mathcal{T}_x \mathcal{M} = \bigtimes_{t \in L} \mathcal{T}_{U_t} \mathrm{St}(n_t, k_t) \times \bigtimes_{t \notin L \cup \mathrm{t_{root}}} \mathcal{T}_{\mathbf{B}_t} S_{k_{t_l}, k_{t_r}, k_t} \times \mathbb{R}^{k_{(\mathrm{t_{root}})_l} \times k_{(\mathrm{t_{root}})_r}}.$$

Note that $\mathcal{T}_Y \mathrm{St}(n,p) = \{Y\Omega + Y^{\perp}K : \Omega^T = -\Omega \in \mathbb{R}^{p \times p}, K \in \mathbb{R}^{(n-p) \times p}\}$. We omit an explicit description of $\mathcal{T}_{B_t} S_{k_{t_l}, k_{t_r}, k_t}$ for brevity.

Let $\phi : \mathcal{M} \to \mathcal{H}$ be the parameter to tensor map in (3). Then for each $\mathbf{X} \in \mathcal{H}$, then there is an inherent ambiguity in its representation by parameters x, i.e., $\mathbf{X} = \phi(x) = \phi(y)$ for distinct parameters x and y with the following relationship between them.

Let $\mathcal{G}$ be the Lie group

$$\mathcal{G} = \{(A_t)_{t \in T} : A_t \in O(k_t)\}.$$

where $O(p)$ is the orthogonal group of $p \times p$ matrices and the group action of component-wise multiplication. Let θ be the group action

$$\begin{aligned} &\theta : \mathcal{M} \times \mathcal{G} \to \mathcal{M} \\ &(x, \mathcal{A}) := ((U_t, B_t), (A_t)) \mapsto \theta_x(\mathcal{A}) := (U_t A_t, A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \mathbf{B}_t). \end{aligned} \tag{5}$$

Then $\phi(x) = \phi(y)$ if and only if there exists a unique $\mathcal{A} = (A_t)_{t \in T} \in \mathcal{G}$ such that $x = \theta_A(y)$ [42, Prop. 3.9]. Therefore these are the only types of ambiguities we must consider in this format.

It follows that the orbit of x,

$$\mathcal{G}\, x = \{\theta_{\mathcal{A}}(x) : \mathcal{A} \in \mathcal{G}\},$$

is the set of all parameters that map to the same tensor $\mathbf{X} = \phi(x)$ under ϕ. This induces an equivalence relation on the set of parameters $\mathcal{M}$,

$$x \sim y \text{ if and only if } y \in \mathcal{G}\, x.$$

If we let $\mathcal{M}/\mathcal{G}$ be the corresponding quotient space of equivalence classes and $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{G}$ denote the quotient map, then pushing ϕ down through π results in an injective function

$$\hat{\phi} : \mathcal{M}/\mathcal{G} \to \mathcal{H}$$

whose image is all of $\mathcal{H}$, and hence is an isomorphism (in fact, a diffeomorphism).

The vertical space, $\mathcal{V}_x \mathcal{M}$, is the subspace of $\mathcal{T}_x\mathcal{M}$ that is tangent to $\pi^{-1}(x)$. That is, $dx^v = (\delta U_t^v, \delta \mathbf{B}_t^v) \in \mathcal{V}_x \mathcal{M}$ when it is of the form [42, Eq. 4.7]

$$\begin{aligned} \delta U_t^v &= U_t D_t && \text{for } t \in L \\ \delta \mathbf{B}_t^v &= D_t \times_3 \mathbf{B}_t - D_{t_l} \times_1 \mathbf{B}_t - D_{t_r} \times_2 \mathbf{B}_t && \text{for } t \in T \setminus L \cup \mathrm{t_{root}} \\ \delta B_{\mathrm{t_{root}}}^v &= -D_{t_l} B_{\mathrm{t_{root}}} - B_{\mathrm{t_{root}}} D_{t_r}^T && \text{for } t = \mathrm{t_{root}} \end{aligned} \tag{6}$$

where $D_t \in \mathrm{Skew}(k_t)$, the set of $k_t \times k_t$ skew symmetric matrices. A straightforward computation shows that $D\phi(x)\big|_{\mathcal{V}_x \mathcal{M}} \equiv 0$, and therefore for every $dx^v \in \mathcal{V}_x \mathcal{M}$, $\phi(x) = \phi(x + dx^v)$. From an optimization point of view, moving from the point x to $x + dx^v$ will not change the current tensor $\phi(x)$ and therefore for any search direction p, we must filter out the corresponding component in $\mathcal{V}_x\,\mathcal{M}$ in order to compute the gradient correctly. We accomplish this by projecting on to a *horizontal space*, which is any complementary subspace to $\mathcal{V}_x\,\mathcal{M}$. One such choice is [42, Eq. 4.8],

$$\mathcal{H}_x\,\mathcal{M} = \left\{ (\delta U_t^h, \delta \mathbf{B}_t^h) : \begin{array}{r} (\delta U_t^h)^T U_t = 0_{k_t} \text{ for } t \in L \\ (\delta B_t^{(1,2)})^T B_t^{(1,2)} = 0_{k_t} \text{ for } t \notin L \cup \mathrm{t_{root}} \end{array} \right\}. \tag{7}$$

Note that there is no restriction on $B^h_{\mathrm{t_{root}}}$, which is a matrix.
This choice has the convenient property that $\mathcal{H}_x\,\mathcal{M}$ is *invariant* under the action of θ, i.e., [42, Prop. 4.9]

$$D\theta(x,\mathcal{A})[\mathcal{H}_x\,\mathcal{M}, 0] = \mathcal{H}_{\theta_x(\mathcal{A})}\,\mathcal{M}, \tag{8}$$

which we shall exploit for our upcoming discussion of a Riemannian metric. The horizontal space $\mathcal{H}_x\,\mathcal{M}$ allows us to uniquely represent abstract tangent vectors in $T_{\pi(x)}\,\mathcal{M}/\mathcal{G}$ with concrete vectors in $\mathcal{H}_x\,\mathcal{M}$.

4. Riemannian Geometry of the HT Format

In this section, we introduce a Riemannian metric on the parameter space $\mathcal{M}$ that will allow us to use parameters x as representations for their equivalence class $\pi(x)$ in a well-defined manner while performing numerical optimization.

4.1. Riemannian metric

Since each distinct equivalence class $\pi(x)$ is uniquely identified with each distinct value of $\phi(x)$, the quotient manifold $\mathcal{M}\,/\,\mathcal{G}$ is really our manifold of interest for the purpose of computations—i.e, we would like to formulate our optimization problem over the equivalence classes $\pi(x)$. Unfortunately, $\mathcal{M}\,/\,\mathcal{G}$ is an abstract mathematical object and thus hard to implement numerically. By introducing a Riemannian metric on $\mathcal{M}$ that respects its quotient structure, we can formulate concrete optimization algorithms in terms of the HT parameters without being affected by the non-uniqueness of the format—i.e., by optimizing over parameters x while implicitly performing optimization over equivalence classes $\pi(x)$. Below, we explain how to explicitly construct this Riemannian metric for the HT format.

Let $\eta_x = (\delta U_t, \delta \mathbf{B}_t), \zeta_x = (\delta V_t, \delta \mathbf{C}_t) \in \mathcal{T}_x\mathcal{M}$ be tangent vectors at the point $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$. Then we define the inner product $g_x(\cdot,\cdot)$ at x as

$$\begin{aligned} g_x(\eta_x, \zeta_x) := & \sum_{t \in T} \mathrm{tr}((U_t^T U_t)^{-1} \delta U_t^T \delta V_t) \\ & + \sum_{t \notin L \cup \mathrm{t_{root}}} \langle \delta \mathbf{B}_t, (U_{t_l}^T U_{t_l}) \times_1 (U_{t_r}^T U_{t_r}) \times_2 (U_t^T U_t)^{-1} \times_3 \delta \mathbf{C}_t \rangle \\ & + \mathrm{tr}(U^T_{(\mathrm{t_{root}})_r} U_{(\mathrm{t_{root}})_r} \delta B^T_{\mathrm{t_{root}}} U^T_{(\mathrm{t_{root}})_l} U_{(\mathrm{t_{root}})_l} \delta C_{\mathrm{t_{root}}}). \end{aligned} \tag{9}$$

By the full-rank conditions on U_t and $\mathbf{B}_t$ at each node, by definition of the HT format, each $U_t^T U_t$ for $t \in T$ is symmetric positive definite and varies smoothly with $x = (U_t, \mathbf{B}_t)$. As a result, g_x is a smooth, symmetric positive definite, bilinear form on $\mathcal{T}_x\mathcal{M}$, i.e., a Riemannian metric. Note that when x is in OHT, as it is in our case, g_x reduces to the standard Euclidean product on the parameter space $\mathcal{T}_x\mathcal{M}$, making it straightforward to compute in this case.

Proposition 2. *On the Riemannian manifold $(\mathcal{M}, g)$, θ defined in (5) acts* isometrically *on $\mathcal{M}$, i.e., for every $\mathcal{A} \in \mathcal{G}$, $\xi_x, \zeta_x \in \mathcal{H}_x\,\mathcal{M}$*

$$g_x(\xi_x, \zeta_x) = g_{\theta_{\mathcal{A}}(x)}(\theta^* \xi_x, \theta^* \zeta_x)$$

where θ^* is the push-forward *map, $\theta^* v = D\theta(x, \mathcal{A})[v]$.*

Proof. Let $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$, $y = (V_t, \mathbf{C}_t) = \theta_{\mathcal{A}}(x) = (U_t A_t, A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \mathbf{B}_t)$ for $\mathcal{A} \in \mathcal{G}$.

If we write $\eta_x = (\delta U_t, \delta \mathbf{B}_t)$, $\zeta_x = (\delta V_t, \delta \mathbf{C}_t)$ for $\eta_x, \zeta_x \in \mathcal{H}_x \mathcal{M}$, then, by (8), it follows that

$$\eta_y = \theta^* \eta_x = (\delta U_t A_t, A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \delta \mathbf{B}_t)$$

and similarly for ζ_y.

We will compare each component of the sum of (9) term by term. For ease of presentation, we only consider interior nodes $t \notin L \cup \mathrm{t_{root}}$, as leaf nodes and the root node are handled in an analogous manner.

For $t \notin L \cup \mathrm{t_{root}}$, let $(\xi_y)_t$ be the component of ξ_y at the node t, i.e.,

$$\begin{aligned}(\xi_y)_t &= A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \delta \mathbf{B}_t \\ &:= \widetilde{\delta \mathbf{B}_t},\end{aligned}$$

and similarly let $\widetilde{\delta \mathbf{C}_t} := (\zeta_y)_t$.

A straightforward computation based on (3) and (5) yields that

$$V_{t_l}^T V_{t_l} \times_1 V_{t_r}^T V_{t_r} \times_2 (V_t^T V_t)^{-1} \times_3 \mathbf{Y} = (A_{t_l}^T U_{t_l}^T U_{t_l} A_{t_l}) \times_1 (A_{t_r}^T U_{t_r}^T U_{t_r} A_{t_r}) \times_2 (A_t^T (U_t^T U_t)^{-1} A_t) \times_3 \mathbf{Y}$$

for appropriately sized $\mathbf{Y}$. In particular, for $\mathbf{Y} = \widetilde{\delta \mathbf{C}_t}$, we have that

$$\begin{aligned}&\langle \widetilde{\delta \mathbf{B}_t}, V_{t_l}^T V_{t_l} \times_1 V_{t_r}^T V_{t_r} \times_2 (V_t^T V_t)^{-1} \times_3 \widetilde{\delta \mathbf{C}_t} \rangle \\ &= \langle A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \delta \mathbf{B}_t, \\ &((A_{t_l}^T U_{t_l}^T U_{t_l} A_{t_l}) \times_1 (A_{t_r}^T U_{t_r}^T U_{t_r} A_{t_r}) \times_2 (A_t^T (U_t^T U_t)^{-1} A_t) \times_3) \circ (A_{t_l}^T \times_1 A_{t_r}^T \times_2 A_t^T \times_3 \delta \mathbf{C}_t) \rangle \\ &= \langle \delta \mathbf{B}_t, (U_{t_l}^T U_{t_l}) \times_1 (U_{t_r}^T U_{t_r}) \times_2 (U_t^T U_t)^{-1} \times_3 \delta \mathbf{C}_t \rangle \quad \text{using Prop. 1.2 and } A_t \in O(k_t).\end{aligned}$$

Adding the terms for each $t \in T$, we obtain

$$g_x(\xi_x, \zeta_x) = g_{\theta_{\mathcal{A}}(x)}(\theta^* \xi_x, \theta^* \zeta_x).$$

□

Although the above computation uses the fact that $A_t \in O(k_t)$, an almost identical calculation yields Proposition 2 when x is non orthogonalized, as considered in [42]. As we are interested in carrying out our optimization using the HT parameters x as proxies for their equivalence classes $\pi(x)$, this proposition states that if we measure inner products between two tangent vectors at the point x, we obtain the same result as if we had measured the inner product between two tangent vectors transformed by $\theta_{\mathcal{A}}$ at the point $\theta_{\mathcal{A}}(x)$. In this sense, once we have a unique association of tangent vectors in $\mathcal{M}/\mathcal{G}$ with a subspace of $\mathcal{T}_x\mathcal{M}$, we can use the actual representatives, the parameters x, instead of the abstract equivalence class $\pi(x)$, in a well-defined way during our optimization. This shows that $\mathcal{M}/\mathcal{G}$, endowed with the Riemannian metric

$$g_{\pi(x)}(\xi, \zeta) := g_x(\xi_x^h, \zeta_x^h)$$

where ξ^h, ζ_x^h are the horizontal lifts at x of ξ, ζ, respectively, is a Riemannian quotient manifold of $\mathcal{M}$ (i.e., $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{G}$ is a Riemannian submersion) [1, Sec. 3.6.2].

