Riemannian Optimization for Solving High-Dimensional Problems with Low-Rank Tensor Structure

THÈSE N$^{\text{O}}$ 6958 (2016)

PRÉSENTÉE LE 11 AVRIL 2016
À LA FACULTÉ DES SCIENCES DE BASE
ALGORITHMES NUMÉRIQUES ET CALCUL HAUTE PERFORMANCE - CHAIRE CADMOS
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Michael Maximilian STEINLECHNER

acceptée sur proposition du jury:

Prof. K. Hess Bellwald, présidente du jury
Prof. D. Kressner, directeur de thèse
Prof. A. Uschmajew, rapporteur
Prof. B. Vandereycken, rapporteur
Prof. F. Nobile, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

Acknowledgements

This thesis is based on my research conducted at the EPF Lausanne from June 2012 till December 2015 and during a research stay at Princeton University from February 2014 till August 2014.

Over the last years, many wonderful people have supported me and contributed to the research presented here. For this, I am grateful. It would not have been possible without you.

First of all, I would like to thank my thesis advisor Prof. Daniel Kressner for providing me with this great opportunity and leading me to this very active area of computational linear algebra. Daniel, it was a great experience and I learned a lot from you. I continue to be impressed by your focus and efficiency, in particular when writing scientific texts.

I am also very thankful to Prof. Bart Vandereycken for introducing me to manifold optimization and being a great mentor. Bart, I would like to thank you for giving me the possibility to visit you in Princeton. It was an inspiring and fun half year with you in Fine Hall and I always enjoyed our scientific and non-scientific discussions.

Likewise, I would like to thank Prof. André Uschmajew for being a referee of my thesis. André, it was a great time working together with you and having you as an office mate during your postdoc at EPFL.

I would also like to express my gratitude to the Prof. Fabio Nobile for his detailed and constructive remarks and Prof. Kathryn Hess Bellwald for agreeing to serve as a jury president.

Furthermore, I want to thank the previous and current PhD students of our group, Ana, Cedric, Christine, Francesco, Francisco, Meiyue and Petar as well as my office mates Jonas and Robert for the nice atmosphere and the many coffee breaks. I really enjoyed the time with you and hope that we will stay in touch.

Financial support by the Swiss National Science Foundation under the SNSF research module *Riemannian optimization for solving high-dimensional problems with low-rank tensor techniques* within the ProDoc *Efficient Numerical Methods for Partial Differential Equations* is thankfully acknowledged.

Last but not least, I would like to thank Elke and my family for their support over the years.

Lausanne, the 14th of March 2016.

▶ Zusammenfassung

In dieser Arbeit untersuchen wir Riemannsche Lösungsverfahren für hochdimensionale Optimierungsprobleme mit Niedrigrangstruktur. Der Begriff der Hochdimensionalität bezieht sich hierbei auf den Suchraum, während die Zielfunktion skalarwertig ist. Probleme solcher Art treten zum Beispiel bei der Rekonstruktion von hochdimensionalen Datensets oder der Lösung von parameterabhängigen partiellen Differentialgleichungen auf.

Da die Anzahl der Freiheitsgrade exponentiell mit der Anzahl der Dimensionen ansteigt, der sogenannte *Fluch der Dimension*, ist der Rechenaufwand für eine direkte Lösung des Optimierungsproblems nur für niedrigdimensionale Probleme handhabbar. Durch die Annahme von Niedrigrangstruktur in der Lösung erhalten wir ein Optimierungsproblem mit Nebenbedingung, welches deutlich weniger Freiheitsgrade aufweist.

Dieses Optimierungsproblem mit Nebenbedingung kann als Optimierung auf einer Mannigfaltigkeit interpretiert werden. Hierbei betrachten wir die Niedrigrangdarstellungen im sogenannten Tucker- oder Tensor-Train-Format, welche eine glatte, eingebettete Untermannigfaltigkeitsstruktur besitzen. Die glatte Struktur erlaubt es uns, effiziente gradientenbasierte Optimierungsverfahren herzuleiten.

Im Speziellen entwickeln wir ein Riemannsches Verfahren der konjugierten Gradienten, um das *Tensor-Completion*-Problem zu lösen. Hierbei versuchen wir, ein hochdimensionales Datenset zu rekonstruieren, bei dem der Grossteil der Einträge unbekannt ist.

Für die Lösung von linearen Gleichungssystemen zeigen wir Wege auf, um den Riemannschen Gradienten vorzukonditionieren und Informationen zweiter Ordnung im Rahmen eines approximativen Newtonverfahrens einfliessen zu lassen.

Zu guter Letzt beschreiben wir ein vorkonditioniertes alternierendes Optimierungsverfahren mit Unterraumkorrektur zur Lösung hochdimensionaler Eigenwertprobleme.

Schlagwörter. Fluch der Dimensionen, Riemannsche Optimierung, Niedrigrankstruktur, Tucker-Format, Tensor-Train-Format, Vorkonditionierung

▶ Abstract

In this thesis, we present a Riemannian framework for the solution of high-dimensional optimization problems with an underlying low-rank tensor structure. Here, the high-dimensionality refers to the size of the search space, while the cost function is scalar-valued. Such problems arise, for example, in the reconstruction of high-dimensional data sets and in the solution of parameter dependent partial differential equations.

As the degrees of freedom grow exponentially with the number of dimensions, the so-called *curse of dimensionality*, directly solving the optimization problem is computationally unfeasible even for moderately high-dimensional problems. We constrain the optimization problem by assuming a low-rank tensor structure of the solution; drastically reducing the degrees of freedom.

We reformulate this constrained optimization as an optimization problem on a manifold using the smooth embedded Riemannian manifold structure of the low-rank representations of the Tucker and tensor train formats. Exploiting this smooth structure, we derive efficient gradient-based optimization algorithms.

In particular, we propose Riemannian conjugate gradient schemes for the solution of the tensor completion problem, where we aim to reconstruct a high-dimensional data set for which the vast majority of entries is unknown.

For the solution of linear systems, we show how we can precondition the Riemannian gradient and leverage second-order information in an approximate Newton scheme.

Finally, we describe a preconditioned alternating optimization scheme with subspace correction for the solution of high-dimensional symmetric eigenvalue problems.

Keywords. Curse of dimensionality, Riemannian optimization, low-rank structure, Tucker format, Tensor Train, preconditioning

▶ Contents

Acknowledgements **1**

Zusammenfassung **3**

Abstract **5**

1 Introduction **11**
- 1.1 High-dimensional problems 12
- 1.2 Riemannian optimization 14
- 1.3 Contributions of this thesis 14

2 Riemannian Optimization **17**
- 2.1 Smooth manifolds 17
- 2.2 Tangent space and derivatives 18
- 2.3 Submanifolds as subsets of manifolds 20
- 2.4 Riemannian manifolds 21
- 2.5 Distances on manifolds and geodesics 22
- 2.6 The exponential map and retractions 23
- 2.7 Parallel and vector transport 25
- 2.8 Line-search algorithms on manifolds 27
  - 2.8.1 Approximate line-search 28
  - 2.8.2 Riemannian steepest descent 29
  - 2.8.3 Riemannian conjugate gradients 29
  - 2.8.4 Convergence of line-search algorithms 30

3 Tensors of Fixed Multilinear Rank **33**
- 3.1 Tensors: Notation and basic properties 33
  - 3.1.1 Reshaping operations 33
  - 3.1.2 Operations on tensors 34
- 3.2 Multilinear rank and the Tucker format 35
- 3.3 Operations on Tucker tensors 37
  - 3.3.1 Reorthogonalization 37
  - 3.3.2 Addition 37
  - 3.3.3 Inner product and norm 38
  - 3.3.4 Hadamard product 39
  - 3.3.5 Rank truncation and higher-order SVD 39
- 3.4 Manifold structure 41
- 3.5 The tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ 46
  - 3.5.1 Projection onto the tangent space 47
  - 3.5.2 Tucker representation of tangent vectors 49
  - 3.5.3 Inner product of two tangent vectors 50

3.6 Retraction . . . 50
3.7 Vector transport . . . 53
3.8 Linear operators acting on Tucker tensors . . . 53

4 Tensors of Fixed Tensor Train Rank **55**
4.1 Tensor train rank and format . . . 55
4.2 Operations on TT tensors . . . 58
  4.2.1 Orthogonalization . . . 58
  4.2.2 Addition . . . 59
  4.2.3 Inner product and norm . . . 59
  4.2.4 Hadamard product . . . 60
  4.2.5 Rank truncation and TT-SVD . . . 61
4.3 Manifold of TT tensors of fixed rank . . . 62
4.4 The tangent space . . . 62
  4.4.1 Projection onto the tangent space . . . 63
  4.4.2 A different parametrization of the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ . . . 65
  4.4.3 TT representation of tangent vectors . . . 66
  4.4.4 Inner product of two tangent vectors . . . 67
4.5 Retraction . . . 67
4.6 Vector transport . . . 68
4.7 Linear operators acting on TT tensors . . . 69

5 Application 1: Tensor Completion **73**
5.1 A Riemannian tensor completion algorithm . . . 74
  5.1.1 Linearized line search . . . 76
  5.1.2 Stopping criteria . . . 76
  5.1.3 Convergence analysis . . . 77
5.2 Tucker case . . . 78
  5.2.1 Evaluation of the cost function and the gradient . . . 79
  5.2.2 Calculation of the new iterate . . . 79
  5.2.3 Computational complexity . . . 80
5.3 Numerical experiments for the Tucker case . . . 80
  5.3.1 Scaling of the algorithm . . . 81
  5.3.2 Reconstruction of synthetic data sets . . . 82
  5.3.3 Applications . . . 84
  5.3.4 Comparison of rank adaptation strategies . . . 88
5.4 Tensor train case . . . 92
  5.4.1 Evaluation of the cost function and the gradient . . . 92
  5.4.2 Calculation of the new iterate . . . 93
  5.4.3 Computational complexity . . . 94
  5.4.4 Adaptive rank adjustment . . . 94
5.5 Numerical experiments for the TT case . . . 96
  5.5.1 Scaling of the algorithm . . . 97
  5.5.2 Comparison to an alternating linear scheme . . . 97
  5.5.3 Reconstruction of synthetic data sets . . . 99
  5.5.4 Interpolation of high-dimensional functions . . . 100

5.5.5 Parameter-dependent PDE 102
5.6 Conclusion 104

6 Application 2: Linear Systems 105
6.1 First-order Riemannian optimization and preconditioning 107
6.1.1 Riemannian gradient descent 108
6.1.2 Truncated preconditioned Richardson iteration 108
6.2 Riemannian optimization using a quadratic model 110
6.2.1 Approximate Newton method 110
6.2.2 The approximate Riemannian Hessian in the Tucker case . . 111
6.2.3 The approximate Riemannian Hessian in the TT case 113
6.2.4 Connection to ALS 119
6.3 Numerical experiments 120
6.3.1 Test case 1: Newton potential 120
6.3.2 Test case 2: Anisotropic diffusion equation 120
6.3.3 Results for the Tucker format 121
6.3.4 Results for the TT format 124
6.3.5 Rank-adaptivity 126
6.4 Conclusion 127

7 Application 3: Eigenvalue Problems 129
7.1 Extraction of two neighboring cores 130
7.2 The block TT format for a collection of vectors 130
7.3 Trace minimization in the block TT format 132
7.3.1 Alternating optimization 132
7.3.2 Core enrichment based on gradient information 133
7.3.3 Convergence analysis 140
7.3.4 Construction of local preconditioners 141
7.4 Numerical experiments 142
7.4.1 Setting 1: One eigenvalue for the Newton potential 143
7.4.2 Setting 2: Multiple eigenvalues for the Newton potential . . . 144
7.4.3 Setting 3: Henon-Heiles potential 145
7.5 Conclusion 147

A Appendix: Smooth decompositions of matrices 149

Bibliography 151

Curriculum Vitae 159

Chapter 1

▶ Introduction

High-dimensional mathematical models frequently appear in science and engineering and are a source of very challenging computational problems. These problems are often formulated as *optimization problems*. For example, a chemist may ask: "How does this molecule look like, that is, what molecular structure *minimizes* its energy?" Answering this question requires an optimization procedure combined with the solution of the Schrödinger equation, a partial differential equation, for this molecular system. Even for rather small molecules, the involved degrees of freedom quickly become computationally infeasible.