In summary, by using this Riemannian metric and restricting our optimization to only consider horizontal tangent vectors, we can implicitly formulate our algorithms on the abstract quotient space by working with the concrete HT parameters. Below, we will derive the Riemannian gradient in this context.

Remark 2. It should be noted that although the horizontal space (7) is complementary to the vertical space (6), it is demonstrably *not* perpendicular to $\mathcal{V}_x \mathcal{M}$ under the Riemannian metric (9). Choosing a horizontal space which is perpendicular to $\mathcal{V}_x \mathcal{M}$ under the standard Euclidean product (i.e., (9) when x is orthogonalized) is beyond the scope of this paper. Suffice to say, it can be done, as a generalization of the approach outlined in [34], resulting in a series of symmetry conditions on various multi-way combinations of parameters. The resulting projection operators involve solving a number of coupled Lyapunov equations, increasing with the depth of T. It remains to be seen whether such equations can be solved efficiently when d is large. We will not dwell on this point here, as we will not be needing orthogonal projections for our computations in the following.

4.2. Riemannian gradient

The problem we are interested in solving is

$$\min_{x \in \mathcal{M}} f(\phi(x))$$

for a smooth objective function $f : \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d} \to \mathbb{R}$. We write $\hat{f} : \mathcal{M} \to \mathbb{R}$, where $\hat{f}(x) = f(\phi(x))$.

We need to derive expressions for the Riemannian gradient to update the HT parameters as part of local optimization procedures. Therefore, our primary quantity of interest is the *Riemannian gradient* of $\hat{f}$.

Definition 6. [1, Sec. 3.6] Given a smooth scalar function $\hat{f}$ on a Riemannian manifold $\mathcal{N}$, the *Riemannian gradient* of $\hat{f}$ at $x \in \mathcal{N}$, denoted $\nabla^R \hat{f}(x)$, is the unique element of $\mathcal{T}_x\mathcal{N}$ which satisfies

$$g_x(\nabla^R \hat{f}(x), \xi) = D\hat{f}(x)\,[\xi] \quad \forall \xi \in \mathcal{T}_x\mathcal{N}$$

with respect to the Riemannian metric $g_x(\cdot, \cdot)$.

Our manifold of interest in this case is $\mathcal{N} = \mathcal{M}/\mathcal{G}$, with the corresponding horizontal space $\mathcal{H}_x\,\mathcal{M}$ in lieu of the abstract tangent space $T_{\pi(x)}\,\mathcal{M}/\mathcal{G}$. Therefore, in the above equation, we can consider the horizontal lift ξ^h of the tangent vector ξ and instead write

$$g_x(\nabla^R \hat{f}(x), \xi^h) = D\hat{f}(x)[\xi^h].$$

Our derivation is similar to that of [42, Sec 6.2.2], except our derivations are more streamlined and cheaper computationally since we reduce the operations performed at the interior nodes $t \in T \setminus L$. By a slight abuse of notation in this section, we denote variational quantities associated to node t as $\delta Z_t \in \mathbb{R}^{n_{t_l} n_{t_r} \times k_t}$ and $\delta \mathbf{Z}_t \in \mathbb{R}^{n_{t_l} \times n_{t_r} \times k_t}$ where $(\delta Z_t)_{(1,2)} = \delta \mathbf{Z}_t$ is the reshaping of δZ_t in to a 3−tensor.

Since $x = (U_t, \mathbf{B}_t)$ is orthogonalized, we use $\langle \cdot, \cdot \rangle$ to denote the Euclidean inner product. By the chain rule, we have that, for any $\xi = (\delta V_t, \delta \mathbf{C}_t) \in \mathcal{H}_x\,\mathcal{M}$,

$$\begin{aligned} D\hat{f}(x)[\xi] &= Df(\phi(x))[D\phi(x)[\xi]] \\ &= \langle \nabla_{\phi(x)} f(\phi(x)), D\phi(x)[\xi] \rangle. \end{aligned}$$

Then each tensor $\delta \mathbf{V}_t \in \mathbb{R}^{n_{t_l} \times n_{t_r} \times k_t}$, with $\delta V_{\mathrm{t_{root}}} = D\phi(x)[\xi]$, satisfies the recursion

$$\delta \mathbf{V}_t = \delta V_{t_l} \times_1 U_{t_r} \times_2 \mathbf{B}_t + U_{t_l} \times_1 \delta V_{t_r} \times_2 \mathbf{B}_t + U_{t_l} \times_1 U_{t_r} \times_2 \delta \mathbf{C}_t, \tag{10}$$

for matrices $\delta V_{t_l} \in \mathbb{R}^{n_{t_l} \times k_{t_l}}$, $\delta V_{t_r} \in \mathbb{R}^{n_{t_r} \times k_{t_r}}$ and tensor $\delta \mathbf{C_t} \in \mathbb{R}^{k_{t_l} \times k_{t_r} \times k_t}$ satisfying [42, Lemma 2]

$$\delta V_{t_l}^T U_{t_l} = 0 \quad \delta V_{t_r}^T U_{t_r} = 0 \quad (\delta C_t^{(1,2)})^T B_t^{(1,2)} = 0. \tag{11}$$

The third orthogonality condition is omitted when $t = \mathrm{t_{root}}$.

Owing to this recursive structure, we compute $\langle \delta \mathbf{U}_t, \delta \mathbf{V}_t \rangle$, where $\delta \mathbf{U}_t$ is the component of the Riemannian gradient at the current node and recursively extract the components of the Riemannian gradient associated to the children, i.e., $\delta U_{t_l}, \delta U_{t_r}$, and $\delta \mathbf{B}_t$. Here we let $\delta U_{\mathrm{t_{root}}} = \nabla_{\phi(x)} f(\phi(x))$ be the Euclidean gradient of $f(\phi(x))$ at $\phi(x)$, reshaped into a matrix of size $n_{(\mathrm{t_{root}})_l} \times n_{(\mathrm{t_{root}})_r}$.

Let $\dfrac{\partial U_t}{\partial U_{t_l}}$ be the linear operator such that $\dfrac{\partial U_t}{\partial U_{t_l}}\delta V_{t_l} = \delta V_{t_l} \times_1 U_{t_r} \times_2 \mathbf{B}_t$ and similarly for $\dfrac{\partial U_t}{\partial U_{t_l}}, \dfrac{\partial U_t}{\partial \mathbf{B_t}}$. Then $\langle \delta \mathbf{U}_t, \delta \mathbf{V}_t \rangle$ is equal to

$$\langle \delta \mathbf{U}_t, \frac{\partial U_t}{\partial U_{t_l}}\delta V_{t_l}\rangle + \langle \delta \mathbf{U}_t, \frac{\partial U_t}{\partial U_{t_r}}\delta V_{t_r}\rangle + \langle \delta \mathbf{U}_t, \frac{\partial U_t}{\partial \mathbf{B}_t}\delta \mathbf{C}_t\rangle. \tag{12}$$

If we set

$$\delta U_{t_l} = P^{\perp}_{U_{t_l}} \left(\frac{\partial U_t}{\partial U_{t_l}}\right)^T \delta \mathbf{U}_t,\ \delta U_{t_r} = P^{\perp}_{U_{t_r}} \left(\frac{\partial U_t}{\partial U_{t_r}}\right)^T \delta \mathbf{U}_t,\ \delta \mathbf{B}_t = \left(P^{\perp}_M \left(\left(\frac{\partial U_t}{\partial B_t}\right)^T \delta \mathbf{U}_t\right)^{(1,2)}\right)_{(1,2)} \tag{13}$$

where

$$\begin{aligned} P^{\perp}_M &= I_{k_t} - B_t^{(1,2)}(B_t^{(1,2)})^T && \text{if } t \neq \mathrm{t_{root}} \\ P^{\perp}_M &= I && \text{if } t = \mathrm{t_{root}} \end{aligned}$$

and $P_{U_t} = (I_{k_t} - U_t U_t^T)$ is the usual projection on to $\mathrm{span}(U_t)^{\perp}$, then we have that (12) is equal to

$$\langle \delta U_{t_l}, \delta V_{t_l}\rangle + \langle \delta U_{t_r}, \delta V_{t_r}\rangle + \langle \delta \mathbf{B}_t, \delta \mathbf{C}_t\rangle$$

and $\delta U_{t_l}, \delta U_{t_r}$, and $\delta \mathbf{B}_t$ satisfy (11). Their recursively decomposed factors will therefore be in the horizontal space $\mathcal{H}_x \mathcal{M}$.

If t_l is a leaf node, then we have extracted the component of the Riemannian gradient associated to t_l. Otherwise, we set $\delta \mathbf{U}_{t_l} = (\delta U_{t_l})_{(1,2)}$ apply the above recursion. We make the same considerations for the right children.

We compute the adjoint partial derivatives via

$$\frac{\partial U_t}{\partial U_{t_l}}^T \delta \mathbf{U}_t = \langle U_{t_r}^T \times_2 \delta \mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)},\ \frac{\partial U_t}{\partial U_{t_r}}^T \delta \mathbf{U}_t = \langle U_{t_l}^T \times_1 \delta \mathbf{U}_t, \mathbf{B}_t \rangle_{(1,3),(1,3)},\ \frac{\partial U_t}{\partial \mathbf{B}_t}^T \delta \mathbf{U}_t = U_{t_l}^T \times_1 U_{t_r}^T \times_2 \delta \mathbf{U}_t \tag{14}$$

For the general case of computing these adjoint operators, we refer to Appendix A. In the above computations, the multilinear product operators are never formed explicitly and instead each operator is applied to various reshapings of the matrix or tensor of interest, see [15] for a reference Matlab implementation.

In order to minimize the number of computations performed on intermediate tensors, which are much larger than $\dim(\mathcal{M})$, we first note that in computing the terms

$$P^{\perp}_{U_{t_l}} \langle U_{t_r}^T \times_2 \delta \mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)},$$

that $\delta \mathbf{U}_t = (P^{\perp}_{U_t} \delta \tilde{U}_t)_{(1,2)}$ for a matrix $\delta \tilde{U}_t \in \mathbb{R}^{n_{t_l} n_{t_r} \times k_t}$. Using (2), the above expression can be written as

$$\langle P^{\perp}_{U_{t_l}} \times_1 U_{t_r}^T \times_2 (P^{\perp}_{U_t} \delta \tilde{U}_t)_{(1,2)}, \mathbf{B}_t \rangle_{(2,3),(2,3)}. \tag{15}$$

We note that in the above, $P^{\perp}_{U_{t_l}} \times_1 U_{t_r}^T \times_2 (P^{\perp}_{U_t} \delta \tilde{U}_t)_{(1,2)} = (U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} P^{\perp}_{U_t} \delta \tilde{U}_t)_{(1,2)}$, and the operator applied to $\delta \tilde{U}_t$ satisfies

$$\begin{aligned} U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} P^{\perp}_{U_t} &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} (I_{n_t} - U_t U_t^T) \\ &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} (I_{n_t} - U_{t_r} \otimes U_{t_l} B_t^{(1,2)} (B_t^{(1,2)})^T U_{t_r}^T \otimes U_{t_l}^T) \\ &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}}. \end{aligned}$$

This means that, using (2), we can write (15) as

$$P_{U_{t_l}}^{\perp} \langle U_{t_r}^T \times_2 (\delta \tilde{U}_t)_{(1,2)}, \mathbf{B}_t \rangle_{(2,3),(2,3)}$$

i.e., we do not have to apply $P_{U_t}^{\perp}$ to the matrix $\delta \tilde{U}_t$ at the parent node of t. Applying this observation recursively and to the other terms in the Riemannian gradient, we merely need to orthogonally project the resulting extracted parameters $(\delta U_t, \delta \mathbf{B}_t)$ on to $\mathcal{H}_x \mathcal{M}$ after applying the formula (13) *without* applying the intermediate operators $P_{U_t}^{\perp}$, reducing the overall computational costs. We summarize our algorithm for computing the Riemannian gradient in Algorithm 1.