Formally speaking, we want to solve an optimization problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^N} f(x), \tag{1.1}$$

where $f : \mathbb{R}^N \to \mathbb{R}$ is a *cost function* and the size of x is exceedingly large, say, $N = n^d = 10^{42}$. To still be able to solve these problems, the underlying structure of the problem has to be identified and exploited. Depending on the source of the problem, assuming a *low-rank structure* of x is a sensible choice. For example, let us assume x is represented as the rank-1 product

$$x = x_d \otimes x_{d-1} \otimes \cdots \otimes x_1, \qquad x_i \in \mathbb{R}^n, i = 1, \ldots, d. \tag{1.2}$$

Then, the vector x is described by only $nd = 10 \cdot 42 = 420$ degrees of freedom instead of $n^d = 10^{42}$. This product structure also allows us to view the vector x as a 42-dimensional *tensor* $\mathbf{X} \in \mathbb{R}^{10 \times 10 \times \cdots \times 10}$. Based on the simple product structure (1.2), much more sophisticated types of low-rank tensor formats have been developed, with different generalizations of the well-known matrix rank to the tensor case.

In this thesis, we focus on the *Tucker* and *tensor train* (TT) formats for dealing with low-rank structure in low and high dimensional tensors, respectively. The Tucker format dates back to the 1960s [Tuc66] and was introduced as a way to perform so-called *multiway factor analysis* for psychometric studies. Since then, it has seen various applications. Most often, the Tucker format is used in the three-dimensional case. For higher dimensions, its storage scales exponentially with the rank, which limits its practicality. The TT format remedies this exponential scaling and is thus better suited for high-dimensional problems. In the numerical linear algebra community, this ansatz was developed by Oseledets and Tyrtyshnikov [OT09, Ose11c] but was already used earlier under the name *matrix product states* (MPS) in the physics community to represent quantum states of 1D spin chains [AKLT87, AKLT88, HWSH13, FNW92, OR95, Sch11, Vid03, Whi92]. We mention a third and slightly more general format, the tree-structured *hierarchical Tucker* (HT) representation [Gra10, HK09], but will not discuss it in this thesis.

One important feature that all tensor formats have in common is their *multilinearity*, which allows us to extend powerful tools from linear algebra to the tensor setting.

Assuming such a low-rank structure of x when viewed as a tensor $\mathbf{X}$ yields the constrained optimization problem

$$\begin{aligned} &\operatorname*{argmin}_{\mathbf{X}\in\mathbb{R}^{n\times\cdots\times n}} f(\mathbf{X}), \\ &\operatorname{rank}(\mathbf{X}) = \mathbf{r}, \end{aligned} \tag{1.3}$$

where the notion of the rank of a tensor $\mathbf{X}$ is defined by the specific tensor format and the cost function f is now seen as a function $f : \mathbb{R}^{n\times n\times\cdots\times n} \to \mathbb{R}$.

Note that the constrained optimization problem (1.3) is non-convex even if the cost function f is a convex function. In these cases, the unconstrained problem (1.1) has a unique minimum, but the rank constraint leads to a highly non-linear constrained problem (1.3) with multiple local minima.

1.1. High-dimensional problems

As instances of high-dimensional problems exhibiting a low-rank tensor structure, we consider the following settings in this thesis.

Tensor completion. In tensor completion, we aim to reconstruct a d-dimensional data set $\mathbf{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ of which the vast majority of entries is missing. In this case, the cost function is given by

$$f(\mathbf{X}) = \frac{1}{2}\| \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}\|^2, \tag{1.4}$$

where the squared norm $\|\cdot\|^2$ is the sum of the squared entries of the tensor. We denote by P_Ω the projection onto the *sampling set* $\Omega \subset \{1, 2, \dots, n_1\} \times \cdots \times \{1, 2, \dots, n_d\}$ corresponding to the indices of the known entries of $\mathbf{A}$,

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

The tensor completion problem (1.4) and variants thereof have been discussed a number of times in the literature. Most of this work builds upon existing work for the special case $d = 2$, also known as matrix completion, see [MWG$^+$] for a comprehensive overview.

Algorithms developed for the Tucker format range from alternating minimization [LMWY09, LS13] and convex optimization, using various generalizations of the nuclear norm [LMWY09, LS13, GRY11, SLS10, STL$^+$14, SPM$^+$11], to Riemannian optimization techniques [KM15].

In the TT format, Grasedyck *et al.* [GKK13] presented an alternating direction method. For the HT format, Da Silva and Herrmann [DSH15] have derived HTOpt, a Riemannian approach closely related to the work presented in this thesis.

Linear systems. An example of a tensor-structured linear system is obtained from a finite difference discretization of a high-dimensional PDE such as the Poisson equation on a uniform tensor grid with $n_1 n_2 \cdots n_d$ grid points. This results in a linear system

$$Ax = f, \quad A = L_d \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes L_1,$$

with $L_\mu \in \mathbb{R}^{n_\mu \times n_\mu}$, $\mu = 1, \ldots, d$, being the one-dimensional finite difference matrices. The solution $x \in \mathbb{R}^{n_1 n_2 \cdots n_d}$ has a natural interpretation as a d-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ corresponding to the tensor grid. Thus, we can also write the linear system as $\mathcal{A}\mathbf{X} = \mathbf{F}$, where $\mathcal{A} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a symmetric linear operator. Finding the solution of such a linear system is equivalent to minimizing the cost function

$$f(\mathbf{X}) = \frac{1}{2}\langle \mathbf{X}, \mathcal{A}\mathbf{X} \rangle - \langle \mathbf{X}, \mathbf{F} \rangle, \tag{1.5}$$

where the inner product $\langle \cdot, \cdot \rangle$ of two tensor is the sum of their elementwise product. In contrast to the tensor completion problem, the Hessian of f at the solution is usually ill-conditioned, which severely slows down the convergence of gradient-based optimization algorithms. Thus, deriving efficient preconditioning schemes is a crucial ingredient when trying to create efficient solution algorithms.

Existing methods to approach high-dimensional linear systems include extensions of standard iterative solvers to the tensor case, such as the Richardson iteration or Krylov subspace methods [BG13, Dol13, KO10b, KS11, KT11a]. The multilinearity of the tensor formats lends itself well to a second class of methods based on the alternating optimization of each factor. Examples are the *alternating linear scheme* (ALS) [HRS12a], *density matrix renormalization group* (DMRG) [DO12, Ose11a] and *alternating minimal energy* (AMEn) [DS14] algorithms.

Eigenvalue problems. Eigenvalue problems are often obtained in a similar setting as linear systems and most prominently appear in the calculation of energy states of physical systems. We want to find eigenvalues $\lambda \in \mathbb{R}$ and eigenvectors $\mathbf{X}$ such that

$$\mathcal{A}\mathbf{X} = \lambda \mathbf{X}.$$

When $\mathcal{A}$ is a symmetric linear operator, the eigenvector $\mathbf{X}$ corresponding to the smallest eigenvalue is obtained from the minimizer of the Rayleigh quotient,

$$f(\mathbf{X}) = \frac{\langle \mathbf{X}, \mathcal{A}\mathbf{X} \rangle}{\langle \mathbf{X}, \mathbf{X} \rangle}.$$

Again, preconditioning is necessary to obtain fast convergence in the optimization algorithms.

If we want to compute p eigenvalues instead of only one, we can replace the Rayleigh quotient with the trace minimization formulation. In the matrix representation A of the operator $\mathcal{A}$, the corresponding optimization problem can be written as

$$\min \left\{ \operatorname{trace}(U^\mathsf{T} A U) \mid U \in \mathbb{R}^{n_1 \cdots n_d \times p},\ U^\mathsf{T} U = I_p \right\}. \tag{1.6}$$

The minimum is attained if U is an orthonormal basis for the span of the first p eigenvectors. To represent U in a tensor format, we will use the *block TT* format [DKOS13].

Similar to linear systems, existing algorithms for eigenvalue computations with low-rank tensor structure fall into two categories: extensions of classic eigenvalue solvers such as Lanczos and LOBPCG to the tensor case [HKST12, HW12, KO10b, KT11b, Leb11] or alternating optimization techniques such as ALS and DMRG [Sch11, DKOS13, KO10a, KT11b].

1.2. Riemannian optimization

Riemannian optimization is a generalization of standard Euclidean optimization methods to smooth manifolds, see [AMS08] for an overview. The fundamental idea is to formulate the optimization problem on the curved manifold instead of the standard Euclidean space. As a result, a suitably reformulated unconstrained optimization procedure will only use feasible points by construction.

Riemannian optimization provides an elegant way to approach the rank constraint in (1.3) by using the fact that the set of matrices of fixed rank,

$$\mathcal{M} = \{X \in \mathbb{R}^{n_1 \times n_2} \mid \operatorname{rank}(X) = r\},$$

forms a smooth embedded submanifold of $\mathbb{R}^{n_1 \times n_2}$, see e.g. [Lee03]. Analogous results are available for the Tucker, TT and HT tensor formats [HRS12b, Usc13, UV13].

Thus, we reformulate the constrained optimization problem (1.3) as an unconstrained optimization problem on $\mathcal{M}$,

$$\operatorname*{argmin}_{\mathbf{X} \in \mathcal{M}} f(\mathbf{X}).$$

We can then exploit the smooth manifold structure to derive efficient gradient-based algorithms. The usual notions of derivatives and line-searches now have to be replaced by their differential geometric counterparts, such as tangent vectors and curves along the manifold.

1.3. Contributions of this thesis

Chapter 2. We review the basic principles of Riemannian optimization and collect the most important results needed to perform Riemannian optimization techniques on embedded submanifolds.

Chapter 3. We discuss the Tucker format and efficient ways to perform operations on it. As we are dealing with large, high-dimensional objects, these operations have to be handled with great care such that their low-rank factorization is kept intact. In Section 3.4 we provide a new proof of the manifold structure of the set of

Tucker tensors of fixed multilinear rank and provide a framework for Riemannian optimization on the manifold of Tucker tensors. In particular, we show in Section 3.6 that the higher-order SVD fulfills the properties of a retraction map, which is needed to calculate the next iterate in a Riemannian gradient scheme.

Chapter 4. We show how we can derive similar results as in Chapter 3 but for the tensor train (TT) format, which allows us to tackle higher-dimensional problems due to its better scaling behaviour with respect to the number of dimensions. We discuss the manifold properties and all necessary ingredients for Riemannian optimization algorithms on the manifold of tensors of fixed TT rank $\mathbf{r}$. We show in Section 4.4.3 that elements of the tangent space can be conveniently represented as TT tensors but with TT rank $2\mathbf{r}$.

Chapter 5. We derive a Riemannian conjugate gradient scheme to solve the tensor completion problem (1.4). The content of this chapter is based on two papers discussing the Tucker [KSV14] and TT [Ste15] variants called geomCG and RTTC, respectively.