Algorithm 1 The Riemannian gradient $\nabla^R f$ at a point $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$

```
Require: $x = (U_t, \mathbf{B}_t)$ parameter representation of the current point
  Compute $\mathbf{X} = \phi(x)$ and $\nabla_{\mathbf{X}} f(\mathbf{X})$, the Euclidean gradient of $f$, a $n_1 \times \dots n_d$ tensor.
  $\delta U_{t_{\text{root}}} \leftarrow (\nabla_{\mathbf{X}} f(\mathbf{X}))_{(1,2)}$
  for $t \in T \setminus L$, visiting parents before their children do
    $\delta \mathbf{U}_t \leftarrow (\delta U_t)_{(1,2)}$
    $\delta U_{t_l} \leftarrow \langle U_{t_r}^T \times_2 \delta \mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)}$, $\delta U_{t_r} \leftarrow \langle U_{t_l}^T \times_1 \delta \mathbf{U}_t, \mathbf{B}_t \rangle_{(1,3),(1,3)}$
    $\delta \mathbf{B}_t \leftarrow U_{t_l}^T \times_1 U_{t_r}^T \times_2 \delta \mathbf{U}_t$
    if $t \neq \text{t}_{\text{root}}$ then
      $\delta \mathbf{B_t} \leftarrow (P_{B_t^{(1,2)}}^{\perp} (\delta B_t)^{(1,2)})_{(1,2)}$
    end if
  end for
  for $t \in L$ do
    $\delta U_t \leftarrow P_{U_t}^{\perp} \delta U_t$
  end for
  return $\nabla^R f \leftarrow (\delta U_t, \delta \mathbf{B}_t)$
```

Algorithm 1 is computing the operator $D\phi(x)^* : \mathcal{T}_{\phi(x)} \mathcal{H} \to \mathcal{H}_x \mathcal{M}$ applied to the Euclidean gradient $\nabla_{\phi(x)} f(\phi(x))$. The forward operator $D\phi(x) : \mathcal{H}_x \mathcal{M} \to \mathcal{T}_{\phi(x)} \mathcal{H}$ can be computed using a component-wise orthogonal projection $P_x^H : \mathcal{T}_x \mathcal{M} \to \mathcal{H}_x \mathcal{M}$ followed by applying (10) recursively.

4.3. Tensor Completion Objective and Gradient

In this section, we specialize the computation of the objective and Riemannian gradient in the HT format to the case where the Euclidean gradient of the objective function is sparse, in particular for tensor completion. This will allow us to scale our method to high dimensions in a straightforward fashion as opposed to the inherently dense considerations in Algorithm 1. Here for simplicity, we suppose that our dimension tree T is *complete*, that is a full binary tree up to level depth$(T) - 1$ and all of the leaves at level depth(T) are on the leftmost side of T, as in Figure 1. This will ease the exposition as well as allow for a more efficient implementation compared to a noncomplete tree.

We consider a separable, smooth objective function on the HT manifold,

$$\hat{f}(x) = f(\phi(x)) = \sum_{\mathbf{i} \in \Omega} f_{\mathbf{i}}(\phi(x)_{\mathbf{i}}), \tag{16}$$

where $f_{\mathbf{i}} : \mathbb{R} \to \mathbb{R}$ is a smooth, single variable function. For the least-squares tensor completion problem, $f_{\mathbf{i}}(a) = \frac{1}{2}(a - b_{\mathbf{i}})^2$.

We denote $\mathbf{i} = (i_1, i_2, \dots, i_d)$ and let $\mathbf{i}_t$ be the subindices of $\mathbf{i}$ indexed by $t \in T$. In this section, we also use the Matlab notation for indexing in to matrices, i.e., $A(m, n)$ is the (m, n)th entry of A, and similarly for tensors. Let $K = \max_{t \in T} k_t$.

4.3.1. *Objective function*

With this notation in mind, we write each entry of $P_\Omega \phi(x)$, indexed by $\mathbf{i} \in \Omega$, as

$$(P_\Omega \phi(x))(\mathbf{i}) = \sum_{r_l=1}^{k_{t_l}} \sum_{r_r=1}^{k_{t_r}} (U_{t_l})(\mathbf{i}_{t_l}, r_l) \cdot (U_{t_r})(\mathbf{i}_{t_r}, r_r) \cdot B_{\mathrm{t_{root}}}(r_l, r_r), \quad \text{where } t = \mathrm{t_{root}}\,.$$

Each entry of U_{t_l}, U_{t_r} can be computed by applying the recursive formula (3), i.e.,

$$U_t(\mathbf{i}_t, r) = \sum_{r_l=1}^{k_{t_l}} \sum_{r_r=1}^{k_{t_r}} (U_{t_l})(\mathbf{i}_{t_l}, r_l) \cdot (U_{t_r})(\mathbf{i}_{t_r}, r_r) \cdot \mathbf{B}_t(r_l, r_r, r)$$

with the substitutions of $t \to t_l, t_r$ as appropriate.

At each node $t \in T$, we perform at most K^3 operations and therefore the computation of $P_\Omega \phi(x)$ requires at most $2|\Omega| d K^3$ operations. The least squares objective, $\frac{1}{2}\|P_\Omega \phi(x) - b\|_2^2$, can be computed in $|\Omega|$ operations.

4.3.2. *Riemannian gradient*

The Riemannian gradient is more involved, notation-wise, to derive explicitly compared to the objective, so in the interest of brevity we only concentrate on the recursion for computing δU_1 below.

We let $\mathbf{Z} = \nabla_{\phi(x)} f(\phi(x))$ denote the Euclidean gradient of $f(\mathbf{X})$ evaluated at $\mathbf{X} = \phi(x)$, which has nonzero entries $\mathbf{Z}(\mathbf{i})$ indexed by $\mathbf{i} \in \Omega$. By expanding out (14), for each $\mathbf{i} \in \Omega$, δU_{t_l} evaluated at the root node with coordinates $\mathbf{i}_{t_l}, r_l$ for $r_l = 1, \ldots, k_{t_l}$ is

$$\delta U_{t_l}(\mathbf{i}_{t_l}, r_l) = \sum_{\mathbf{i} = (\mathbf{i}_{t_l}, \mathbf{i}_{t_r}) \in \Omega} \mathbf{Z}(\mathbf{i}) \sum_{r_r=1}^{k_{t_r}} U_{t_r}(\mathbf{i}_{t_r}, r_r) B_{\mathrm{t_{root}}}(r_l, r_r), \quad \text{where } t = \mathrm{t_{root}}\,.$$

For each $t \in T \setminus L \cup \mathrm{t_{root}}$, we let $\widetilde{\delta U_{t_l}}$ denote the length k_{t_l} vector, which depends on $\mathbf{i}_t$ that satisfies for each $\mathbf{i} \in \Omega, r_l = 1, ..., k_{t_l}$

$$(\widetilde{\delta U_{t_l}})(\mathbf{i}_t, r_{t_l}) = \sum_{r_r=1}^{k_{t_r}} \sum_{r_t=1}^{k_t} U_{t_r}(\mathbf{i}_{t_r}, r_r) \mathbf{B}_t(r_l, r_r, r_t) \widetilde{\delta U_t}(\mathbf{i}_t, r_t).$$

This recursive construction above, as well as similar considerations for the right children for each node, yield Algorithm 2. For each node $t \in T \setminus \mathrm{t_{root}}$, we perform $3|\Omega|K^3$ operations except at the root where we perform $3|\Omega|K^2$ operations. The overall computation of the Riemannian gradient requires at most $6d|\Omega|K^3$ operations, when T is complete, and a negligible $O(dK)$ additional storage to store the vectors $\widetilde{\delta U_t}$ for each fixed $\mathbf{i} \in \Omega$. The computations above are followed by componentwise orthogonal projection on to $\mathcal{H}_x \mathcal{M}$, which requires $O(d(NK^2 + K^4))$ operations and are dominated by the $O(d|\Omega|K^3)$ time complexity when $|\Omega|$ is large.

In total, each evaluation of the objective, with or without the Riemannian gradient, requires $O(d|\Omega|K^3)$ operations in this case. Since $f(\mathbf{X})$ exhibits this separable structure and the parameters $x = (U_t, \mathbf{B}_t)$ are typically very small, it is straightforward to compute the objective and its gradient in an embarrassingly parallel manner for very large problems.

By comparison, the gradient in the Tucker tensor completion case [31] requires $O(d(|\Omega| + N)K^d + K^{d+1})$ operations, which scales much more poorly when $d \geq 4$ compared to using Algorithm 2. This discrepancy is a result of the structural differences between Tucker and Hierarchical Tucker tensors, the latter of which allows one to exploit additional low-rank behaviour of the core tensor in the Tucker format.

In certain situations, when say $|\Omega| = pN^d$ for some $p \in [10^{-3}, 1]$ and d is sufficiently small, say $d = 4, 5$, it may be more efficient from a computer hardware point of view to use the dense linear algebra formulation in Algorithm 1 together with an efficient dense linear algebra library such as BLAS, rather than Algorithm 2. The dense formulation requires $O(N^d K)$ operations when T is a balanced tree, which may be smaller than the $O(d|\Omega|K^3)$ operations needed in this case.

Algorithm 2 Objective & Riemannian gradient for separable objectives

```
Require: $x = (U_t, \mathbf{B}_t)$ parameter representation of the current point
  $f_x \leftarrow 0$, $\delta U_t, \delta \mathbf{B}_t \leftarrow 0$, $\widetilde{\delta U_t} \leftarrow 0$
  for $\mathbf{i} \in \Omega$ do
    for $t \in T \setminus L$, visiting children before their parents do
      for $z = 1, 2, \ldots, k_t$ do
        $U_t(\mathbf{i}_t, z) \leftarrow \sum_{w=1}^{k_l} \sum_{y=1}^{k_r} (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot (U_{t_r})(\mathbf{i}_{t_r}, y) \cdot \mathbf{B}_t(w, y, z)$
      end for
    end for
    $f_x \leftarrow f_x + f_{\mathbf{i}}(U_{t_{\text{root}}}(\mathbf{i}))$
    $\widetilde{\delta U_{t_{\text{root}}}} \leftarrow \nabla f_{\mathbf{i}}(U_{t_{\text{root}}}(\mathbf{i}))$
    for $t \in T \setminus L$, visiting parents before their children do
      for $w = 1, \ldots, k_{t_l}$, $y = 1, \ldots, k_{t_r}$, $z = 1, \ldots, k_t$ do
        $\delta \mathbf{B}_t(w, y, z) \leftarrow \delta \mathbf{B}_t(w, y, z) + \widetilde{\delta U_t}(z) \cdot (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot (U_{t_r})(\mathbf{i}_{t_r}, y)$
      end for
      for $w = 1, \ldots, k_{t_l}$ do
        $\widetilde{\delta U_{t_l}}(w) \leftarrow \sum_{y=1}^{k_{t_r}} \sum_{z=1}^{k_t} (U_{t_r})(\mathbf{i}_{t_r}, y) \cdot \mathbf{B}_t(w, y, z) \cdot \widetilde{\delta U_t}(z)$
      end for
      for $y = 1, \ldots, k_{t_r}$ do
        $\widetilde{\delta U_{t_r}}(y) \leftarrow \sum_{w=1}^{k_{t_l}} \sum_{z=1}^{k_t} (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot \mathbf{B}_t(w, y, z) \cdot \widetilde{\delta U_t}(z)$
      end for
    end for
    for $t \in L$ do
      for $z = 1, 2, \ldots, k_t$ do
        $\delta U_t(\mathbf{i}_t, z) \leftarrow \delta U_t(\mathbf{i}_t, z) + \widetilde{\delta U_t}(z)$
      end for
    end for
    Project $(\delta U_t, \delta \mathbf{B}_t)$ componentwise on to $\mathcal{H}_x \mathcal{M}$
  end for
  return $f(x) \leftarrow f_x$, $\nabla^R f(x) \leftarrow (\delta U_t, \delta \mathbf{B}_t)$
```

5. Optimization

5.1. Reorthogonalization as a retraction

The exponential mapping on a Riemannian manifold captures the notion of "minimal distance" movement in a particular tangent direction. Although it has many theoretically desirable properties, the exponential mapping is often numerically difficult or expensive to compute as it involves computing matrix exponentials or solving ODEs. The strict, distance-minimizing properties of the exponential mapping can be relaxed, while still preserving algorithmic convergence, resulting in the notion of a *retraction* on a manifold.

Definition 7. A retraction on a manifold $\mathcal{N}$ is a smooth mapping R from the tangent bundle $\mathcal{T}\mathcal{N}$ onto $\mathcal{N}$ with the following properties: Let R_x denote the restriction of R to $\mathcal{T}_x\mathcal{N}$.

- $R_x(0_x) = x$, where 0_x denotes the zero element of $\mathcal{T}_x\mathcal{N}$
- With the canonical identification $\mathcal{T}_{0_x}\mathcal{T}_x\mathcal{N} \simeq \mathcal{T}_x\mathcal{N}$, R_x satisfies

$$DR_x(0_x) = \text{id}_{\mathcal{T}_x\mathcal{N}}$$

  where $\text{id}_{\mathcal{T}_x\mathcal{N}}$ denotes the identity mapping on $\mathcal{T}_x\mathcal{N}$ (Local rigidity condition).

A retraction approximates the action of the exponential mapping to first order and hence much of the analysis for algorithms utilizing the exponential mapping can also be immediately carried over to those using retractions.

A computationally feasible retraction on the HT parameters is given by QR- or square-root-based re-orthogonalization (Appendix C). The QR-based orthogonalization of (potentially nonorthogonal) parameters $x = (U_t, \mathbf{B}_t)$, denoted $QR(x)$, is given in Algorithm 3 [18, Alg. 3].

Proposition 3. *Given $x \in \mathcal{M}$, $\eta \in \mathcal{T}_x \mathcal{M}$, let $QR(x)$ be the QR-based orthogonalization defined in Algorithm 3. Then $R_x(\eta) := QR(x + \eta)$ is a retraction on $\mathcal{M}$.*

We refer to Appendix B for the proof of this proposition.

As before for the Riemannian metric, we can treat the retractions on the HT parameter space as implicitly being retractions on the quotient space as outlined below.
Since $R_x(\eta)$ is a retraction on the parameter space $\mathcal{M}$, and our horizontal space is invariant under the Lie group action, by the discussion in [1, 4.1.2], we have the following

Proposition 4. *The mapping*

$$R_{\pi(X)}(z) = \pi(R_X(Z))$$

is a retraction on $\mathcal{M}/\mathcal{G}$, where $R_X(Z)$ is the QR or square-root based retraction (Appendix C) previously defined on $\mathcal{M}$, $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{G}$ is the projection operator, and Z is the horizontal lift at X of the tangent vector z at $\pi(X)$.