For the Tucker format, we illustrate the use of the developed algorithm for the recovery of multidimensional images and for the approximation of multivariate functions. Furthermore, we demonstrate competitive performance when compared to other non-Riemannian approaches to tensor completion both on synthetic datasets and applications. Exploiting the manifold structure of Tucker tensors of fixed multilinear rank, we are able to obtain an algorithm with computational complexity

$$O(|\Omega| r^d)$$

per iteration step, where $|\Omega|$ denotes the number of samples. The size of the sampling set has to exceed the dimension of the manifold, $\dim \mathcal{M} = r^d + dnr - dr^2$. Hence, for reasonable sampling sets, the computational complexity is $O(r^{2d} + dnr^{d+1})$. Thus, the algorithm scales linearly in the tensor size n, but still exponentially in the number of dimensions d. Consequently, this allows for the treatment of moderately dimensional, but very large data sets. As this scaling corresponds to the degrees of freedom of the Tucker manifold, it is necessary to use a different tensor representation to improve upon this complexity. Motivated by this, we extend this work to the tensor train (TT) format, where we obtain a scaling behaviour of

$$O(d|\Omega| r^2),$$

and hence avoid the exponential dependence on the number of dimensions. Numerical experiments and comparison to existing methods show the effectiveness of our approach. Furthermore, we demonstrate that our algorithm can obtain competitive reconstructions from uniform random sampling of few entries compared to adaptive sampling techniques such as cross-approximation.

In addition to the work presented in the two papers [KSV14, Ste15], we explore new rank adaptation schemes for the Tucker format in Section 5.3.4 and show their influence on the ability to reconstruct tensors with very few entries.

Chapter 6. We derive Riemannian optimization schemes for the solution of linear systems with tensor product structure using the cost function (1.5). The content of this chapter is based on the report [KSV15].

We propose two preconditioned gradient methods on the manifold of tensors of fixed rank: A Riemannian version of the preconditioned Richardson approach as well as an approximate Newton scheme based on the Riemannian Hessian. Special care is taken to the efficient solution of the resulting Newton equation. In numerical experiments, we compare the efficiency of our Riemannian algorithms with other established tensor-based approaches such as the preconditioned Richardson method and ALS. The results demonstrate that the approximate Newton scheme is significantly faster in cases where the application of the linear operator is expensive.

Chapter 7. We develop a low-rank tensor method with subspace correction for eigenvalue problems, called eigenvalue AMEn (EVAMEn), to solve the trace minimization problem (1.6). In our method we combine an ALS approach in the Block-TT format with a local subspace correction based on preconditioned residual information. Numerical experiments demonstrate the benefits obtained from incorporating the residual information and the importance of using local preconditioners.

The content of this chapter is based on the paper [KSU14] with an additional discussion of the convergence in Section 7.3.3.

Chapter 2

Riemannian Optimization

In this chapter, we introduce the concepts needed to formulate optimization algorithms for cost functions defined on a manifold. As a first step, we will need to review some basic concepts such as smooth maps, embedded submanifolds of $\mathbb{R}^n$ and their tangent spaces, which can be found in the first chapters of most textbooks on differential geometry. In this thesis, we only consider manifolds that are a subset of $\mathbb{R}^n$, namely matrices and tensors with a certain structure. This is possible because we can naturally identify every matrix $A \in \mathbb{R}^{n_1 \times n_2}$ with an element of $\mathbb{R}^{n_1 n_2}$. Only considering subsets $\mathcal{M} \subset \mathbb{R}^n$ simplifies the presentation considerably and avoids certain topological pitfalls.

Our presentation will mostly follow the lecture notes on differential geometry of Robbin/Salamon [RS13] for the foundations of smooth manifolds and the book by Absil, Mahony, and Sepulchre [AMS08] for optimization algorithms on manifolds. We illustrate the concepts on a simple example, the 2-sphere as a smooth embedded submanifold of $\mathbb{R}^3$, before we tackle the more complicated low-rank tensor manifolds in the next chapters.

2.1. Smooth manifolds

Before we start to define what a *smooth manifold* is, we first have to explain what we mean when we say *smooth.* Let $\mathcal{U} \subset \mathbb{R}^n$ and $\mathcal{V} \subset \mathbb{R}^m$ be open. A map $f : \mathcal{U} \to \mathcal{V}$ is C^∞ or *smooth* if it is infinitely times differentiable, that is, all its partial derivatives $\partial^{k_1+\cdots+k_n}/\partial x_1^{k_1}\cdots\partial x_n^{k_n} f$ exist and are continuous. If f is not a map between two open sets, we can resort to local open neighborhoods: A map $f : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$, is called *smooth* if for every $x \in \mathcal{X}$ there exist an open neighborhood $\mathcal{U} \subset \mathbb{R}^n$ around x and a smooth function $g : \mathcal{U} \to \mathbb{R}^m$ such that $g|_{\mathcal{U} \cap \mathcal{X}} = f$.

A *(C^∞-)diffeomorphism* is a bijection $f : \mathcal{X} \to \mathcal{Y}$ where both f itself and its inverse f^{-1} are smooth. The existence of such a function f makes the two sets $\mathcal{X}$ and $\mathcal{Y}$ *diffeomorphic* to each other.

Let us think of an m-dimensional set $\mathcal{M} \subset \mathbb{R}^n$ as a (complicated) surface in $\mathbb{R}^n$. Our goal is to locally identify patches of $\mathcal{M}$ with the Euclidean space $\mathbb{R}^m$. Then we can create local coordinate representations, *charts*, which will allow us to reduce calculus on $\mathcal{M}$ to standard calculus on $\mathbb{R}^m$. We illustrate this concept in Figure 2.1 and formally define it as follows.

Definition 2.1 (Smooth Manifold, [RS13, Def. 1.3])**.** *A subset $\mathcal{M} \subset \mathbb{R}^n$ is an m-dimensional* smooth (embedded) submanifold *of $\mathbb{R}^n$ if each $x \in \mathcal{M}$ has an open neighborhood $\mathcal{U} \in \mathbb{R}^n$ such that $\mathcal{U} \cap \mathcal{M}$ is diffeomorphic to an open subset $\mathcal{V} \in \mathbb{R}^m$. A*

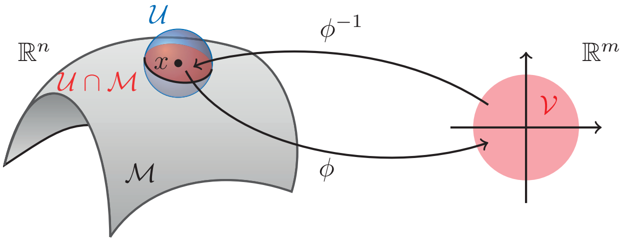


Figure 2.1: Illustration of a smooth m-dimensional submanifold of $\mathbb{R}^n$. A chart ϕ maps a neighborhood $\mathcal{U} \cap \mathcal{M}$ of a point $x \in \mathcal{M}$ to an open subset $\mathcal{V} \in \mathbb{R}^m$.

diffeomorphism $\phi : \mathcal{U} \cap \mathcal{M} \to \mathcal{V}$ is called a coordinate chart *of $\mathcal{M}$, while its inverse $\phi^{-1} : \mathcal{V} \to \mathcal{U} \cap \mathcal{M}$ is called a* parametrization *of $\mathcal{U} \cap \mathcal{M}$.*

For simple manifolds, we can directly construct coordinate charts, as shown in the following example.

Example 2.2 (2-Sphere)**.** *We want to show that the unit sphere*

$$\mathbb{S}^2 = \{x \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 = 1\}$$

is a smooth embedded submanifold of $\mathbb{R}^3$ of dimension 2. With the open sets $\mathcal{U} = \{x \in \mathbb{R}^3 \mid x_3 > 0\}$ and $\mathcal{V} = \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 < 1\}$, the diffeomorphism

$$\phi^{-1} : \mathcal{V} \to \mathcal{U} \cap \mathcal{M}, \quad (x_1, x_2) \mapsto \left(x_1, x_2, \sqrt{1 - x_1^2 - x_2^2}\right)$$

parametrizes *the north hemisphere $\mathcal{U} \cap \mathcal{M}$. Likewise, $\phi(x_1, x_2, x_3) = (x_1, x_2)$ is a* coordinate chart *of $\mathbb{S}^2$. To cover the whole sphere $\mathbb{S}^2$, we need in total six parametrizations for each of the hemispheres. The remaining ones can be obtained in an analogous way by intersecting with the open subsets determined by $x_3 < 0$, $x_2 > 0$, $x_2 < 0$, $x_1 > 0$, $x_1 < 0$. Note that this choice of coordinate charts is not unique. For example,* stereographic projection *allows us to describe $\mathbb{S}^2$ by just two charts.*

In general, directly constructing suitable charts is complicated. As we will see later on, there are other, easier ways to show that a certain set $\mathcal{M}$ is an embedded submanifold of $\mathbb{R}^n$.

2.2. Tangent space and derivatives

For embedded submanifolds of $\mathbb{R}^n$, the notion of a tangent space coincides with the intuitive concept of a plane that lies tangent to the surface described by the manifold. Formally, we can define it via the derivative of curves in $\mathcal{M}$.

Definition 2.3 ([RS13, Def. 1.21])**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be a smooth m-dimensional submanifold. A vector $\xi \in \mathbb{R}^n$ is called a* tangent vector *of $\mathcal{M}$ at a point $x \in \mathcal{M}$ if there is a smooth curve $\gamma : \mathbb{R} \to \mathcal{M}$ such that*

$$\gamma(0) = x, \qquad \gamma'(0) = \lim_{t \to 0} \frac{\gamma(t) - \gamma(0)}{t} = \xi.$$

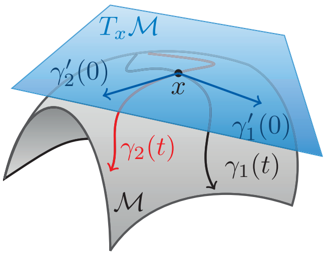


Figure 2.2: Illustration of the tangent space $T_x\mathcal{M}$ of an embedded submanifold $\mathcal{M} \subset \mathbb{R}^n$ at a point $x \in \mathcal{M}$. We see two different tangent vectors $\gamma_1'(0)$ and $\gamma_2'(0)$ realized by the two curves γ_1 and γ_2 in the manifold.

The set of tangent vectors of $\mathcal{M}$ at x,

$$T_x\mathcal{M} := \{\gamma'(0) \mid \gamma : \mathbb{R} \to \mathcal{M} \text{ smooth }, \gamma(0) = x\},$$

is called the tangent space *of $\mathcal{M}$ at x. The set of all tangent vectors to a manifold $\mathcal{M}$ is called the* tangent bundle

$$T\mathcal{M} = \bigcup_{x\in\mathcal{M}} T_x\mathcal{M}.$$

A graphical depiction of this definition is shown in Figure 2.2. It can be shown that the tangent space has the structure of a vector space of the same dimension as the manifold itself.

Proposition 2.4 ([RS13, Thm. 1.23])**.** *$T_x\mathcal{M}$ is an m-dimensional linear subspace of $\mathbb{R}^n$.*

With the tangent space at hand, we can now define the derivative of a scalar-valued smooth map living on a manifold.

Definition 2.5 ([RS13, Def. 1.31])**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be a smooth m-dimensional submanifold and $f : \mathcal{M} \to \mathbb{R}^k$ be a smooth map. The derivative of f at a point $x \in \mathcal{M}$ is the map*

$$Df(x) : T_x\mathcal{M} \to \mathbb{R}^k$$

defined as follows. Given a tangent vector $\xi \in T_x\mathcal{M}$, choose a smooth curve $\gamma : \mathbb{R} \to \mathcal{M}$ satisfying $\gamma(0) = x$ and $\gamma'(0) = \xi$. We define the vector $Df(x)[\xi] \in \mathbb{R}^k$ by

$$Df(x)[\xi] := \left.\frac{d}{dt}\right|_{t=0} f(\gamma(t)) = \lim_{h\to 0} \frac{f(\gamma(h)) - f(x)}{h} .$$

One can show that this definition does not depend on the choice of γ. More generally, if $f : \mathcal{M} \to \mathcal{N}$, where $\mathcal{N}$ is a smooth k-dimensional submanifold, then $Df(x) : T_x\mathcal{M} \to T_{f(x)}\mathcal{N}$.