Algorithm 3 QR-based orthogonalization

```
Require: HT parameters $x = (U_t, \mathbf{B}_t)$
  return $y = (V_t, \mathbf{C}_t)$ orthogonalized parameters such that $\phi(x) = \phi(y)$
  for $t \in L$ do
    $Q_t R_t = U_t$, where $Q_t$ is orthogonal and $R_t$ is upper triangular
    $V_t \leftarrow Q_t$
  end for
  for $t \in T \setminus (L \cup \mathrm{t_{root}})$, visiting children before their parents do
    $Z_t \leftarrow R_{t_l} \times_1 R_{t_r} \times_2 \mathbf{B}_t$
    $Q_t R_t = Z_t^{(1,2)}$, where $Q_t$ is orthogonal and $R_t$ is upper triangular
    $\mathbf{C}_t \leftarrow (Q_t)_{(1,2)}$
  end for
  $C_{\mathrm{t_{root}}} \leftarrow R_{(\mathrm{t_{root}})_l} B_{\mathrm{t_{root}}} R^T_{(\mathrm{t_{root}})_r}$
```

5.2. Vector transport

Now that we have a method for "moving" in a particular direction along the HT manifold, we need a means of mapping tangent vectors from one point to another. For this purpose, we use the notion of vector transport, which relaxes the isometry constraints of parallel transport to decrease computational complexity. Even though we make this approximation, we still enjoy increased convergence rates compared to steepest descent (see [1, Sec. 8.1.1] for more details).

Since our parameter space $\mathcal{M}$ is a subset of Euclidean space, given a point $x \in \mathcal{M}$ and a horizontal vector $\eta_x \in \mathcal{H}_x \mathcal{M}$, we take our vector transport $\mathcal{T}_{x,\eta_x} : \mathcal{H}_x \mathcal{M} \to \mathcal{H}_{R_x(\eta_x)} \mathcal{M}$ of the vector $\xi_x \in \mathcal{H}_x \mathcal{M}$ to be

$$\mathcal{T}_{x,\eta_x} \xi_x := P^h_{R_x(\eta_x)} \xi_x$$

where P^h_x is the component-wise projection onto the horizontal space at x [1, Sec. 8.1.4]. This mapping is well defined on $\mathcal{M}/\mathcal{G}$ since $\mathcal{H}_x \mathcal{M}$ is invariant under θ, and induces a vector transport on the quotient space.

5.3. Smooth optimization methods

Now that we have established the necessary components for manifold optimization, we present a number of concrete optimization algorithms for solving

$$\min_{x \in \mathcal{M}} f(\phi(x)).$$

5.3.1. First-order methods

Given the expressions for the Riemannian gradient and retraction, it is straightforward to implement the classical Steepest Descent algorithm with an Armijo line search on this Riemannian manifold, specialized from the general Riemannian manifold case [1] to the HT manifold in Algorithm 4. This algorithm consists of computing the Riemannian gradient, followed by a line search, HT parameter update, and a reorthogonalization. Since this algorithm has a poor convergence rate, we rely on more sophisticated optimization algorithms such as the nonlinear conjugate gradient method as outlined in Algorithm 4.

Here g_i denotes the Riemannian gradient at iteration i of the algorithm, p_i is the search direction for the optimization method, and α_i is the step length.

We choose the Polak-Ribiere approach

$$\beta_i = \frac{\langle g_i, g_i - \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}}(g_{i-1})\rangle}{\langle g_{i-1}, g_{i-1}\rangle}$$

to compute the CG-parameter β_i, so that the search direction p_i satisfies

$$p_i = -g_i + \beta_i \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} p_{i-1}$$

and $p_1 = -g_1$.

5.3.2. Line search

As any gradient based optimization scheme, we need a good initial step size and a computationally efficient line search. Following [37], we use a variation of the limited-minimization line search approach to set the initial step length based on the previous search direction and gradient that are vector transported to the current point—i.e, we have

$$\begin{aligned} s_i &= \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} \alpha_{i-1} p_{i-1} \\ y_i &= g_i - \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} g_{i-1}. \end{aligned}$$

In this context, s_i is the manifold analogue for the Euclidean difference between iterates, $x_i - x_{i-1}$ and y_i is the manifold analogue for the difference of gradients between iterates, $g_i - g_{i-1}$, which are standard optimization quantities in optimization algorithms set in $\mathbb{R}^n$.

Our initial step size for the direction p_i is given as

$$\alpha_0 = -g_i^T p_i / (L_i \|p_i\|_2^2)$$

where $L_i = y_i^T s_i / \|s_i\|_2^2$ is the estimate of the Lipschitz constant for the gradient [38, Eq. 16]. Because we are operating in the HT parameter space, the above computations require $O(\dim(M)) = O(dNK + (d-1)K^3)$ operations, much less than the $2|\Omega|(d+1)K^d$ operations used in [31] to initialize their line search. We justify this choice because we are working on large-scale problems where we have to limit the number of operations in the full tensor space, even when $|\Omega|$ is small.

Moreover, computing the gradient is much more expensive than evaluating the objective. For this reason, we use a simple Armijo-type back-/forward-tracking approach that only involves function evaluations and seeks to minimize the $1D$ function $f(x + \alpha p_i)$ quasi-optimally, i.e., to find $m \in \mathbb{Z}$ such that $\alpha = \theta^m \alpha_0$ for $\sigma > 0$

$$\begin{aligned} f(x_i + \alpha p_i) - f(x_i) &\le \sigma \alpha g_i^T p_i \qquad (17) \\ f(x_i + \alpha p_i) &\le \min\{f(x_i + \theta \alpha p_i), f(x_i + \theta^{-1} \alpha p_i)\} \end{aligned}$$

Algorithm 4 General Nonlinear Conjugate Gradient method for minimizing a function f over $\mathcal{H}$

```
Require: Initial guess $x_0 = (U_t, \mathbf{B}_t)$, $0 < \sigma < 1$ sufficient decrease parameter for the Armijo line search,
  $0 < \theta < 1$ step size decrease parameter, $\gamma > 0$ CG restart parameter
  $p_{-1} \leftarrow 0$
  $i \leftarrow 0$
  for i = $0, 1, 2, \ldots$ until convergence do
    $\mathbf{X}_i \leftarrow \phi(x_i)$
    $f_i \leftarrow f(\mathbf{X}_i)$
    $g_i \leftarrow \nabla^R \hat{f}(x_i)$                                    ▷ Riemannian gradient of $\hat{f}(x)$ at $x_i$
    $s_i \leftarrow \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} \alpha_{i-1} p_{i-1}$     ▷ Vector transport the previous search direction
    $y_i \leftarrow g_i - \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} g_{i-1}$
    $L_i \leftarrow y_i^T s_i / \|s_i\|^2$                                    ▷ Lipschitz constant estimate
    $p_i \leftarrow -g_i + \beta_i \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} p_{i-1}$
    if $\langle p_i, g_i \rangle > -\gamma$ then
      $p_i = -g_i$                                                            ▷ Restart CG direction
    end if
    if $y_i^T s_i > 0$ then
      $\alpha \leftarrow -g_i^T p_i / (L_i \|p_i\|_2^2)$
    else
      $\alpha \leftarrow \alpha_{i-1}$
    end if
    Find $m \in \mathbb{Z}$ such that $\alpha_i = \alpha \theta^m$ and
        $f(x_i + \alpha_i p_i) - f_i \leq \sigma \alpha_i g_i^T p_i$
        $f(x_i + \alpha_i p_i) < \min\{f(x_i + \alpha_i \theta p_i), f(x_k + \alpha_i \theta^{-1} p_i)\}$   ▷ Find a quasi-optimal minimizer
    $x_{i+1} \leftarrow R_{x_i}(\alpha_i p_i)$                                 ▷ Reorthogonalize
    $i \leftarrow i + 1$
  end for
```

so $\alpha \approx \alpha^* = \operatorname{argmin}_\alpha f(x_i + \alpha p_i)$ in the sense that increasing or decreasing α by a factor of θ will increase $f(x_i + \alpha p_i)$. After the first few iterations of our optimization procedure, we observe empirically that our line search only involves two or three additional function evaluations to verify the second inequality in (17), i.e., our initial step length α_0 is quasi-optimal.

Because $\phi(R_x(\alpha\eta)) = \phi(x + \alpha\eta)$ for any $x \in \mathcal{M}$ and horizontal vector η, where R_x is either the QR or square-root based retraction, Armijo linesearches do not require reorthogonalization, which further reduces computational costs.

5.3.3. Gauss-Newton Method

Because of the least-squares structure of our tensor completion problem (1), we can approximate the Hessian by the Gauss-Newton Hessian

$$H_{GN} := D\phi^*(x) D\phi(x) : \mathcal{H}_x \mathcal{M} \to \mathcal{H}_x \mathcal{M}.$$

Note that we do not use the "true" Gauss-Newton Hessian, $D\phi^*(x) P_\Omega^* P_\Omega D\phi(x)$, for the tensor completion case, since for even moderate subsampling ratios, $P_\Omega^* P_\Omega$ is close to the zero operator and this Hessian is very poorly conditioned as a result.

Since $D\phi(x) : \mathcal{H}_x \mathcal{M} \to \mathcal{T}_{\phi(x)} \mathcal{H}$ is an isomorphism, it is easy to see that H_{GN} is symmetric and positive definite on $\mathcal{H}_x \mathcal{M}$. The solution to the Gauss-Newton equation is then

$$H_{GN} \xi = -\nabla^R f(x)$$

for $\xi \in \mathcal{H}_x \mathcal{M}$. We can simplify the computation of H_{GN} by exploiting the recursive structure of $D\phi^*(x)$

and $D\phi(x)$, thereby avoiding intermediate vectors of size $\mathbb{R}^{n_1\times n_2\times\ldots\times n_d}$ in the process. We write at the root

$$\begin{aligned}\delta U'_{t_l} &= (I - U_{t_l}U_{t_l}^T)\frac{\partial U_t}{\partial U_{t_l}}^T D\phi(x)[\xi],\\ \delta U'_{t_r} &= (I - U_{t_r}U_{t_r}^T)\frac{\partial U_t}{\partial U_{t_r}}^T D\phi(x)[\xi],\\ \delta B'_t &= \frac{\partial U_t}{\partial \mathbf{B}_t}^T D\phi(x)[\xi]\end{aligned}$$

where

$$D\phi(x)[\xi] = \delta U_{t_l} \times_1 U_{t_r} \times_2 B_t + U_{t_l} \times_1 \delta U_{t_r} \times_2 B_t + U_{t_l} \times_1 U_{t_r} \times_2 \delta B_t, \quad t = \mathrm{t_{root}}\,.$$

In the above expression, $D\phi(x)$ is horizontal, so that for each $t \in T \setminus \mathrm{t_{root}}$, δU_t is perpendicular to U_t (11). A straightforward computation simplifies the above expression to

$$\begin{aligned}\delta U'_{t_l} &= \left(\frac{\delta U_{\mathrm{t_{root}}}}{\delta U_{t_l}}\right)^T \delta U_{\mathrm{t_{root}}} = \delta U_{t_l} B_{\mathrm{t_{root}}} B_{\mathrm{t_{root}}}^T\\ &\qquad\qquad\qquad\quad := \delta U_{t_l} G_{t_l},\\ \delta U'_{t_r} &= \left(\frac{\delta U_{\mathrm{t_{root}}}}{\delta U_{t_r}}\right)^T \delta U_{\mathrm{t_{root}}} = \delta U_{t_r} B_{\mathrm{t_{root}}}^T B_{\mathrm{t_{root}}}\\ &\qquad\qquad\qquad\quad := \delta U_{t_r} G_{t_r},\\ \delta B'_t &= \left(\frac{\delta U_{\mathrm{t_{root}}}}{\delta B_{\mathrm{t_{root}}}}\right)^T \delta U_{\mathrm{t_{root}}} = \delta B_{\mathrm{t_{root}}}.\end{aligned}$$

This expression gives us the components of the horizontal vector $\delta U'_{t_l}, \delta U'_{t_r}$ sent to the left and right children, respectively, as well as the horizontal vector $\delta B'_t$.

We proceed recursively by considering a node $t \in T \setminus L \cup \mathrm{t_{root}}$ and let $\delta U_t G_t$ be the contribution from the parent node of t. By applying the adjoint partial derivatives, followed by an orthogonal projection on to $\mathcal{H}_x\,\mathcal{M}$, we arrive at a simplified form for the Gauss-Newton Hessian

$$\begin{aligned}P^{\perp}_{U_{t_l}} \frac{\delta U_t}{\delta U_{t_l}}^T \delta U_t G_t &= \langle \delta U_{t_l} \times_1 G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)}\\ &:= \delta U_{t_l} G_{t_l},\\ P^{\perp}_{U_{t_r}} \frac{\delta U_t}{\delta U_{t_r}}^T \delta U_t G_t &= \delta U_{t_r} G_{t_r},\\ P^{\perp}_{B_t^{(1,2)}} \frac{\delta U_t}{\delta \mathbf{B}_t}^T \delta U_t G_t &= \delta G_t \times_3 \mathbf{B}_t.\end{aligned}$$

In these expressions, the matrices G_t are the Gramian matrices associated to the HT format, initially introduced in [18] and used for truncation of a general tensor to the HT format as in [40]. They satisfy, for $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$,

$$\begin{aligned}G_{\mathrm{t_{root}}} &= 1 \qquad (18)\\ G_{t_l} &= \langle G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)}\\ G_{t_r} &= \langle G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(1,3),(1,3)},\end{aligned}$$

i.e., the same recursion as G_t in the above derivations. Each G_t is a $k_t \times k_t$ symmetric positive definite matrix (owing to the full rank constraints of the HT format) and also satisfies

$$\lambda_j(G_t) = \sigma_j(X^{(t)})^2 \qquad (19)$$

where $\lambda_j(A)$ is the jth eigenvalue of the matrix A and $\sigma_j(A)$ is the jth singular value of A.