The following two properties of smooth maps between manifolds are particularly useful.

Definition 2.6. *Let $\mathcal{M} \subset \mathbb{R}^n$ be a smooth m-dimensional manifold and $\mathcal{N} \subset \mathbb{R}^k$ be a smooth l-dimensional manifold. A smooth map $f : \mathcal{M} \to \mathcal{N}$ is called*

- *an* immersion *if its differential $Df(x) : T_x\mathcal{M} \to T_{f(x)}\mathcal{N}$ is injective for all $x \in \mathcal{M}$.*
- *a* submersion *if its differential $Df(x) : T_x\mathcal{M} \to T_{f(x)}\mathcal{N}$ is surjective for all $x \in \mathcal{M}$.*

Based on the inverse function theorem, we can state a much more useful way to check if a certain set is a submanifold of $\mathbb{R}^n$. We also obtain an explicit expression of its tangent space.

Theorem 2.7 ([RS13, c.f. Thm. 1.10, Thm 1.23])**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be a set. Let $\mathcal{U} \in \mathbb{R}^n$ be an open neighborhood of $x \in \mathcal{M}$ and $f : \mathcal{U} \to \mathbb{R}^{n-m}$ be a smooth map. Then, if the differential $Df(y) : \mathbb{R}^n \to \mathbb{R}^{n-m}$ is surjective for every $y \in \mathcal{U} \cap \mathcal{M}$ and*

$$\mathcal{U} \cap \mathcal{M} = f^{-1}(0) = \{z \in \mathcal{U} \mid f(z) = 0\},$$

then $\mathcal{M}$ is a smooth submanifold of $\mathbb{R}^n$ of dimension m. Furthermore, the tangent space of $\mathcal{M}$ is given by

$$T_x\mathcal{M} = \ker Df(x).$$

As an application of this theorem, we revisit Example 2.2 and obtain a much more concise proof of the manifold structure of the sphere $\mathbb{S}^2$ without the need to explicitly construct charts.

Example 2.8 (2-Sphere)**.** *The unit sphere is given by $\mathbb{S}^2 = \{x \in \mathbb{R}^3 \mid \|x\| = 1\}$. Thus, we are led to consider the map $f : \mathbb{R}^3 \to \mathbb{R}, x \mapsto x^\mathsf{T} x - 1$ for which we have $f^{-1}(0) = \mathbb{S}^2$. This map is smooth and its differential*

$$Df(x) : \mathbb{R}^3 \to \mathbb{R}, \quad \xi \mapsto 2x^\mathsf{T}\xi$$

is surjective at all points $x \in \mathbb{R}^3$ except for the origin – which is not on the sphere and thus not an element of $\mathcal{U} \cap \mathbb{S}^2$. Hence, the sphere $\mathbb{S}^2$ is a smooth submanifold of $\mathbb{R}^3$ of dimension $3 - 1 = 2$. Its tangent space consists of all vectors $\xi \in \mathbb{R}^3$ which lie in the orthogonal complement of the vector x:

$$T_x\mathbb{S}^2 = \ker Df(x) = \{\xi \in \mathbb{R}^3 \mid x^\mathsf{T}\xi = 0\}.$$

2.3. Submanifolds as subsets of manifolds

Note that up to now, we have only considered submanifolds as subsets of $\mathbb{R}^n$. The following simple definition describes submanifolds as subsets of other manifolds.

Definition 2.9 ([RS13, Def. 1.37])**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be an m-dimensional embedded submanifold of $\mathbb{R}^n$. A subset $\mathcal{N} \subset \mathcal{M}$ is called a k-dimensional embedded submanifold of $\mathcal{M}$, if $\mathcal{N}$ itself is a k-dimensional embedded submanifold of $\mathbb{R}^n$.*

We can also extend the level set description of Theorem 2.7 to this setting to characterize submanifolds.

Proposition 2.10 ([Lee03, Prop. 8.12])**.** *Let $\mathcal{N}$ be a subset of a smooth manifold $\mathcal{M}$ of dimension m. Then $\mathcal{N}$ is an embedded submanifold of $\mathcal{M}$ of dimension k if and only if every point p in $\mathcal{N}$ has a neighborhood $\mathcal{U} \subset \mathcal{M}$ such that $\mathcal{U} \cap \mathcal{N}$ is a level set of a submersion $f : \mathcal{U} \to \mathbb{R}^{m-k}$.*

2.4. Riemannian manifolds

According to Proposition 2.4, the tangent space $T_x\mathcal{M}$ has a vector space structure. Equipping it with an inner product $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ allows us to measure distances and angles. In general, we can choose a different inner product for each point $x \in \mathcal{M}$.

A *vector field* on $\mathcal{M}$ is a smooth map $\xi : \mathcal{M} \to \mathbb{R}^n$ that assigns a tangent vector $\xi(x) \in T_x\mathcal{M}$ to every point $x \in \mathcal{M}$. If the map

$$\mathcal{M} \to \mathbb{R}, \quad x \mapsto g_x(\xi(x), \eta(x))$$

is smooth for every pair of vector fields ξ, η on $\mathcal{M}$, then we call g_x a *Riemannian metric*. Note that the name is confusing, as it is an *inner product* on the tangent space. It does induce a *metric* in the usual sense, the *Riemannian distance*, see Section 2.5. A manifold equipped with a Riemannian metric is called a *Riemannian manifold*.

Note that if $\mathcal{M} \subset \mathbb{R}^n$ is a smooth embedded submanifold, then it is particularly simple to obtain a suitable Riemannian metric, as we can restrict the standard inner product of $\mathbb{R}^n$ to $T_x\mathcal{M} \times T_x\mathcal{M}$,

$$g_x : T_x\mathcal{M} \times T_x\mathcal{M}, \quad g_x(\xi, \eta) = \langle \xi, \eta \rangle.$$

The inner product defined by the Riemannian metric induces a norm on $T_x\mathcal{M}$,

$$\|\xi\| = \sqrt{g_x(\xi, \xi)}. \tag{2.1}$$

As $T_x\mathcal{M}$ is a subspace of $\mathbb{R}^n$, we can define the *normal space* as its orthogonal complement in $\mathbb{R}^n$,

$$N_x\mathcal{M} = (T_x\mathcal{M})^\perp := \{\xi \in \mathbb{R}^n \mid \langle \eta, \xi \rangle = 0 \;\; \forall \eta \in T_x\mathcal{M}\}.$$

Let us also define the orthogonal projections

$$\mathrm{P}_{T_x\mathcal{M}} : \mathbb{R}^n \to T_x\mathcal{M}, \quad \mathrm{P}_{N_x\mathcal{M}} : \mathbb{R}^n \to N_x\mathcal{M},$$

into the tangent space and normal space, respectively. Then, we can uniquely decompose every $\xi \in \mathbb{R}^n$ into

$$\xi = \mathrm{P}_{T_x\mathcal{M}}\, \xi + \mathrm{P}_{N_x\mathcal{M}}\, \xi.$$

Example 2.11 (2-Sphere)**.** *Using the standard inner product from $\mathbb{R}^3$ and restricting it to $T_x\mathbb{S}^2$, we obtain*

$$T_x\mathbb{S}^2 = \{\xi \in \mathbb{R}^3 \mid x^\mathsf{T}\xi = 0\}, \quad N_x\mathbb{S}^2 = \{\alpha x \mid \alpha \in \mathbb{R}\}$$

and the corresponding orthogonal projectors

$$\mathrm{P}_{T_x\mathcal{M}}\, \xi = (I_3 - xx^\mathsf{T})\xi, \quad \mathrm{P}_{N_x\mathcal{M}}\, \xi = xx^\mathsf{T}\xi.$$

With an inner product for the tangent space at hand, we can define the *Riemannian gradient.*

Definition 2.12. *Let $\mathcal{M}$ be a Riemannian submanifold of $\mathbb{R}^n$ and let $f : \mathcal{M} \to \mathbb{R}$ be a smooth function. Its* Riemannian gradient *at $x \in \mathcal{M}$ is defined as the unique vector* $\operatorname{grad} f(x) \in T_x\mathcal{M}$ *such that*

$$g_x\big(\operatorname{grad} f(x), \xi\big) = Df(x)[\xi] \quad \forall \xi \in T_x\mathcal{M}.$$

If f is the restriction of a smooth function $\widetilde{f} : \mathbb{R}^n \to \mathbb{R}$ defined on the whole $\mathbb{R}^n$ and $g_x(\cdot,\cdot)$ is the standard Euclidean inner product, then the Riemannian gradient takes the simple form

$$\operatorname{grad} f(x) = \mathrm{P}_{T_x\mathcal{M}} \operatorname{Grad} \widetilde{f}(x),$$

where Grad denotes the standard Euclidean gradient. The Euclidean gradient points into the direction of steepest ascent of $\widetilde{f}$. Similarly, the Riemannian gradient points into the direction *in the tangent space* which produces the steepest ascent of f,

$$\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|} = \operatorname*{argmax}_{\substack{\xi \in T_x\mathcal{M},\\ \|\xi\|=1}} Df(x)[\xi]. \tag{2.2}$$

This property allows us to define a steepest-descent-type optimization algorithm, where (2.2) is used to determine the search direction.

2.5. Distances on manifolds and geodesics

The induced norm (2.1) on $T_x\mathcal{M}$ allows us to measure the length of a smooth curve $\gamma : [0,1] \to \mathcal{M}$ within the manifold by applying the usual definition of the arc length,

$$L(\gamma) = \int_0^1 \|\gamma'(t)\| \,\mathrm{d}t.$$

This allows us to define the distance of two points $x, y \in \mathcal{M}$ as follows. Let Γ denote the set of all possible smooth curves $\gamma : [0,1] \to \mathcal{M}$ between x and y. Then, the *Riemannian distance* is the minimal arc length over all possible curves,

$$\operatorname{dist}(x,y) : \mathcal{M} \times \mathcal{M} \to \mathbb{R}, \quad \operatorname{dist}(x,y) = \inf_{\gamma \in \Gamma} L(\gamma).$$

This Riemannian distance defines a metric in the usual sense.

In Euclidean space, the shortest path between two points is the straight line, that is, a curve $\gamma(t)$ which has zero acceleration, $\gamma''(t) = 0$. We will now extend this concept of straight lines to the Riemannian setting.

Let $I \subset \mathbb{R}$ be an open interval. A *vector field along the curve* γ is a smooth map $\xi : I \to \mathbb{R}^n$ that assigns to each point $\gamma(t) \in \mathcal{M}$ a tangent vector $\xi(t) \in T_{\gamma(t)}\mathcal{M}$. The set of all such vector fields along γ is a vector space and is denoted by

$$\operatorname{Vect}(\gamma) := \{\xi : I \to \mathbb{R}^n \mid \xi \text{ smooth}, \ \xi(t) \in T_{\gamma(t)}\mathcal{M} \ \ \forall\, t \in I\}.$$

The derivative $\xi'(t)$ (in the usual sense) is not necessarily part of the tangent space $T_{\gamma(t)}\mathcal{M}$. The so-called *covariant derivative* of $\xi(t)$ at $t \in I$ is defined as its tangent component

$$\nabla : \mathrm{Vect}(\gamma) \to \mathrm{Vect}(\gamma), \quad \nabla\xi(t) := \mathrm{P}_{T_{\gamma(t)}\mathcal{M}}\, \xi'(t).$$

As $\gamma'(t)$ itself is an element of $\mathrm{Vect}(\gamma)$, we can compute its covariant derivative as the projection of the second derivative of γ into the tangent space,

$$\nabla\gamma'(t) = \mathrm{P}_{T_{\gamma(t)}\mathcal{M}}\, \gamma''(t),$$

which allows us to define "straight lines" on $\mathcal{M}$ precisely as curves whose acceleration at time t has no components within $T_{\gamma(t)}\mathcal{M}$ and only bends the curve so that it follows the surface. These curves are called *geodesics*.