Assuming that each G_t is well conditioned, applying the inverse of H_{GN} follows directly, summarized in Algorithm 5. For the case where our solution HT tensor exhibits *quickly*-decaying singular values of

Algorithm 5 $H_{GN}^{-1}\zeta$

```
Require: Current point $x = (U_t, \mathbf{B}_t)$, horizontal vector $\zeta = (\delta U_t, \delta \mathbf{B}_t)$
  Compute $(G_t)_{t\in T}$ using (18)
  for $t \in T \setminus \mathrm{t_{root}}$ do
    if $t \in L$ then
      $\widetilde{\delta U_t} \leftarrow \delta U_t G_t^{-1}$
    else
      $\widetilde{\delta \mathbf{B}_t} \leftarrow G_t^{-1} \times_3 \delta \mathbf{B}_t$
    end if
  end for
  return $H_{GN}^{-1}\zeta \leftarrow (\widetilde{\delta U_t}, \widetilde{\delta \mathbf{B}_t})$
```

the matricizations, as is typically the assumption on the underlying tensor, the Gauss-Newton Hessian becomes poorly conditioned as the iterates converge to the solution, owing to (19). This can be remedied by introducing a small $\epsilon > 0$ and applying $(G_t + \epsilon I)^{-1}$ instead of G_t^{-1} in Algorithm 5 or by applying H_{GN}^{-1} by applying H_{GN} in a truncated PCG method. For efficiency purposes, we find the former option preferable. Alternatively, we can also avoid ill-conditioning via regularization, as we will see in the next section.

Remark 3. We note that applying the inverse Gauss-Newton Hessian to a tangent vector is akin to ensuring that the projection on to the horizontal space is orthogonal, as in [42, 6.2.2]. Using this method, however, is much faster than the previously proposed method, because applying Algorithm 5 only involves matrix-matrix operations on the small parameters, as opposed to operations on much larger intermediate matrices that live in the spaces between the full tensor space and the parameter space.

5.4. Regularization

In the tensor completion case, when there is little data available, interpolating via the Gauss-Newton method is susceptible to overfitting if one chooses the ranks $(k_t)_{t\in T}$ for the interpolated tensor too high. In that case, one can converge to solutions in null(P_Ω) that try leave the current manifold, associated to the ranks $(k_t)_{t\in T}$, to another nearby manifold corresponding to higher ranks. This can lead to degraded results in practice, as the actual ranks for the solution tensor are almost always unknown. One can use cross-validation techniques to estimate the proper internal ranks of the tensor, but we still need to ensure that the solution tensor has the predicted ranks for this approach to be successful – i.e., the iterates x must stay away from the boundary of $\mathcal{H}$.

To avoid our HT iterates converging to the manifold boundary, we introduce a regularization term on the singular values of the HT tensor $\phi(x) = \mathbf{X}$. To accomplish this, we exploit the hierarchical structure of $\mathbf{X}$ and specifically the property of the Gramian matrices G_t in (19) to ensure that *all* matricizations of $\mathbf{X}$ remain well-conditioned *without* having to perform SVDs on each matricization $X^{(t)}$. The latter approach would be prohibitively expensive when d or N are even moderately large.

Instead, we penalize the growth of the Frobenius norm of $X^{(t)}$ and $(X^{(t)})^\dagger$, which indirectly controls the largest and smallest singular values of $X^{(t)}$. We implement this regularization via the Gramian matrices in the following way. From (19), it follows that $\mathrm{tr}(G_t) = \|G_t\|_* = \|X^{(t)}\|_F^2$ and likewise $\mathrm{tr}(G_t^{-1}) = \|G_t^{-1}\|_* = \|(X^{(t)})^\dagger\|_F^2$. Our regularizer is then

$$R((\mathbf{B}_{t'})_{t'\in T}) = \sum_{t\in T} \mathrm{tr}(G_t) + \mathrm{tr}(G_t^{-1}).$$

A straightforward calculation shows that for $\mathcal{A} \in \mathcal{G}$

$$(G_t)_{t\in T,x} = (A_t^T G_t A_t)_{t\in T,\theta_{\mathcal{A}}(x)}$$

for A_t orthogonal. Therefore, our regularizer R is well-defined on the quotient manifold in the sense that it is $\theta-$invariant on the parameter space $\mathcal{M}$. This is the same regularization term considered in [31], used for (theoretically) preventing the iterate from approaching the boundary of $\mathcal{H}$. In our case, we can leverage the structure of the Gramian matrices to implement this regularizer in a computationally feasible way.

Since in the definition of the Gramian matrices (18), G_t is computed recursively via tensor-tensor contractions (which are smooth operations), it follows that the mapping $g : (\mathbf{B}_t)_{t\in T\setminus L} \to (G_t)_{t\in T}$ is smooth. In order to compute its derivatives, we consider a node $t \in T \setminus \mathrm{t_{root}}$ and consider the variations of its left and right children, i.e.,

$$\begin{aligned}\delta G_{t_r} &= \frac{\partial G_{t_r}}{\partial \mathbf{B}_t}\delta \mathbf{B}_t + \frac{\partial G_{t_r}}{\partial G_t}\delta G_t \\ \delta G_{t_l} &= \frac{\partial G_{t_l}}{\partial \mathbf{B}_t}\delta \mathbf{B}_t + \frac{\partial G_{t_l}}{\partial G_t}\delta G_t.\end{aligned} \tag{20}$$

We can take the adjoint of this recursive formulation, and thus obtain the gradient of g, if we compute the adjoint partial derivatives in (20) as well as taking the adjoint of the recursion itself. To visualize this process, we consider the relationship between input variables and output variables in the recursion as a series of small directed graphs.

$$\begin{tikzcd}
\delta B_t \arrow[dr, "\frac{\partial G_{t_r}}{\partial B_t}"'] & & \delta G_t \arrow[dl, "\frac{\partial G_{t_r}}{\partial G_t}"] & \delta B_t \arrow[dr, "\frac{\partial G_{t_l}}{\partial B_t}"'] & & \delta G_t \arrow[dl, "\frac{\partial G_{t_l}}{\partial G_t}"] \\
& \delta G_{t_r} & & & \delta G_{t_l} &
\end{tikzcd}$$

Figure 3: Forward Gramian derivative map

These graphs can be understood in the context of Algorithmic Differentiation, whereby the forward mode of this derivative map propagates variables up the tree and the adjoint mode propagates variables down the tree and adds (accumulates) the contributions of the relevant variables.

$$\begin{tikzcd}
\delta B_t & & \delta G_t & \delta B_t & & \delta G_t \\
& \delta G_{t_r} \arrow[ul, "\frac{\partial G_{t_r}}{\partial B_t}^T"] \arrow[ur, "\frac{\partial G_{t_r}}{\partial G_t}^T"'] & & & \delta G_{t_l} \arrow[ul, "\frac{\partial G_{t_l}}{\partial B_t}^T"] \arrow[ur, "\frac{\partial G_{t_l}}{\partial G_t}^T"'] &
\end{tikzcd}$$

Figure 4: Adjoint Gramian derivative map

Since we only consider tangent vectors δB_t that are in the horizontal space at x, each extracted component is projected on to $(B_t^{(1,2)})^\perp$. We summarize our results in the following algorithms.

Applying Algorithm 7 to the gradient of $R(\mathbf{B}_t)$,

$$\nabla R(\mathbf{B}_t) = (V_t(I_{k_t} - S_t^{-2})V_t^T),$$

where $G_t = V_t S_t V_t^T$ is the eigenvalue decomposition of G_t, yields the Riemannian gradient of the regularizer. Note that here, we avoid having to compute SVDs of *any* matricizations of the full data $\phi(x)$, resulting in

Algorithm 6 $Dg[\delta \mathbf{B}_t]$

```
Require: Current point $x = (U_t, \mathbf{B}_t)$, horizontal vector $dx = (\delta U_t, \delta \mathbf{B}_t)$
  Compute $(G_t)_{t\in T}$ using (18)
  $\delta G_{\mathrm{t_{root}}} \leftarrow 0$
  for $t \in T \setminus L$, visiting parents before children do
    $\delta G_{t_l} \leftarrow \langle \delta G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)} + 2\langle G_t \times_3 \delta\mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)}$
    $\delta G_{t_r} \leftarrow \langle \delta G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(1,3),(1,3)} + 2\langle G_t \times_3 \delta\mathbf{B}_\mathbf{t}, \mathbf{B}_t\rangle_{(1,3),(1,3)}$
  end for
  return $Dg[\delta \mathbf{B}_t] \leftarrow (\delta G_t)_{t\in T}$
```

Algorithm 7 $Dg^*[\delta G_t]$

```
Require: Current point $x = (U_t, \mathbf{B}_t)$, Gramian variations $(\delta G_t)_{t\in T}$, $\delta G_{\mathrm{t_{root}}} = 0$
  Compute $(G_t)_{t\in T}$ using (18)
  for $t \in T$ do
    $\widetilde{\delta G_t} \leftarrow \delta G_t$
  end for
  for $t \in T \setminus L$, visiting children before parents do
    $\delta \mathbf{B}_t \leftarrow (\widetilde{\delta G_{t_l}} + \widetilde{\delta G_{t_l}}^T) \times_1 G_t \times_3 \mathbf{B}_t + (\widetilde{\delta G_{t_r}} + \widetilde{\delta G_{t_r}}^T) \times_2 G_t \times_3 \mathbf{B}_t$
    if $t \neq \mathrm{t_{root}}$ then
      $\delta \mathbf{B}_t \leftarrow (P^{\perp}_{B^{(1,2)}} \delta \mathbf{B}_t^{(1,2)})_{(1,2)}$
      $\widetilde{\delta G_t} \leftarrow \delta G_t + \langle \widetilde{G_{t_l}} \times_1 \mathbf{B}_t, \mathbf{B}_t\rangle_{(1,2),(1,2)} + \langle \widetilde{G_{t_r}} \times_2 \mathbf{B}_t, \mathbf{B}_t\rangle_{(1,2),(1,2)}$
    end if
  end for
  return $Dg^*[\delta G_t] \leftarrow (\delta \mathbf{B}_t)_{t\in T}$
```

a method which is much faster than other tensor completion methods that require the SVDs on tensors in $\mathbb{R}^{n_1\times n_2 \times \ldots \times n_d}$ [17]. Note that the cost of computing this regularizer $R(\mathbf{B}_t)$ and its gradient are almost negligible compared to the cost of computing the objective and its Riemannian gradient.

Finally, we should also note that the use of this regularizer is not designed to improve the recovery quality of problem instances with a relatively large amount of data and is useful primarily in the case where there is very little data so as to prevent overfitting, as we shall see in the numerical results section.

5.5. Convergence analysis

Our analysis here follows from similar considerations in [31, Sec. 3.6].

Theorem 5. *Let $\{x_i\}$ be an infinite sequence of iterates, with x_i generated at iteration i, generated from Algorithm 4 for the Gramian-regularized objective with $\lambda > 0$*

$$f(x) = \frac{1}{2}\|P_\Omega \phi(x) - b\|_2^2 + \lambda^2 \sum_{t\in T\setminus \mathrm{t}_{root}} \mathrm{tr}(G_t(x)) + \mathrm{tr}(G_t^{-1}(x)).$$

Then $\lim_{i\to\infty} \|\nabla^R f(x_i)\| = 0$.

Proof. To show convergence, we merely need to show that the iterates remain in a sequentially compact set, since any accumulation point of $\{x_i\}$ is a critical point of f, by [1, Thm 4.3.1]. But this follows because by construction, since $f(x_i) \leq f(x_0) := C^2$ for all i. Letting $\mathbf{X}_i := \phi(x_i)$

$$\frac{1}{2}\|P_\Omega \phi(x_i) - b\|_2^2 + \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \mathrm{tr}(G_t(x_i)) + \mathrm{tr}(G_t^{-1}(x_i)) =$$

$$\frac{1}{2}\|P_\Omega \mathbf{X}_i - b\|_2^2 + \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|X_i^{(t)}\|_F^2 + \|(X_i^{(t)})^\dagger\|_F^2 \leq C^2$$

This shows, in particular, that

$$\lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|X_i^{(t)}\|_F^2 \leq C^2 \quad \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|(X_i^{(t)})^\dagger\|_F^2 \leq C^2$$

and therefore we have upper and lower bounds on the maximum and minimum singular values of $X_i^{(t)}$

$$\sigma_{\max}(X_i^{(t)}) \leq \|X_i^{(t)}\|_F \leq C/\lambda \quad \sigma_{\min}^{-1}(X_i^{(t)}) \leq \|(X_i^{(t)})^\dagger\|_F \leq C/\lambda$$

and therefore the iterates X_i stay within the compact set

$$\mathcal{C} = \{\mathbf{X} \in \mathcal{H} : \sigma_{\min}(X_k^{(t)}) \geq \lambda/C, \sigma_{\max}(X_k^{(t)}) \leq C/\lambda, t \in T \setminus \mathrm{t}_{\mathrm{root}}\}.$$

One can show, as a modification of the proof in [42], that $\hat{\phi} : \mathcal{M}/\mathcal{G} \to \mathcal{H}$ is a homeomorphism on to its image, so that $\hat{\phi}^{-1}(C)$ is compact in $\mathcal{M}/\mathcal{G}$. We can introduce a metric on $\mathcal{M}/\mathcal{G}$, which generates the topology on the quotient space, as

$$d(\pi(x), \pi(y)) = \inf_{\mathcal{A}, \mathcal{B} \in \mathcal{G}} \|\theta_\mathcal{A}(x) - \theta_\mathcal{B}(y)\|_T \tag{21}$$

where $\|x - y\|_T = \sum_{t \in L} \|U_t - V_t\|_F + \sum_{t \in T \setminus L} \|\mathbf{B}_t - \mathbf{C}_t\|_F$ is the natural metric on $\mathcal{M}$ and $x = (U_t, \mathbf{B}_t)$, $y = (V_t, \mathbf{C}_t)$.