Definition 2.13 ([RS13, Def. 2.18])**.** *Let $\mathcal{M} \in \mathbb{R}^n$ be a smooth Riemannian submanifold and $I \subset \mathbb{R}$ an interval. A smooth curve $\gamma : I \to \mathcal{M}$ is called a* geodesic *if*

$$\nabla\gamma'(t) = \mathrm{P}_{T_{\gamma(t)}\mathcal{M}}\, \xi''(t) = 0 \quad \forall\, t \in I.$$

2.6. The exponential map and retractions

By the Picard-Lindelöf theorem, we can show that for every $\xi \in T_x\mathcal{M}$ there exists a *maximal interval* $I_{x,\xi} \subset \mathbb{R}$ around 0 and a unique geodesic $\gamma : I_{x,\xi} \to \mathcal{M}$ satisfying $\gamma(0) = x$ and $\gamma'(0) = \xi$. We can then always choose ξ small enough such that $1 \in I_{x,\xi}$. This allows us to define the *exponential map* on the subset $\mathcal{U}_x := \{\xi \in T_x\mathcal{M} \mid 1 \in I_{x,\xi}\} \subset T_x\mathcal{M}$ by

$$\mathrm{Exp}_x : \mathcal{U}_x \to \mathcal{M}, \quad \xi \mapsto \mathrm{Exp}_x(\xi) = \gamma(1).$$

The following proposition is a direct consequence.

Proposition 2.14 ([RS13, Cor. 2.37])**.** *The map Exp_x is smooth and has the following properties:*

$$\mathrm{Exp}_x(0) = x, \quad D\,\mathrm{Exp}_x(0) = \mathrm{id}_{T_x\mathcal{M}}\,.$$

For some manifolds, we can derive an analytic expression for the geodesics and the corresponding exponential map.

Example 2.15 (2-Sphere)**.** *On the sphere $\mathbb{S}^2$, the geodesics correspond to the great circles and can be described in terms of the initial values $\gamma(0) \in \mathcal{M}$ and $\gamma'(0) \in T_{\gamma(0)}\mathcal{M}$ as*

$$\gamma(t) = \cos(\|\gamma'(0)\|t)\gamma(0) + \frac{\sin(\|\gamma'(0)\|t)}{\|\gamma'(0)\|}\gamma'(0), \tag{2.3}$$

where the norm $\|\cdot\|$ is the standard Euclidean norm of $\mathbb{R}^3$ restricted to $T_x\mathbb{S}^2$. The exponential map on $\mathbb{S}^2$ is then given by

$$\mathrm{Exp}_x(\xi) = \cos(\|\xi\|)x + \frac{\sin(\|\xi\|)}{\|\xi\|}\xi,$$

for which we can check the properties of Proposition 2.14:

$$\mathrm{Exp}_x(0) = \lim_{t\to 0} \mathrm{Exp}_x(t\xi) = \lim_{t\to 0} \cos(\|t\xi\|)x + \frac{\sin(\|t\xi\|)}{\|t\xi\|}t\xi = x,$$

and

$$D\operatorname{Exp}_x(0)\xi = \frac{d}{dt}\bigg|_{t=0} \operatorname{Exp}_x(t\xi) = \big(-\sin(\|t\xi\|)\|\xi\|x + \cos(\|t\xi\|)\xi\big)\Big|_{t=0} = \xi.$$

To see that (2.3) *is indeed a geodesic according to Definition 2.13, we calculate*

$$\begin{aligned}\nabla\gamma'(t) = \mathrm{P}_{T_{\gamma(t)}\mathcal{M}}\,\gamma''(t) &= (I - \gamma(t)\gamma(t)^{\mathsf{T}})\gamma''(t)\\ &= (I - \gamma(t)\gamma(t)^{\mathsf{T}})\gamma(t)\|\gamma'(t)\|^2 = 0.\end{aligned}$$

The exponential map can be interpreted as way to "add" a tangent vector ξ to a point x on the manifold: Starting from a point $x \in \mathcal{M}$, we move a distance of $\|\xi\|$ into the direction of ξ along the corresponding geodesic. This is a crucial ingredient when deriving optimization algorithms on manifolds, as it allows us move from the current iterate $x_k \in \mathcal{M}$ in the direction of some search direction $\xi \in T_{x_k}\mathcal{M}$ to obtain a new iterate "$x_k + \xi$" by simply setting

$$x_{k+1} = \operatorname{Exp}_{x_k}(\xi).$$

Unfortunately, for most Riemannian submanifolds $\mathcal{M}$, there is no analytic form of the Exponential map available. In particular, this is the case for the low-rank tensor manifolds examined in this thesis. To derive an expression for the geodesic γ, we have to solve the second order ordinary differential equation (ODE) of Definition 2.13 for the initial values $\gamma(0) = x$ and $\gamma'(0) = \xi$. In principle, we could solve this ODE by numerical integration, e.g., by a Runge-Kutta scheme, but this quickly becomes computationally infeasible.

The prohibitive computational cost of evaluating the exponential map for a general Riemannian submanifold $\mathcal{M}$ led to the development of cheaper alternatives which still possess those properties which are important in an optimization framework, see e.g. [EAS99, Man02, MMH94, Smi94].

The concept of a *retraction* as such an alternative goes back to Shub [Shu86]. It is based on the idea that for optimization algorithms, a first-order approximation of the exponential map is *good enough.* In particular, we ensure that the retraction is a smooth map that still fulfills the zeroth and first order conditions of Proposition 2.14, but drop the requirement that the underlying curve γ is geodesic. As a consequence, the distance travelled along γ does not necessarily equal $\|\xi\|$ anymore.

Definition 2.16 (Retraction, [AM12, Def. 1])**.** *Let $\mathcal{M}$ be a smooth submanifold of $\mathbb{R}^n$. Let 0_x denote the zero element of $T_x\mathcal{M}$. A mapping R from the tangent bundle $T\mathcal{M}$ into $\mathcal{M}$ is said to be a* retraction on $\mathcal{M}$ around $x \in \mathcal{M}$ *if there exists a neighborhood $\mathcal{U}$ of $(x, 0_x)$ in $T\mathcal{M}$ such that the following properties hold:*

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

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

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

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

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

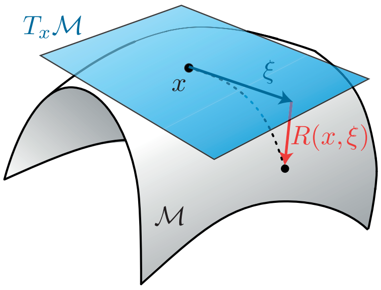


Figure 2.3: Illustration of the concept of a retraction for a smooth submanifold of $\mathbb{R}^n$*.*

Note that even though R may be defined on the whole tangent bundle $T\mathcal{M}$, its smoothness is only required locally in a neighborhood $\mathcal{U}$ of $(x, 0_x)$.

In Figure 2.3, we see an illustration of the concept of a retraction map. It follows directly from the definition of the exponential map and Proposition 2.14 that the exponential map itself is a possible retraction on $\mathcal{M}$.

If $\mathcal{M}$ is an embedded submanifold then it was shown [AM12, Prop. 5] that the metric projection

$$\mathrm{P}_{\mathcal{M}}(x+\xi) = \operatorname*{argmin}_{y\in\mathcal{M}} \|x+\xi-y\| \tag{2.4}$$

induces – if ξ is small enough – the so-called *projective retraction*

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

which satisfies the properties of Definition 2.16.

Example 2.17 (2-Sphere)**.** *The orthogonal projection onto* $\mathbb{S}^2$*,*

$$R : \mathcal{U} \to \mathbb{S}^2, \quad (x,\xi) \mapsto \frac{x+\xi}{\|x+\xi\|},$$

defines a retraction on $\mathbb{S}^2$ *according to* (2.4)*. Checking the requirements of Definition 2.16, we see that* R *is a smooth map for all* $\xi \in T_x\mathcal{M}$ *which fulfill* $\|\xi\| < \|x\| = 1$*. Furthermore, we have that*

$$R(y, 0_y) = \frac{y}{\|y\|} = y,$$

as y *lies on the sphere and*

$$\begin{aligned} DR(x,\xi)(0_x) = \frac{d}{dt}\bigg|_{t=0} R(x,t\xi) &= \frac{\xi\|x+t\xi\|^2 - (x+t\xi)(x+t\xi)^{\mathsf{T}}\xi}{\|x+t\xi\|^3}\bigg|_{t=0} \\ &= \xi - xx^{\mathsf{T}}\xi = \mathrm{P}_{T_x\mathbb{S}^2}\,\xi = \xi. \end{aligned}$$

Thus, R *indeed defines a retraction on* $\mathbb{S}^2$*.*

2.7. Parallel and vector transport

Moving along a curve γ in the manifold, one may ask how to compare two tangent vectors ξ, η which are elements of two different tangent spaces $\xi \in T_{\gamma(t_0)}\mathcal{M}$ and

$\eta \in T_{\gamma(t)}\mathcal{M}$, respectively. In this section, we will see how we can transport the tangent vector ξ along γ in a meaningful way.

Definition 2.18 ([RS13, Def. 3.1])**.** *Let $I \subset \mathbb{R}$ be an interval and $\gamma : I \to \mathcal{M}$ a smooth curve in $\mathcal{M}$. A vector field $\xi(t)$ along γ is called* parallel *if $\nabla\xi(t) = 0$ for all $t \in I$.*

One can show that for every given tangent vector $\xi_0 \in T_{\gamma(t_0)}\mathcal{M}$, there is a unique parallel vector field $\xi(t) \in \mathrm{Vect}(\gamma)$ such that $\xi(t_0) = \xi_0$. Using this vector field $\xi(t)$, we can define the *parallel transport along* γ as the collection of maps $\Phi_\gamma^{t_0\to t}$ for any $t_0, t \in I$ with

$$\Phi_\gamma^{t_0\to t} : T_{\gamma(t_0)}\mathcal{M} \to T_{\gamma(t)}\mathcal{M}, \quad \xi_0 \mapsto \Phi_\gamma^{t_0\to t}(\xi_0) := \xi(t).$$

Thus, to obtain an expression for parallel transport and its corresponding parallel vector field $\xi(t)$, we have to solve the ODE $\nabla\xi(t) := \mathrm{P}_{T_{\gamma(t)}\mathcal{M}}\, \xi'(t) = 0$ with initial condition $\xi(0) = \xi_0$. Similar to the geodesics and the exponential map discussed in the previous section, this can be computationally demanding. A possible substitute is given by the so-called *vector transport* introduced by Absil *et al.* [AMS08, Sec. 8.1].

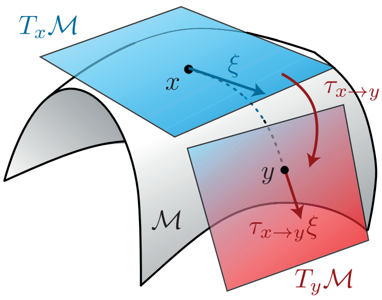


Figure 2.4: Illustration of the concept of vector transport for a smooth submanifold of $\mathbb{R}^n$.