Note that this pseudo-metric is a metric which generates the topology on $\mathcal{M}/\mathcal{G}$ by [6, Thm 2.1] since $\{\theta_\mathcal{A}\}_{\mathcal{A} \in \mathcal{G}}$ is a group of isometries acting on $\mathcal{M}$ and the orbits of the action are closed by [42, Thm 2]. Note that this metric is equivalent to

$$d(\pi(x), \pi(y)) = \inf_{\mathcal{A} \in \mathcal{G}} \|x - \theta_\mathcal{A}(y)\|_T \tag{22}$$

which is well-defined and equal to (21) since $\|\theta_\mathcal{A}(x) - \theta_\mathcal{B}(y)\|_T = \|x - \theta_{\mathcal{A}^{-1}\mathcal{B}}(y)\|_T$ and $\mathcal{A}, \mathcal{B}$ vary over $\mathcal{G}$. Therefore, if we have a sequence $\{x_i\}$ in $\pi^{-1}(\hat{\phi}^{-1}(C))$, by compactness of $\phi^{-1}(C)$, without loss of generality we have

$$\pi(x_i) \to \pi(y) \in \hat{\phi}^{-1}(C).$$

Then, by the characterization (22), there exists a sequence $\mathcal{A}_i \subset \mathcal{G}$ such that

$$d(x_i, \theta_{\mathcal{A}_i}(y)) \to 0$$

Since $\mathcal{G}$ is compact, there exists a subsequence $\{\mathcal{A}_{i_j}\}$ that converges to $\mathcal{A} \in \mathcal{G}$. It then follows that

$$d(x_{i_j}, \theta_\mathcal{A}(y)) \leq d(x_{i_j}, \theta_{\mathcal{A}_{i_j}}(y)) + d(\theta_{\mathcal{A}_{i_j}}(y), \theta_\mathcal{A}(y)) \to 0 \quad \text{as } j \to \infty$$

And so $\pi^{-1}(\hat{\phi}^{-1}(C))$ is sequentially compact in $\mathcal{M}$. Therefore since the sequence x_k generated by Algorithm 4 stays inside $\pi^{-1}(\hat{\phi}^{-1}(C))$ for all i, a subsequence of x_i converges to some $x \in \pi^{-1}(\hat{\phi}^{-1}(C))$, and so x is a critical point of f. $\square$

6. Numerical examples

To address the challenges of large-scale tensor completion problems, as encountered in exploration seismology, we implemented the approach outlined in this paper in a highly optimized parallel Matlab toolbox entitled HTOpt (available at `https://www.slim.eos.ubc.ca/SoftwareLicensed/license.cgi?package=HTopt` for academic use). Contrary to the HT toolbox [32], whose primary function is performing operations on known HT tensors, our toolbox is designed to solve optimization problems in the HT format such as the seismic tensor completion problem. Our package includes the general optimization on HT manifolds detailed in Algorithm 1 as well as sparsity-exploiting objective & Riemannian gradient in Algorithm 2, implemented in Matlab. We also include a parallel implementation using the Parallel Matlab toolbox for both of these algorithms. All of the following experiments were run on a single IBM x3550 workstation with 2 quad-core Intel 2.6Ghz processors with 16GB of RAM running Linux 2.6.18.

A simplified variation of the experiments below using the seismic data were presented previously in [11]. The experiments in our conference proceedings subsamples sources and use a Conjugate-Gradient method to solve the interpolation problem. In this paper, we use receiver subsampling and our subsequently developed Gauss-Newton and regularization methods to improve the recovery substantially while simultaneously greatly reducing the number of iterations required.

To the best of our knowledge, there is no other existing method which is able to interpolate HT tensors from a fixed sampling set Ω. As such, we compare our Gauss-Newton method with the interpolation scheme detailed in [31], denoted geomCG, for interpolating tensors with missing entries on the Tucker manifold. We have implemented a completely Matlab-based version of geomCG, which does not take advantage of the sparsity of the residual when computing the objective and Riemannian gradient, but uses Matlab's internal calls to LAPACK libraries to compute matrix-matrix products and is much more efficient for this problem. To verify that our implementation of the Tucker-based interpolation scheme is correct, we compare our method to the reference mex implementation for geomCG included in [31]. In the interest of brevity, we refer to the HTOpt package for more details on this front. Since we take advantage of dense linear algebra routines, we find that our Matlab implementation is significantly faster than the mex code of [31] when $K \geq 20$ and $|\Omega|$ is a significant fraction of N^d, as is the case in the examples below.

6.1. Seismic data

We briefly summarize the structure of seismic data in this section. Seismic data is collected via a boat equipped with an airgun and, for our purposes, a 2D array of receivers positioned on the ocean floor. The boat periodically fires a pressure wave in to the earth, which reflects off of subterranean discontinuities and produces a returning wave that is measured at the receiver array. The resulting data volume is five-dimensional, with two spatial source coordinates, denoted x_{src}, y_{src}, two receiver coordinates, denoted x_{rec}, y_{rec}, and time. For these experiments, we take a Fourier transform along the time axis and extract a single 4D volume by fixing a frequency and let $\mathbf{D}$ denote the resulting *frequency slice* with dimensions $n_{src} \times n_{src} \times n_{rec} \times n_{rec}$.

From a practical point of view, the acquisition of seismic data from a physical system only allows us to subsample *receiver* coordinates, i.e., $\Omega = [n_{src}] \times [n_{src}] \times \mathcal{I}$ for some $\mathcal{I} \subset [n_{rec}] \times [n_{rec}]$ with $|\mathcal{I}| < n_{rec}^2$, rather than the standard tensor completion approach, which assumes that $\Omega \subset [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}]$ is random and unstructured. As a result, we use the dimension tree

$$\begin{tikzcd}
 & & & \{x_{src}, x_{rec}, y_{src}, y_{rec}\} \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & \{x_{src}, x_{rec}\} \arrow[dl, dash] \arrow[dr, dash] & & & & \{y_{src}, y_{rec}\} \arrow[dl, dash] \arrow[dr, dash] & \\
\{x_{src}\} & & \{x_{rec}\} & & \{y_{src}\} & & \{y_{rec}\}
\end{tikzcd}$$

for completing seismic data. With this choice, the fully sampled data $\mathbf{D}$ has *quickly* decaying singular values in each matricization $D^{(t)}$ and is therefore represented well in the HT format. Additionally, the subsampled

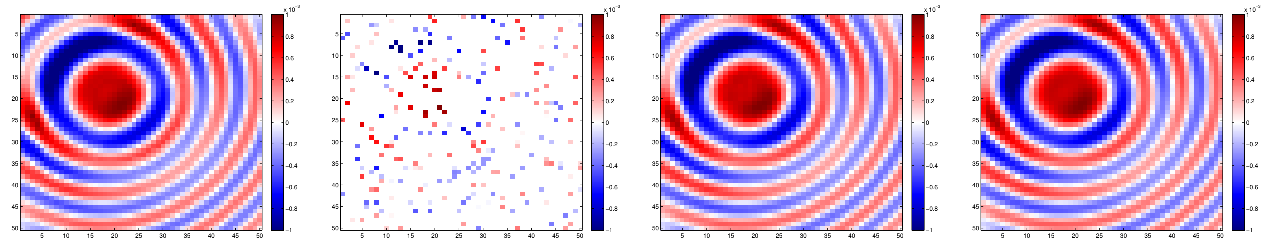

(a) True data (b) Input data $b = P_\Omega \mathbf{X}^*$ (c) HT(30,80) - SNR 30.5 dB (d) geomCG(20)-0 - SNR 29.4 dB

Figure 5: Reconstruction results for 90% missing points, best results for geomCG and HTOpt.

data $P_\Omega \mathbf{D}$ has increased singular values in all matricizations, and is poorly represented as a HT tensor with fixed ranks $\mathbf{k}$ as a result. We examine this effect empirically in [11] and note that this data organization is used in [14] in the context of solution operators of the wave equation. Although this approach is limited to considerations of seismic data, for larger dimensions/different domains, potentially the method of [5] can choose an appropriate dimension tree automatically. In the next section, we also include the case when $\Omega \subset [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}]$, i.e. the "missing points" scenario, to demonstrate the added difficulty of the "missing receivers" case described above.

6.2. Single reflector data

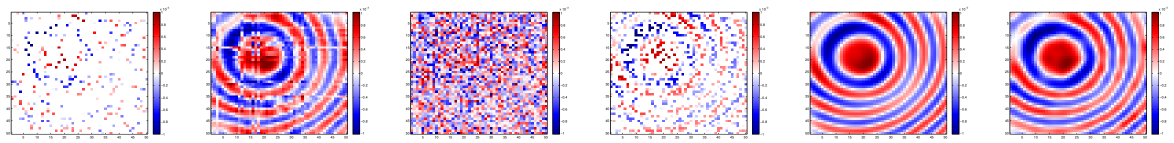
For this data set, we generate data from a very simple seismic model consisting of two horizontal layers with a moderate difference in wavespeed and density between them. We generate this data with $n_{src} = n_{rec} = 50$ and extract a frequency slice at 4.21Hz, rescaled to have unit norm.

We consider the two sampling scenarios discussed in the previous section: we remove random points from the tensor, with results shown in Figure 5, and we remove random receivers from the tensor, with results shown in Figure 6. Here geomCG(r_{leaf}) - w denote the Tucker interpolation algorithm with rank r_{leaf} in each mode and w rank continuation steps, i.e., the approach proposed in [31]. We also let HT($r_{\text{leaf}}, r_{x_{\text{src}}x_{\text{rec}}}$) denote the HT interpolation method with rank r_{leaf} as in the Tucker interpolation and rank $r_{x_{\text{src}}x_{\text{rec}}}$ as the internal rank for the dimension tree. As is customary in the seismic literature, we measure recovery quality in terms of SNR, namely

$$\text{SNR}(\mathbf{X}, \mathbf{D}) = -20 \log_{10} \left(\frac{\|\mathbf{X}_{\Omega^c} - \mathbf{D}_{\Omega^c}\|}{\|\mathbf{D}_{\Omega^c}\|} \right) \text{dB},$$

where $\mathbf{X}$ is our interpolated signal, $\mathbf{D}$ is our reference solution, and $\Omega^c = [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}] \setminus \Omega$. As we can see in Figure 5, the HT formulation is able to take advantage of low-rank separability of the seismic volume to produce a much higher quality solution than that of the Tucker tensor completion. The rank continuation scheme does not seem to be improving the recovery quality of the Tucker solution to the same degree as in [31], although it does seem to mitigate some of the overfitting errors for geomCG(30). We display slices for fixed source coordinates and varying receiver coordinates in Figure 5 for randomly missing points and Figure 6 for randomly missing receivers. By exploiting the low-rank structure of the HT format compared to the Tucker format, we are able to achieve much better results than Tucker tensor completion, especially for the realistic case of missing receiver samples.

In all instances for these experiments, the HT tensor completion outperforms the conventional Tucker approach both in terms of recovery quality and recovery speed. We note that geomCG does not scale as well computationally as our HT algorithm for $d > 3$, as the complexity analysis in [31] predicts. As such, we only consider the HT interpolation for the next sections, where we will solve the tensor completion problem for much larger data volumes.



(a) 90% missing receivers (b) HT(20,20) - SNR 7.04 dB (c) geomCG(20)-0 - SNR -1.92 dB (d) 70% missing receivers (e) HT(20,20) - SNR 20.4 dB (f) geomCG(30)-5 - SNR 16.8 dB

Figure 6: Reconstruction results for sampled receiver coordinates, best results for geomCG and HTOpt. *(a-c)* 90% missing receivers. *(d-f)*: 70% missing receivers.