Definition 2.19 (Vector transport, cf. [AMS08, Def. 8.1.1])**.** *A smooth mapping*

$$\begin{gathered}\tau : \{(\eta, \xi) \mid \eta, \xi \in T_x\mathcal{M}, x \in \mathcal{M}\} \to T\mathcal{M}, \\ (\eta, \xi) \mapsto \tau_\eta(\xi)\end{gathered}$$

constitutes a vector transport *if the following properties hold for all $x \in \mathcal{M}$:*

1. *There exists a retraction R* associated with *τ such that $\tau_\eta(\xi) \in T_{R(x,\eta)}\mathcal{M}$,*
2. *$\tau_{0_x}(\xi) = \xi, \quad \forall \xi \in T_x\mathcal{M}$,*
3. *$\forall \eta \in T_x\mathcal{M}$, the mapping $\tau_\eta : T_x\mathcal{M} \to T_{R(x,\eta)}, \xi \mapsto \tau_\eta(\xi)$ is linear.*

In particular, it can be shown [AMS08, Prop. 8.1.2] that the parallel transport $\Phi_\gamma^{0\to 1}$ along the curve $\gamma : t \mapsto R(x, t\eta)$ constitutes a vector transport,

$$\tau_\eta(\xi) = \Phi_\gamma^{0\to 1}(\xi)$$

For embedded submanifolds of $\mathbb{R}^n$, a simple vector transport is given by the orthogonal projection into the tangent space.

Proposition 2.20 (c.f [AMS08, Sec. 8.1.3.])**.** *Let $\mathcal{M} \in \mathbb{R}^n$ be an embedded submanifold equipped with a retraction R. Then, a vector transport is obtained from*

$$\tau_\eta(\xi) := \mathrm{P}_{T_{R(x,\eta)}\mathcal{M}}\, \xi$$

In optimization algorithms, it will be more natural to think of the vector transport as a way to transport the tangent vector $\xi \in T_x\mathcal{M}$ from the tangent space at $x \in \mathcal{M}$ to the tangent space at a new point $y \in \mathcal{M}$. This new point is given by $y = R(x, \zeta)$, where we have moved in the direction $\zeta \in T_x\mathcal{M}$ starting from x.

This allows us to introduce the shorthand notation

$$\tau_{x\to y} : T_x\mathcal{M} \to T_y\mathcal{M}, \quad \tau_{x\to y}(\xi) := \tau_\zeta(\xi).$$

An illustration of the concept of the vector transport $\tau_{x\to y}$ is shown in Figure 2.4.

2.8. Line-search algorithms on manifolds

We have now introduced all necessary tools to create gradient-based line-search algorithms on an m-dimensional Riemannian submanifold $\mathcal{M} \subset \mathbb{R}^n$ to solve problems of the form

$$\operatorname*{argmin}_{x\in\mathcal{M}} f(x),$$

where the *cost function* $f : \mathcal{M} \to \mathbb{R}$ is a scalar-valued function defined on the manifold $\mathcal{M}$.

Remark 2.21. *For the rest of this thesis, we will assume that the cost function $f : \mathcal{M} \to \mathbb{R}$ can always be seen as a restriction of a smooth function $\widetilde{f} : \mathbb{R}^n \to \mathbb{R}$ defined on the whole Euclidean space $\mathbb{R}^n$, with $f = \widetilde{f}\,|_{\mathcal{M}}$. To simplify the notation, we will from now on always write f and implicitly refer to its extension $\widetilde{f}$ where necessary.*

A line-search algorithm on $\mathcal{M}$ for f creates a sequence of iterates $(x_k)_{k\in\mathbb{N}} = \{x_0, x_1, \ldots\}$ with $x_k \in \mathcal{M}$ which (hopefully) converges to a (local) minimizer x^* of f. Such an algorithm can be constructed from the following main ingredients:

1. *A new* search direction $\eta_k \in T_{x_k}\mathcal{M}$ *at the current iterate* $x_k \in \mathcal{M}$*:*

   To construct a search direction, we can make use of Def. 2.12, the negative Riemannian gradient $\eta_k = -\operatorname{grad} f(x_k)$, which points into direction of steepest descent within the tangent space, see (2.2).

2. *A way to move from x_k in the direction of η_k:*

   This can be accomplished by a retraction map $R(x_k, \eta_k)$, see Def. 2.16.

3. *A way to determine how far we need to go in this direction:*

   An appropriate step size α_k can either be obtained from exact line-search, or, if this is not feasible, from cheaper alternatives such as Armijo backtracking or linearized line-search, see Sec. 2.8.1.

With these ingredients, the iteration is given by

$$x_{k+1} = R(x_k, \alpha_k \eta_k).$$

Two particular instances of this iteration will be discussed in Section 2.8.2 *(Riemannian steepest descent)* and Section 2.8.3 *(Riemannian conjugate gradient).*

For now, we introduce two characterizations on the search direction η_k and the step size α_k which will be helpful in the convergence discussion.

Definition 2.22 (Gradient-related sequence, [AMS08, Def. 4.2.1])**.** *A sequence* $(\eta_n)_{n\in\mathbb{N}}$ *of search directions,* $\eta_n \in T_{x_n}\mathcal{M}$*, is called* gradient-related *if, for any subsequence* $(x_{n_k})_{k\in\mathbb{N}}$ *that converges to a non-critical point of* f*, the corresponding subsequence* $(\eta_{n_k})_{k\in\mathbb{N}}$ *is bounded and satisfies*

$$\limsup_{k\to\infty} \, g_x\big(\operatorname{grad} f(x_{n_k}), \eta_{n_k}\big) < 0.$$

Definition 2.23 (Armijo condition, c.f. [AMS08, Def. 4.2.2])**.** *Let* $x \in \mathcal{M}$ *be the current iterate and* $\eta \in T_x\mathcal{M}$ *be the search direction. Furthermore, let* $\beta, c \in]0,1[$ *be fixed parameters and* $\alpha_0 > 0$ *an initial step size. The* Armijo step size *is given by* $\alpha := \beta^m \alpha_0$*, where* $m \in \mathbb{N}$ *is the smallest integer such that*

$$f(x) - f(R(x, \alpha\eta)) \geq -c\, g_x\big(\operatorname{grad} f(x), \alpha\eta\big). \tag{2.5}$$

Definition 2.23 is a straight-forward generalization of the classic Armijo rule in Euclidean space, see e.g. [NW06], to the Riemannian setting. Note that when f is continuously differentiable and the search direction is gradient-related, we can always find an $\bar{\alpha} > 0$ such that all step sizes $\alpha \in]0, \bar{\alpha}]$ fulfill the Armijo condition (2.5), see e.g. [RW12, SU15].

Algorithm 2.1 Gradient-related optimization on $\mathcal{M}$ with Armijo line-search

```
Input: Initial guess $x_0 \in \mathcal{M}$.
Output: Sequence of iterates $(x_k)$.
  for $k = 1, 2, \dots$ do
    Choose search direction $\eta_k \in T_{x_k}\mathcal{M}$ s.t. $(\eta_k)$ is gradient-related.
    Choose step size $\alpha_k$ which fulfills the Armijo condition (2.5).
    $x_{k+1} \leftarrow R(x_k, \alpha_k \eta_k)$
  end for
```

2.8.1. Approximate line-search

Determining a good step size α_k is crucial to ensure fast convergence of the optimization algorithm. The locally optimal choice would be the minimizer of the objective function along the curve $\gamma : t \mapsto R_{x_k}(t\eta_k)$ determined by the current search direction η_k:

$$\alpha_k = \operatorname*{argmin}_{\alpha} f(R(x_k, \alpha\eta_k)). \tag{2.6}$$

It is easy to check that choosing α_k in this way fulfills the Armijo condition (2.5) of Def. 2.23. Let us denote the Armijo step size with $\bar{\alpha}$. If $\alpha_k \leq \bar{\alpha}$, then α_k is also a valid step size. If $\alpha_k > \bar{\alpha}$, then we obtain

$$\begin{aligned} f(x) - f(R(x, \alpha_k \eta)) \geq f(x) - f(R(x, \bar{\alpha}\eta)) &\geq -c\langle \operatorname{grad} f(x), \bar{\alpha}\eta \rangle \\ &= -\frac{c\bar{\alpha}}{\alpha_k} \alpha_k \langle \operatorname{grad} f(x), \eta \rangle. \end{aligned}$$

And thus α_k also fulfills the Armijo condition but for a different constant $\widetilde{c} := \frac{c\bar{\alpha}}{\alpha_k} \in]0, 1[$.

Unfortunately, computing this minimizer can be a hard nonlinear optimization problem itself. A possible cheaper alternative is a classic backtracking approach, where an initial step size α_0 is chosen and then reduced as $\alpha_k = \beta^m \alpha_0$, where $\beta \in]0, 1[$ and m is chosen according to the Armijo rule, see Def. 2.23. Usual choices are $\alpha_0 = 1$, $\beta = \frac{1}{2}$, $c = 10^{-4}$, see [NW06].

As described in [Van13], the nonlinear optimization problem (2.6) can also be linearized by dropping the retraction R and thereby only minimizing *within the tangent space*,

$$\alpha_k \approx \operatorname*{argmin}_{\alpha} f(x_k + \alpha \eta_k), \tag{2.7}$$

where we now view both x_k and η_k as elements in the vector space $T_x\mathcal{M}$ to make sense of the linear combination and f is suitably extended, see Remark 2.21. In the context of this thesis, we only consider cost functions f which are quadratic. This special case allows us to solve (2.7) very efficiently, as we have an explicit solution available. The accuracy of this approximate line-search depends on the curvature of the manifold at the current iterate, but in all our experiments, we have never observed a situation where this estimate was not good enough. To make sure that the iterates fulfill the Armijo condition, a simple backtracking scheme can be added, where we choose the approximate α_k as the starting guess α.

2.8.2. Riemannian steepest descent

The simplest choice of search direction is the negative Riemannian gradient,

$$\eta_k = -\operatorname{grad} f(x_k).$$

According to (2.2), η_k then points into direction of steepest descent *within the tangent space*, yielding a direct Riemannian analogue to the classic steepest descent algorithm in Euclidean space.

The resulting optimization scheme is shown in Algorithm 2.2, where we use a linearized line search together with a backtracking scheme.

2.8.3. Riemannian conjugate gradients

In the nonlinear conjugate gradient scheme, the new search direction η_k is computed as a linear combination of the current gradient $\xi_k \in T_{x_k}\mathcal{M}$ and the previous direction

Algorithm 2.2 Riemannian steepest descent

```
Input: Initial guess $x_0 \in \mathcal{M}$, backtracking parameter $c \in ]0,1[$.
Output: Sequence of iterates $(x_k)$.
  for $k = 1, 2, \dots$ do
    $\eta_k \leftarrow -\operatorname{grad} f(x_k)$                % direction is negative Riem. gradient
    $\alpha_k \leftarrow \operatorname{argmin}_\alpha f(x_k + \alpha\eta_k)$        % step size by linearized line search
    Find smallest $m \geq 0$ such that                % Armijo backtracking for sufficient dec.
        $f(x_k) - f(R(x_k, 2^{-m}\alpha_k\eta_k)) \geq -c\, g_{x_k}(\xi_k, 2^{-m}\alpha_k\eta_k)$
    $x_{k+1} \leftarrow R(x_k, 2^{-m}\alpha_k\eta_k)$             % obtain next iterate by retraction
  end for
```

$\eta_{k-1} \in T_{x_{k-1}}\mathcal{M}$, scaled by a factor β_k. To perform this combination, we first have to transport η_{k-1} to the current tangent space $T_{x_k}\mathcal{M}$ using a vector transport $\tau_{x_{k-1}\to x_k}$, see Section 2.7. Thus, the new search direction is given by

$$\eta_k = -\xi_k + \beta_k \tau_{x_{k-1}\to x_k}\eta_{k-1}.$$

As we only consider embedded submanifolds of $\mathbb{R}^n$, the orthogonal projection onto the tangent space $\mathrm{P}_{T_{x_k}\mathcal{M}}$ yields a vector transport, see Prop. 2.20, and the equation can be written as

$$\eta_k = -\xi_k + \beta_k \,\mathrm{P}_{T_{x_k}\mathcal{M}}\, \eta_{k-1}.$$

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

$$\beta_k = \frac{\|\xi_k\|}{\|\xi_{k-1}\|}, \tag{2.8}$$

or the Polak-Ribière+ update adapted to Riemannian optimization, see [AMS08, NW06]:

$$\beta_k = \max\left\{0, \frac{g_{x_k}(\operatorname{grad} f(x_k), \operatorname{grad} f(x_k) - \tau_{x_{k-1}\to x_k}(\operatorname{grad} f(x_{k-1})))}{\|\operatorname{grad} f(x_{k-1})\|^2}\right\}. \tag{2.9}$$

The resulting Riemannian nonlinear CG scheme is shown in Algorithm 2.3.

2.8.4. Convergence of line-search algorithms

The following two theorems discuss the global convergence of Algorithm 2.1 to a critical point.

Theorem 2.24 ([AMS08, Thm. 4.3.1])**.** *Let (x_k) be an infinite sequence of iterates generated by Algorithm 2.1. Then every accumulation point $x_* \in \mathcal{M}$ of (x_k) is a critical point of the cost function f and hence satisfies* $\operatorname{grad} f(x_*) = 0$.

Unfortunately, if the manifold $\mathcal{M}$ is not closed, then the accumulation point x_* does not need to be an element of $\mathcal{M}$. Under the assumption that all iterates stay inside a compact set, we can provide a stronger result. In particular, this assumption is valid if $\mathcal{M}$ itself is a compact manifold.

Algorithm 2.3 Riemannian nonlinear CG

```
Input: Initial guess $x_0 \in \mathcal{M}$, backtracking parameter $c \in ]0,1[$.
Output: Sequence of iterates $(x_k)$.
  $\xi_0 \leftarrow \operatorname{grad} f(x_0)$                 % compute Riemannian gradient
  $\eta_0 \leftarrow -\xi_0$                                    % first step is steepest descent
  $\alpha_0 \leftarrow \operatorname{argmin}_\alpha f(x_0 + \alpha\eta_0)$   % step size by linearized line-search
  $x_1 \leftarrow R(x_0, \alpha_0\eta_0)$                       % obtain next iterate by retraction

  for $k = 1, 2, \dots$ do
    $\xi_k \leftarrow \operatorname{grad} f(x_k)$               % compute Riemannian gradient
    $\eta_k \leftarrow -\xi_k + \beta_k \tau_{x_{k-1}\to x_k}(\eta_{k-1})$   % conjugate direction by update rule
    $\alpha_k \leftarrow \operatorname{argmin}_\alpha f(x_k + \alpha\eta_k)$  % step size by linearized line search
    Find smallest $m \geq 0$ such that                          % Armijo backtracking for sufficient dec.
        $f(x_k) - f(R(x_k, 2^{-m}\alpha_k\eta_k)) \geq -c\, g_{x_k}(\xi_k, 2^{-m}\alpha_k\eta_k)$
    $x_{k+1} \leftarrow R(x_k, 2^{-m}\alpha_k\eta_k)$           % obtain next iterate by retraction
  end for
```

Corollary 2.25 ([AMS08, Cor. 4.3.2]). *Let (x_k) be an infinite sequence of iterates generated by Algorithm 2.1. Assume that the iterates stay inside the compact level set $L = \{x \in \mathcal{M} \mid f(x) \leq f(x_0)\}$. Then $\lim_{k\to\infty} \|\operatorname{grad} f(x_k)\| = 0$.*

When choosing the search direction to be the steepest descent direction, $\eta_k = -\operatorname{grad} f(x_k)$, then we can prove linear convergence to a critical point of f, where the asymptotic convergence rate depends on the eigenvalues of the *Riemannian Hessian* of f at the critical point.

Definition 2.26 ([AMS08, Prop. 5.5.6]). *Let R be a retraction and x_* be a critical point of a real-valued function f, that is,* $\operatorname{grad} f(x_*) = 0$. *Then, the* Riemannian Hessian $\mathcal{H}_{x_*}$ *of f at x_* is given by*

$$\mathcal{H}_{x_*} = \operatorname{Hess}(f \circ R(x_*, \cdot))(0_{x_*}),$$

where Hess *denotes the Euclidean Hessian.*

Theorem 2.27 ([AMS08, Thm. 4.5.6]). *Let (x_k) be an infinite sequence of iterates generated by Algorithm 2.1 with $\eta_k = -\operatorname{grad} f(x_k)$, converging to a critical point x_*. Let $\lambda_{\min}$ and $\lambda_{\max}$ be the smallest and largest eigenvalues of the Riemannian Hessian of f at x_*. Assume that $\lambda_{\min} > 0$ (x_* is a local minimizer). Then, given C in the interval $]C_*, 1[$ with*

$$C_* = 1 - \min\left(2c\,\alpha_k\,\lambda_{\min}, 4c\,(1-c)\beta\,\frac{\lambda_{\min}}{\lambda_{\max}}\right),$$

there exists an integer $K \geq 0$ such that

$$f(x_{k+1}) - f(x_*) \leq C(f(x_k) - f(x_*)) \quad \forall k \geq K.$$

For nonlinear CG, deriving similar results is difficult already in the Euclidean case, see e.g. [NW06] for a discussion. The conjugate search direction update is not guaranteed to yield a gradient-related sequence. To enforce this, a safeguard can be added that sets the update parameter β_k to zero as soon as the new search direction is not sufficiently gradient-related. By doing this, the results of Theorem 2.24 and Corollary 2.25 are also applicable to the nonlinear CG. Although we cannot prove superlinear convergence, numerical experiments show that the Riemannian nonlinear CG given by Algorithm 2.3 performs significantly better than steepest descent.

Chapter 3

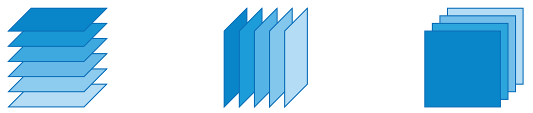
▶ Tensors of Fixed Multilinear Rank

In this chapter, we will first introduce basic properties of tensors, their representation in the Tucker format and the corresponding notion of multilinear rank. Alongside, we discuss basic operations such as addition, inner products and rank truncation. Then, we discuss the manifold properties of the set of tensors of fixed multilinear rank and derive corresponding explicit expressions for the differential geometric quantities that are needed to construct Riemannian optimization algorithms.

3.1. Tensors: Notation and basic properties

A tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is a multidimensional array of *dimension* or *order* d and *tensor size* n_μ along each *mode* $\mu = 1, \ldots, d$. In the case $d = 1$ a tensor is just a vector, for $d = 2$ a matrix. We can extend the usual column-major ordering of a matrix such that for a third-order tensor $\mathbf{X}$, the element $\mathbf{X}(i_1, i_2, i_3)$ corresponds to the element in the i_1th row and i_2th column with depth i_3.

Fixing one index while varying all others yields a *slice* of $\mathbf{X}$, a subarray of dimension $(d-1)$. In Figure 3.1, we illustrate the possible slices $\mathbf{X}(i_1, :, :)$, $\mathbf{X}(:, i_2, :)$, $\mathbf{X}(:, :, i_3)$ for $d = 3$, where we use MATLAB's colon notation to designate a range of indices.



(a) $\mathbf{X}(i_1, :, :)$ (b) $\mathbf{X}(:, i_2, :)$ (c) $\mathbf{X}(:, :, i_3)$

Figure 3.1: Possible slices of a third order tensor $\mathbf{X}$.

3.1.1. Reshaping operations

Vectorization. The isomorphism $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \simeq \mathbb{R}^{n_1 n_2 \cdots n_d}$ directly defines the *vectorization* operation, which fills the elements of $\mathbf{X}$ one after the other into a long vector. If $X \in \mathbb{R}^{n_1 \times n_2}$ is a matrix, this amounts to stacking the columns of X:

$$\operatorname{vec}(X) = \begin{bmatrix} X(:,1) \\ X(:,2) \\ \vdots \\ X(:,n_2) \end{bmatrix}.$$

Formally, we can define it by the index map $\iota : \{1, \dots, n_1\} \times \dots \times \{1, \dots, n_d\} \to \{1, 2, \dots, \prod_{\mu=1}^{d} n_\mu\}$, which is given by

$$\iota(i_1, i_2, \dots, i_d) = 1 + \sum_{\mu=1}^{d} (i_\mu - 1) \prod_{\nu=1}^{\mu-1} n_\nu. \tag{3.1}$$

Thus, if $x = \mathrm{vec}(\mathbf{X})$, then $x(\iota(i_1, \dots, i_d)) = \mathbf{X}(i_1, \dots, i_d)$, a colexicographic ordering of the entries.

Matricization. The elements of a tensor can also be arranged as a matrix. In the *μth matricization*, the tensor $\mathbf{X}$ is reshaped into a matrix $\mathbf{X}_{(\mu)} \in \mathbb{R}^{n_\mu \times n_1 \cdots n_{\mu-1} n_{\mu+1} \cdots n_d}$. Again, we can formally define a corresponding index map

$$\iota_\mu : \{1, \dots, n_1\} \times \dots \times \{1, \dots, n_d\} \to \{1, \dots, n_\mu\} \times \{1, 2, \dots, \prod_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} n_\mu\},$$

$$\iota_\mu(i_1, i_2, \dots, i_d) = (i_\mu, i_{\neq\mu}), \quad \text{with } i_{\neq\mu} = 1 + \sum_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} (i_\mu - 1) \prod_{\substack{\tau=1 \\ \tau \neq \mu}}^{\nu-1} n_\tau.$$

To illustrate this formula, we show an example for the first matricization of a third-order tensor in Figure 3.2. The colexicographic ordering of the column index $i_{\neq\mu}$ is compatible with the vectorization (3.1). We note that this is just one way of systematically reshaping a tensor into matrices. A different way of matricization, called *unfolding*, will be discussed in Chapter 4 for the tensor train format.

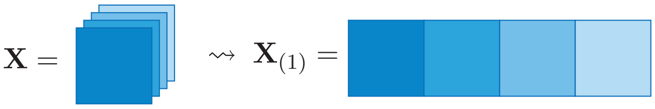


Figure 3.2: First matricization $\mathbf{X}_{(1)}$ of a third-order tensor $\mathbf{X}$.

3.1.2. Operations on tensors

Elementwise operations of matrices such as addition and multiplication by a scalar generalize naturally to the tensor case.

Inner product and norm. To define an inner product of two tensors $\mathbf{X}$ and $\mathbf{Y}$, we use the sum of the element-wise product which we can relate to the vector and matrix case by the reshaping operations defined in Subsection 3.1.1:

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \langle \mathrm{vec}(\mathbf{X}), \mathrm{vec}(\mathbf{Y}) \rangle = \mathrm{trace}\left(\mathbf{X}_{(\mu)}^\mathsf{T} \mathbf{Y}_{(\mu)}\right), \quad \mu = 1, \dots, d.$$

This induces the norm

$$\|\mathbf{X}\| = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle}.$$

In the matrix case $d = 2$, these definitions reduce to the trace inner product and the Frobenius norm, respectively.