	Single reflector data - sampling percentage (missing points)					
	10%		30%		50%	
	SNR [dB]	time [s]	SNR [dB]	time [s]	SNR [dB]	time [s]
geomCG(20) - 0	28.5	1023	30.5	397	30.7	340
geomCG(30) - 0	-6.7	1848	21.8	3621	31.5	2321
geomCG(30) - 5	16.1	492	13.8	397	15.5	269
HTOpt(20,60)	30.1	83	30.4	59	30.4	57
HTOpt(20,80)	30.3	121	30.8	75	30.8	53
HTOpt(30,80)	**31.6**	196	**32.9**	133	**33.1**	114

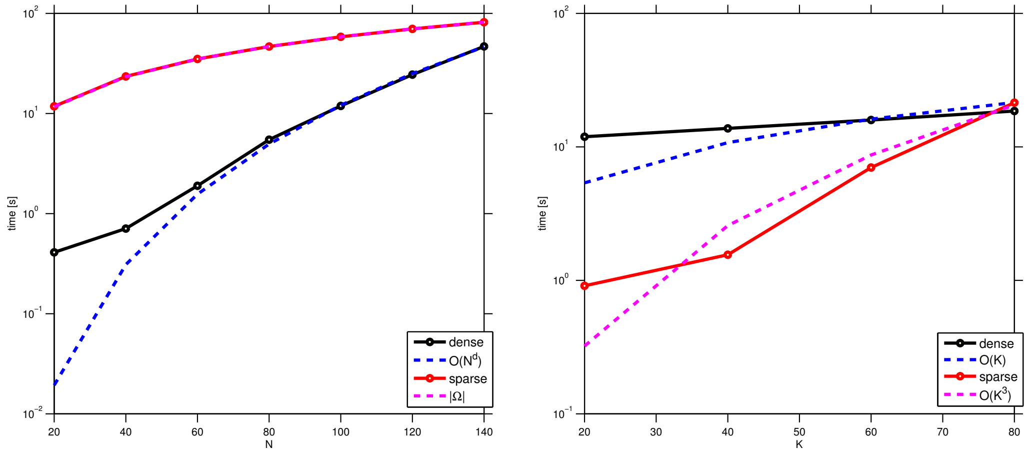
Table 1: Reconstruction results for single reflector data - missing points - mean SNR over 5 random training sets

6.3. Performance

We investigate the empirical performance scaling of our approach as N, d, K, and $|\Omega|$ increase, as well as the number of processors for the parallel case, in Figure 7 and Figure 8. Here we denote the use of Algorithm 1 as the "dense" case and Figure 2 as the "sparse" case. We run our optimization code in Steepest Descent mode with a single iteration for the line search, and average the running time over 10 iterations and 5 random problem instances. Our empirical performance results agree very closely with the theoretical complexity estimates, which are $O(N^d K)$ for the dense case and $O(|\Omega| d K^3)$ for the sparse case. Our parallel implementation for the sparse case scales very close to the theoretical time $O(1/\#$ processors).

	Single reflector data - sampling percentage (missing receivers)					
	10%		30%		50%	
	SNR [dB]	time [s]	SNR [dB]	time [s]	SNR [dB]	time [s]
geomCG(20) - 0	-5.1	899	9.9	898	18.5	891
geomCG(30) - 0	-3.6	1796	-4.7	1834	6.1	1802
geomCG(30) - 5	-6.4	727	11.1	670	14.2	356
HTOpt(20,20)	**6.1**	111	**19.8**	101	20.1	66
HTOpt(30,20)	2.8	117	18.1	109	19.8	94
HTOpt(30,40)	0.0	130	13.4	126	**21.6**	108

Table 2: Reconstruction results for single reflector data - missing receivers - mean SNR over 5 random training test sets



Fixed K, d, varying N, $|\Omega| = 1000N$ Fixed $N, d, |\Omega|$, varying K

Figure 7: Dense & sparse objective, gradient performance.

6.4. Synthetic BG Compass data

This data set was provided to us by BG and consists of 5D data generated from an unknown synthetic model. Here $n_{src} = 68$ and $n_{rec} = 401$ and we extract frequency slices at 4.86 Hz, 7.34 Hz, and 12.3 Hz. On physical grounds, we expect a slower decay of the singular values at higher frequencies and thus the problem is much more difficult at 12.3 Hz compared to 4.86 Hz.

At these frequencies, the data has relatively low spatial frequency content in the receiver coordinates, and thus we subsample the receivers by a factor of 2 to $n_{rec} = 201$, for the purposes of speeding up the overall computation and ensuring that the intermediate vectors in the optimization are able to fit in memory. Our overall data volume has dimensions $\mathbf{D} \in \mathbb{R}^{68\times 68\times 201\times 201}$.

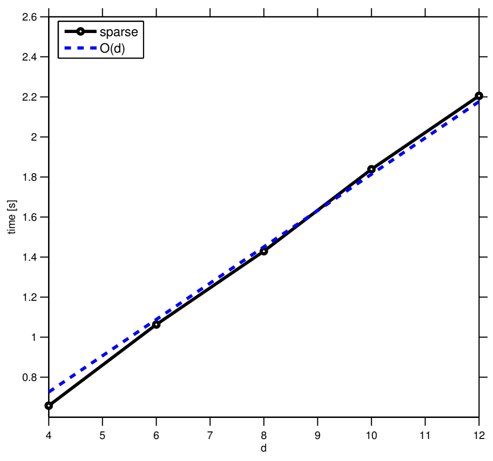
We randomly remove varying amounts of receivers from this reduced data volume and interpolate using 50 iterations of the GN method discussed earlier. We display several recovered slices for fixed source coordinates and varying receiver coordinates (so-called *common source gathers* in seismic terminology) in Figure 9.

We summarize our recovery results for tensor completion on these data sets from missing receivers in Table 3 and the various recovery parameters we use in Table 4. When the subsampling rate is extremely high (90% missing receivers in these examples), the recovery can suffer from overfitting issues, which leads to spurious artifacts in the recovered volume and lower SNRs overall. Using the Gramian-based regularization method discussed earlier, we can mitigate some of those artifacts and boost recovered SNRs, as seen in Figure 10.

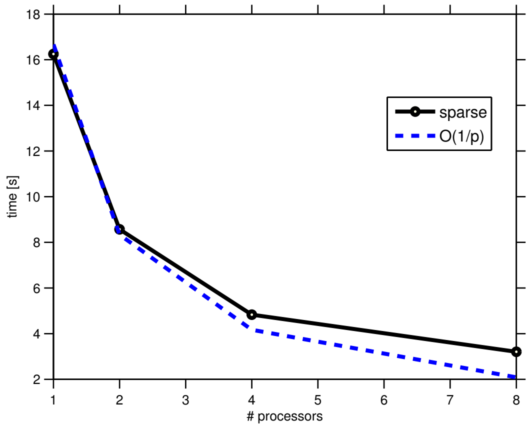
7. Conclusions and discussion

In this work we have developed the algorithmic components to solve optimization problems on the manifold of fixed-rank Hierarchical Tucker tensors. By exploiting this manifold structure, we solve the tensor completion problem where the tensors of interest exhibit low-rank behavior. Our algorithm is computationally efficient because we mostly rely on operations on the small HT parameter space. The manifold optimization itself guarantees that we do not run into convergence issues, which arise when we ignore the quotient structure of the HT format. Our application of this framework to seismic examples confirms the validity of our new approach and outperforms existing Tucker-based approaches for large data volumes. To stabilize the recovery for high subsampling ratios, we introduced an additional regularization term that exploits properties of the Gramian matrices without the need to compute SVDs in the ambient space.

While the method clearly performs well on large-scale problems, there are still a number of theoretical questions regarding the performance of this approach. In particular, the generalization of matrix completion



Fixed $N, K, |\Omega|$, varying d



Fixed N, K, d, Ω, varying number of processors

Figure 8: Sparse objective, gradient performance.

recovery guarantees to the HT format remains an open problem. As in many alternative approaches to matrix/tensor completion, the selection of the rank parameters and regularization parameters remain challenging both theoretically and from a practical point of view. However, the paper clearly illustrates that the HT format is a viable option to represent and complete high-dimensional data volumes in a computationally feasible manner.

8. Acknowledgements

We would like to thank the sponsors of the SINBAD consortium for their continued support. We would also like to thank the BG Group for providing us with the Compass data set.

Frequency	% Missing	Train SNR (dB)	Test SNR (dB)	Runtime (s)
4.86 Hz	25%	21.2	21	4033
	50%	21.3	20.9	4169
	75%	21.5	19.9	4333
	90%	19.9	10.4	4679
	90%*	20.8*	13.0*	5043
7.34 Hz	25%	17.3	17.0	4875
	50%	17.4	16.9	4860
	75%	17.7	16.5	5422
	90%	16.6	9.82	4582
	90%*	16.6*	10.5*	4947
12.3 Hz	25%	14.9	14.2	5950
	50%	15.2	13.8	7083
	75%	15.8	9.9	7387
	90%	13.9	5.39	4578
	90%*	14*	6.5*	4966

Table 3: HT Recovery results - randomly missing receivers. Starred quantities are computed with regularization.

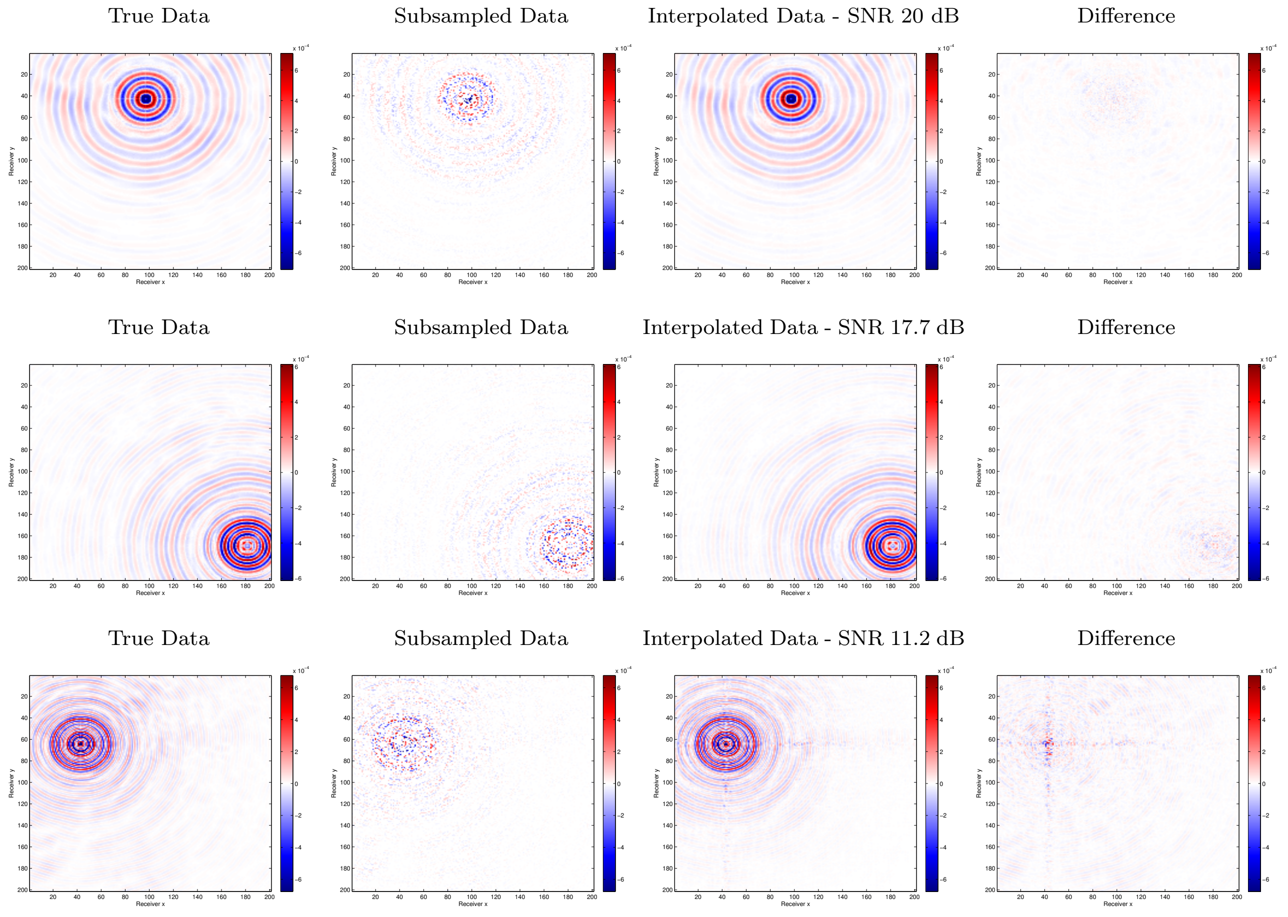


Figure 9: 75% missing receivers, fixed source coordinates. *Top:* 4.68 Hz, *Middle:* 7.34 Hz, *Bottom:* 12.3 Hz.

Frequency	$k_{x_{src}x_{rec}}$	$k_{x_{src}}$	$k_{x_{rec}}$	HT-SVD SNR (dB)
4.86 Hz	150	68	120	21.1
7.34 Hz	200	68	120	17.0
12.3 Hz	250	68	150	13.9

Table 4: HT parameters for each data set and the corresponding SNR of the HT-SVD approximation of each data set. The 12.3 Hz data is of much higher rank than the other two data sets and thus is much more difficult to recover.

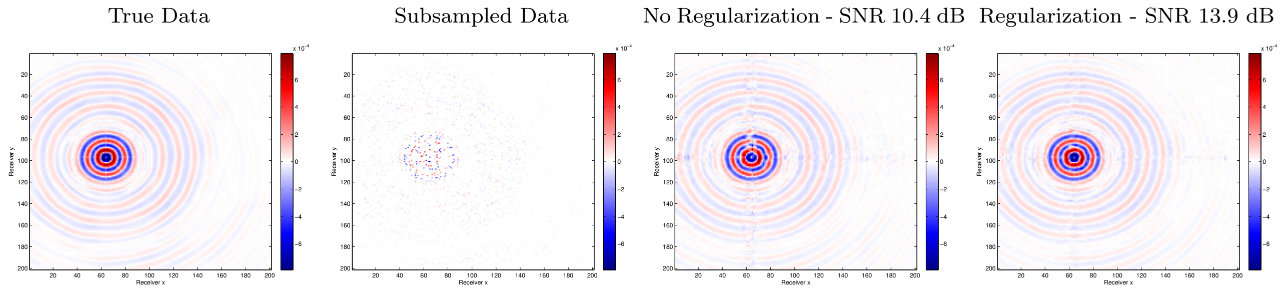


Figure 10: Regularization reduces some of the spurious artifacts and reduces overfitting in the case where there is very little data. 4.86 Hz data, 90% missing receivers.

Appendix A. Adjoint multilinear operators

To derive expressions for the adjoint derivatives, we first consider the general multilinear product

$$A_1 \times_1 A_2 \times_2 \dots A_d \times_d \mathbf{B} \in \mathbb{R}^{m_1 \times \dots \times m_d}$$

with $A_i \in \mathbb{R}^{m_i \times n_i}$ and $\mathbf{B} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$. Now, let P_i be the linear operator that fixes each A_j for $j \neq i$ in the above expression, i.e.,

$$\begin{aligned} P_i &: \mathbb{R}^{m_i \times n_i} \to \mathbb{R}^{m_1 \times \dots \times m_d} \\ P_i(C) &:= A_1 \times_1 A_2 \times_2 \dots A_{i-1} \times_{i-1} C \times_i A_{i+1} \times_{i+1} \dots A_d \times_d \mathbf{B}. \end{aligned}$$

In matricized form, this operator can be written as

$$(P_i(C))^{([d]\setminus i)} = CB^{(i)}(A_d^T \otimes A_{d-1}^T \otimes \dots \otimes A_{i+1}^T \otimes A_{i-1}^T \otimes \dots A_1^T)$$

Taking the inner product of the matrix $(P_i(C)^{([d]\setminus i)}$ and a tensor $\mathbf{Y}$ matricized along the ith mode yields

$$\begin{aligned} \langle P_i(C)^{[d]\setminus i}, Y^{(i)} \rangle &= \operatorname{tr}(CB^{(i)}(A_d^T \otimes A_{d-1}^T \otimes \dots \otimes A_{i+1}^T \otimes A_{i-1}^T \otimes \dots A_1^T)(Y^{(i)})^T) \\ &= \langle C, Z \rangle \end{aligned}$$

where $Z = Y^{(i)}(A_d \otimes A_{d-1} \otimes \dots \otimes A_{i+1} \otimes A_{i-1} \otimes \dots A_1)(B^{(i)})^T$.

We note that $(A_d \otimes A_{d-1} \otimes \dots \otimes A_{i+1} \otimes A_{i-1} \otimes \dots A_1)(B^{(i)})^T$ is the matricized form of

$$\mathbf{W} = A_1^T \times_1 A_2^T \times_2 \dots A_{i-1}^T \times_{i-1} I_{m_i} \times_i A_{i+1}^T \times_{i+1} \dots A_d^T \times_d \mathbf{B}$$

along the modes $[d] \setminus i$, and that $Y^{(i)}W^{([d]\setminus i)}$ is the tensor contraction $\langle \mathbf{Y}, \mathbf{W} \rangle_{([d]\setminus i),([d]\setminus i)}$. It follows that the adjoint of the operator $P_i(C)$ in the standard Euclidean inner product is given by

$$P_i^*(\mathbf{Y}) = \langle A_1^T \times_1 A_2^T \times_2 \dots A_{i-1}^T \times_{i-1} I_{m_i} \times_i A_{i+1}^T \times_{i+1} \dots A_d^T \times_d \mathbf{Y}, \mathbf{B} \rangle_{([d]\setminus i),([d]\setminus i)}.$$

Likewise, for the linear operator

$$P_{\mathbf{B}}(\mathbf{C}) := A_1 \times_1 \dots A_d \times_d \mathbf{C}$$

we find that its adjoint is given by

$$P_{\mathbf{B}}^*(\mathbf{Y}) = A_1^T \times_1 \dots A_d^T \times_d \mathbf{Y}.$$

Appendix B. Proof of Proposition 3

Proof. It is easy to see that the first point in Definition 7 is satisfied, since for $X \in \mathrm{St}(n,p)$, $\mathrm{qf}(X) = X$

Let $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$ and $\eta = (\delta U_t, \delta \mathbf{B_t}) \in \mathcal{T}_x \mathcal{M}$. To avoid notational overload, we use the slight abuse of notation that $B_t := B_t^{(1,2)}$ for $t \neq \mathrm{t_{root}}$.

Let $s \in [0,t) \mapsto x(s)$ be a curve in the parameter space $\mathcal{M}$ with $x(0) = x$ and $x'(0) = \eta$ and $x(s) = (U_t(s), B_t(s))$ and $x'(s) = (\delta U_t(s), \delta B_t(s))$.

Then we have that, in Kronecker form,

$$DR_x(0_x)[\eta] = \begin{cases} \frac{d}{ds}\,\mathrm{qf}(x(s)_t)\big|_{s=0} & \text{if } t \in L \\ \frac{d}{ds}\,\mathrm{qf}((R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)\big|_{s=0} & \text{if } t \notin \mathrm{t_{root}} \cup L \\ \frac{d}{ds}(R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)\big|_{s=0} & \text{if } t = \mathrm{t_{root}} \end{cases}$$

The fact that $DR_x(0_x)[\eta]_t = \delta U_t$ for $t \in L$ follows from Example 8.1.5 in [1].

To compute $DR_x(0_x)[\eta]_t$ for $t \notin L \cup \mathrm{t_{root}}$, we first note the formula from [1]

$$D\,\mathrm{qf}(Y)[U] = \mathrm{qf}(Y)\rho_{\mathrm{skew}}(\mathrm{qf}(Y)^T U (\mathrm{qf}(Y)^T Y)^{-1}) + (I - \mathrm{qf}(Y)\,\mathrm{qf}(Y)^T) U (\mathrm{qf}(Y)^T Y)^{-1} \tag{B.1}$$

where $Y \in \mathbb{R}_*^{n \times k}$, $U \in T_Y \mathbb{R}_*^{n\times k} \simeq \mathbb{R}^{n \times k}$ and $\mathrm{qf}(Y)$ is the Q-factor of the QR-decomposition of Y.

Therefore, if we set $Z(s) = (R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)$, where $R_t(s)$ is the R-factor of the QR-decomposition of the matrix associated to node t, we have

$$Z'(0) = [(R'_{t_r}(0) \otimes I_{k_l}) + (I_{k_r} \otimes R'_{t_l}(0)]B_t + \delta B_t$$

As a result of the discussion in Example 8.1.5 in [1], since $R_t(0) = I_{k_t}$ we have that

$$R'_t(0) = \begin{cases} \rho_{UT}(U_t^T \delta U_t) & \text{for } t \in L \\ \rho_{UT}(B_t^T \delta B_t) & \text{for } t \notin L \cup \mathrm{t_{root}} \end{cases}$$

where $\rho_{UT}(A)$ is the projection onto the upper triangular term of the unique decomposition of a matrix into the sum of a skew-symmetric term and an upper triangular term.

Since $U_t \in \mathrm{St}(n_t, k_t)$ and $B_t \in \mathrm{St}(k_{t_l} k_{t_r}, k_t)$, in light of the fact that for $X \in St(n,k)$,

$$T_X St(n,k) = \{X\Omega + X^\perp K : \Omega = -\Omega^T\},$$

then $X^T \delta X$ is skew symmetric, for any tangent vector δX, which implies that $\rho_{UT}(X^T \delta X)$ is zero.

It follows that $R'_t(0) = 0$ for all $t \in T \setminus \mathrm{t_{root}}$, and therefore

$$Z'(0) = \delta B_t$$

from which we immediately obtain

$$DR_x(0_x)[\eta]_t = \delta B_t \quad \text{for } t \notin L \cup \mathrm{t_{root}}$$

A similar approach holds when $t = \mathrm{t_{root}}$, and therefore, $R_x(\eta)$ is a retraction on $\mathcal{M}$. $\square$

Appendix C. Square-root based retraction

Another straightforward projection onto the orthonormal parameter space is immediate from the remark that for a general full-rank $n \times p$ matrix X, with $n > p$, the matrix $X(X^T X)^{-1/2}$ is an orthonormal basis for the column space of X. In Algorithm 8, we only need to compute the eigenvalue decomposition of a $k_t \times k_t$ matrix, which may be done more efficiently than computing the QR-factorization of a $k_{t_l} k_{t_r} \times k_t$ matrix in some instances.

Algorithm 8 Square-root-based orthogonalization

```
Require: $x = (U_t, \mathbf{B}_t)$ unorthogonalized
  for $t \in L$ do
    $M_t = U_t^T U_t$
    Compute the eigenvalue decomposition of $M_t$, $M_t = V_t D_t V_t^T$
    $U'_t \leftarrow U_t M_t^{-1/2}$
  end for
  for $t \in T \setminus L$, visiting children before their parents do
    $C_t \leftarrow (M_{t_l}^{1/2} \times_1 M_{t_r}^{1/2} \times_2 \mathbf{B}_t)^{(1,2)}$
    if $t = \mathrm{t_{root}}$ then
      $B'_t \leftarrow (C_t)_{(1,2)}$
    else
      Compute the eigenvalue decomposition of $C_t^T C_t$, $C_t^T C_t = V_t D_t V_t^T$
      $B'_t \leftarrow (C_t (C_t^T C_t)^{-1/2})_{(1,2)}$
    end if
  end for
  return $x' = (U'_t, B'_t)$ in OHT
```

References

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

[2] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[3] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. `http://www.sandia.gov/~tgkolda/TensorToolbox/`, January 2012.

[4] J. Ballani, L. Grasedyck, and M. Kluge. Black box approximation of tensors in hierarchical tucker format. *Linear Algebra and its Applications*, 2011.

[5] Jonas Ballani and Lars Grasedyck. Tree adaptive approximation in the hierarchical tensor format. *Preprint*, 141, 2013.

[6] Francesca Cagliari, Barbara Di Fabio, and Claudia Landi. The natural pseudo-distance as a quotient pseudo-metric, and applications. *AMS Acta, Universita di Bologna*, 3499, 2012.

[7] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[8] E.J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[9] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[10] J. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35:283–319, 1970. 10.1007/BF02310791.

[11] Curt Da Silva and Felix J. Herrmann. Hierarchical tucker tensor optimization - applications to tensor completion. In *10th international conference on Sampling Theory and Applications (SampTA 2013)*, pages 384–387, Bremen, Germany, July 2013.

[12] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[13] V. De Silva and L.H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[14] Laurent Demanet. *Curvelets, Wave Atoms, and Wave Equations.* PhD thesis, California Institute of Technology, 2006.

[15] M. P. Friedlander E. van den Berg. Spot – a linear-operator toolbox.

[16] Lars Eldén and Berkant Savas. A newton-grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM Journal on Matrix Analysis and applications*, 31(2):248–271, 2009.

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

[18] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

[19] Lars Grasedyck, Melanie Kluge, and Sebastian Krämer. Alternating directions fitting (adf) of hierarchical low rank tensors. *Preprint*, 149, 2013.

[20] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[21] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.

[22] R.A. Harshman. Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. 1970.

[23] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

[24] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120(4):701–731, 2012.

[25] Bo Huang, Cun Mu, Donald Goldfarb, and John Wright. Provable low-rank tensor recovery. 2014.

[26] B.N. Khoromskij. Tensor-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 2011.

[27] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

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

[29] N. Kreimer and M.D. Sacchi. Tensor completion via nuclear norm minimization for 5d seismic data reconstruction. In *SEG Technical Program Expanded Abstracts 2012*, pages 1–5. Society of Exploration Geophysicists, 2012.

[30] Nadia Kreimer and Mauricio D Sacchi. A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation. *Geophysics*, 77(3):V113–V122, 2012.

[31] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by riemannian optimization. 2013.

[32] D. Kressner and C. Tobler. htucker - a matlab toolbox for tensors in hierarchical tucker format. `http://sma.epfl.ch/~anchpcommon/publications/htucker.pdf`, 2013.

[33] Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical Approximation by Hierarchical Tucker and Tensor-Train Tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, April 2013.

[34] B. Mishra and R. Sepulchre. R3mc: A riemannian three-factor algorithm for low-rank matrix completion. *arXiv.org*, June 2013.

[35] Cun Mu, Bo Huang, John Wright, and Donald Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. *arXiv preprint arXiv:1307.5870*, 2013.

[36] Samet Oymak, Amin Jalali, Maryam Fazel, Yonina C Eldar, and Babak Hassibi. Simultaneously structured models with application to sparse and low-rank matrices. *arXiv preprint arXiv:1212.3753*, 2012.

[37] Z J Shi and J Shen. New Inexact Line Search Method for Unconstrained Optimization. *Journal of Optimization Theory and Applications*, 127(2):425–446, November 2005.

[38] Zhen-Jun Shi. Convergence of line search methods for unconstrained optimization. *Applied Mathematics and Computation*, 157(2):393–405, 2004.

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

[40] C. Tobler. *Low Rank Tensor Methods for Linear Systems and Eigenvalue Problems*. PhD thesis, ETH Zürich, 2012.

[41] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 33(2):639–652, 2012.

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