Multiplication with a matrix. The *μth-mode product* defines the multiplication of a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ by a matrix $A_\mu \in \mathbb{R}^{m \times n_\mu}$ along the μth mode,

$$\mathbf{Y} = \mathbf{X} \times_\mu A_\mu \quad \Leftrightarrow \quad \mathbf{Y}_{(\mu)} = A_\mu \mathbf{X}_{(\mu)}. \tag{3.2}$$

The resulting tensor $\mathbf{Y}$ is then of size $n_1 \times \cdots \times n_{\mu-1} \times m \times n_{\mu+1} \times \cdots \times n_d$. A useful property connects the μth-mode product with the Kronecker product:

$$\mathbf{Y} = \mathbf{X} \times_1 A_1 \times \cdots \times_d A_d \;\Leftrightarrow\; \mathbf{Y}_{(\mu)} = A_\mu \mathbf{X}_{(\mu)} (A_d \otimes \cdots \otimes A_{\mu+1} \otimes A_{\mu-1} \otimes \cdots \otimes A_1)^{\mathsf{T}}.$$

Tensor Kronecker product. The Kronecker product of two matrices can also be generalized to the tensor setting, see [KT14, Sec. 7]. Let $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $\mathbf{Y} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ be two d-dimensional tensors. Then, the *tensor Kronecker* product $\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y}$ yields a tensor of size $n_1 m_1 \times \cdots \times n_d m_d$, defined by

$$\mathbf{Z}(k_1, \ldots, k_d) := \mathbf{X}(i_1, \ldots, i_d) \mathbf{Y}(j_1, \ldots, j_d), \qquad k_\mu = (i_\mu - 1) m_\mu + j_\mu, \tag{3.3}$$

with $i_\mu = 1, \ldots, n_\mu$ and $j_\mu = 1, \ldots, m_\mu$ for every mode $\mu = 1, \ldots, d$.

3.2. Multilinear rank and the Tucker format

Any rank-r matrix $X \in \mathbb{R}^{n_1 \times n_2}$ can be decomposed into $X = USV^{\mathsf{T}}$, where $U \in \mathbb{R}^{n_1 \times r}$, $V \in \mathbb{R}^{n_2 \times r}$ are matrices with orthonormal columns and $S \in \mathbb{R}^{r \times r}$. Such a decomposition may be obtained by the SVD, but we do not require S to be a diagonal matrix with nonnegative entries. The *Tucker format*, introduced in the 1960s by L. Tucker [Tuc66], is a direct generalization of this decomposition to higher dimensions. It has been successfully employed in various applications such as chemo- [Hen94] and psychometrics [KVM01] under the name *three-way component analysis* as a generalization of the principal component analysis (PCA) based on the standard matrix SVD. Other applications include signal [DL97] and image processing [VT02]. Representing a d-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n \times \cdots \times n}$ in a factored form, we aim to drastically reduce the storage requirement of n^d, at the expense of direct access to individual entries of the tensor. To this end, the Tucker format generalizes the notion of a low-rank format from the matrix to the tensor case, using the definition of the *multilinear rank*.

Definition 3.1 (Multilinear rank)**.** *The* multilinear rank *of a tensor* $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *is defined as the d-tuple*

$$\operatorname{rank}_{\mathrm{ML}}(\mathbf{X}) = (r_1, r_2, \ldots, r_d) = \Big(\operatorname{rank}(\mathbf{X}_{(1)}), \operatorname{rank}(\mathbf{X}_{(2)}), \ldots, \operatorname{rank}(\mathbf{X}_{(d)})\Big).$$

The Tucker format represents a tensor $\mathbf{X}$ of multilinear rank $\mathbf{r} = (r_1, r_2, \ldots, r_d)$ as

$$\mathbf{X}(i_1, \ldots, i_d) = \sum_{j_1=1}^{r_1} \cdots \sum_{j_d=1}^{r_d} \mathbf{S}(j_1, j_2, \ldots, j_d) U_1(i_1, j_1) U_2(i_2, j_2) \cdots U_d(i_d, j_d), \tag{3.4}$$

for some *core tensor* $\mathbf{S} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ and *factor matrices* $U_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$, $\mu = 1, \ldots, d$. In the following, we always choose the factor matrices to have orthonormal columns, $U_\mu^\mathsf{T} U_\mu = I_{r_\mu}$.

We denote by $\mathcal{M}_\mathbf{r}$ the set of *tensors of fixed multilinear rank*,

$$\mathcal{M}_\mathbf{r} := \{\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \mid \operatorname{rank}_{\mathrm{ML}}(\mathbf{X}) = \mathbf{r}\}.$$

Later on, we will see that this set exhibits a smooth submanifold structure which forms the basis for Riemannian optimization algorithms involving tensors of fixed multilinear rank.

Using the μth-mode product (3.2), one can write (3.4) more compactly as

$$\mathbf{X} = \mathbf{S} \times_1 U_1 \times_2 U_2 \, \cdots \times_d U_d = \mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu. \tag{3.5}$$

We note that in the matrix case $d = 2$, this formula reduces to

$$X = U_1 S U_2^\mathsf{T},$$

which is an SVD-like decomposition but with a general, not necessarily diagonal core matrix S. Each additional dimension adds another factor matrix to the representation, see Figure 3.3 for an illustration for $d = 3$.

When viewed as an element of $\mathbb{R}^{n_1 n_2 \cdots n_d}$ using the vectorization operation, a Tucker tensor $\mathbf{X}$ exhibits a Kronecker product structure,

$$\operatorname{vec}(\mathbf{X}) = (U_d \otimes U_{d-1} \otimes \cdots \otimes U_1) \operatorname{vec}(\mathbf{S}) = \left(\bigotimes_{\mu=1}^{d} U_\mu \right) \operatorname{vec}(\mathbf{S}). \tag{3.6}$$

This relation will be useful when deriving explicit expressions for some operations involving Tucker tensors.

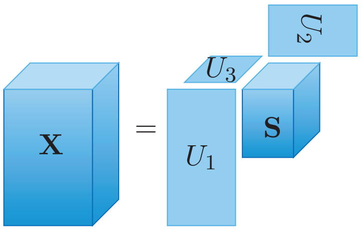


Figure 3.3: Illustration of the Tucker format for $d = 3$.

Remark 3.2 (Uniqueness)**.** *Note that the representation of a tensor $\mathbf{X} \in \mathcal{M}_\mathbf{r}$ in the Tucker format 3.5 is not unique: By introducing arbitrary orthogonal matrices $Q_\mu \in \mathbb{R}^{r_\mu \times r_\mu}$ we can also represent $\mathbf{X}$ as*

$$\mathbf{X} = \widetilde{\mathbf{S}} \times_1 U_1 Q_1 \times_2 U_2 Q_2 \cdots \times_d U_d Q_d,$$

with the modified core tensor

$$\widetilde{\mathbf{S}} = \mathbf{S} \times_1 Q_1^\mathsf{T} \times_2 Q_2^\mathsf{T} \cdots \times Q_d^\mathsf{T}.$$

Remark 3.3. *As the unfoldings $\mathbf{X}_{(\mu)}$ are not independent of each other, it is not possible to choose the multilinear rank arbitrarily. In particular, it has to hold*

$$r_\mu \leq \prod_{\substack{\nu=1 \\ \nu\neq\mu}}^{d} r_\nu. \tag{3.7}$$

This can be seen from the Tucker format (3.5)*, as $\mathbf{S}_{(\mu)}$, a matrix of size $r_\mu \times \prod_{\nu\neq\mu} r_\nu$, is required to have full rank r_μ. This is only possible if* (3.7) *holds.*

From now on, we will always assume that (3.7) is fulfilled for any rank tuple $\mathbf{r}$.

3.3. Operations on Tucker tensors

In this section, we compute some operations such as the addition or inner product of two Tucker tensors based solely on the factorized form (3.5). This is a crucial ingredient for algorithms working with Tucker tensors, as forming the full tensor in $\mathbb{R}^{n_1\times\cdots\times n_d}$ is not computationally feasible for all but the smallest examples.

When stating the computational complexity of the operations, we assume for simplicity that both the tensor size and rank are approximately equally distributed along each mode and we set

$$n := \max_{\mu\in\{1,\dots,d\}} n_\mu, \quad r := \max_{\mu\in\{1,\dots,d\}} r_\mu.$$

3.3.1. Reorthogonalization

In this thesis, we always assume that the factor matrices U_μ of a Tucker tensor $\mathbf{X} = \mathbf{S} \times_1 U_1 \cdots \times_d U_d$ have orthonormal columns. If this is not the case, we can reorthogonalize the U_μ without changing the tensor $\mathbf{X}$ itself. This is possible due to the non-uniqueness of the Tucker format, see Remark 3.2. After performing a QR-decomposition of each factor matrix,

$$[\widetilde{U}_\mu, R_\mu] = \mathrm{qr}(U_\mu),$$

we transfer the non-orthogonal parts R_μ to the core tensor $\mathbf{S}$:

$$\widetilde{\mathbf{S}} = \mathbf{S} \times_1 R_1 \times_2 R_2 \cdots \times_d R_d.$$

Then, $\widetilde{\mathbf{S}} \times_1 \widetilde{U}_1 \cdots \times_d \widetilde{U}_d$ is a Tucker decomposition of $\mathbf{X}$, where each $\widetilde{U}_\mu$ has orthonormal columns.

3.3.2. Addition

Consider the two tensors $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}, \mathbf{Y} \in \mathcal{M}_{\widetilde{\mathbf{r}}}$ represented in the Tucker format as

$$\mathbf{X} = \mathbf{S} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d, \quad \mathbf{Y} = \widetilde{\mathbf{S}} \times_1 \widetilde{U}_1 \times_2 \widetilde{U}_2 \cdots \times_d \widetilde{U}_d. \tag{3.8}$$

Then the addition $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ can be directly formulated in the Tucker format, where $\mathbf{Z} = \mathbf{R} \times_1 V_1 \times_2 V_2 \cdots \times_d V_d$ is a Tucker tensor of multilinear rank at most $\mathbf{r} + \widetilde{\mathbf{r}}$ with

$$V_\mu = \begin{bmatrix} U_\mu & \widetilde{U}_\mu \end{bmatrix} \in \mathbb{R}^{n_\mu \times (r_\mu + \widetilde{r}_\mu)}, \quad \mu = 1, \dots, d$$

and a block-superdiagonal core tensor $\mathbf{R} \in \mathbb{R}^{(r_1+\widetilde{r}_1)\times\cdots\times(r_d+\widetilde{r}_d)}$, shown in Figure 3.4 for the case $d = 3$. This procedure should be followed by a reorthogonalization step for the newly obtained V_μ.

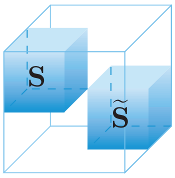
$\mathbf{R} =$ 

Figure 3.4: Block-superdiagonal structure of the core tensor $\mathbf{R}$ *resulting from an addition of two Tucker tensors with cores* $\mathbf{S}$ *and* $\widetilde{\mathbf{S}}$*, illustrated for the case* $d = 3$.

3.3.3. *Inner product and norm*

The inner product of two Tucker tensors $\mathbf{X}$ and $\mathbf{Y}$ of the form (3.8) can be reduced to the inner product of two smaller tensors, using the Kronecker product structure of the vectorization, see (3.6):

$$\begin{aligned} \langle \mathbf{X}, \mathbf{Y} \rangle &= \operatorname{vec}(\mathbf{X})^\mathsf{T} \operatorname{vec}(\mathbf{Y}) = \operatorname{vec}(\mathbf{S})^\mathsf{T} \left(\bigotimes_{\mu=1}^{d} U_\mu^\mathsf{T} \right) \left(\bigotimes_{\mu=1}^{d} \widetilde{U}_\mu \right) \operatorname{vec}(\widetilde{\mathbf{S}}) \\ &= \operatorname{vec}(\mathbf{S})^\mathsf{T} \left(\bigotimes_{\mu=1}^{d} U_\mu^\mathsf{T} \widetilde{U}_\mu \right) \operatorname{vec}(\widetilde{\mathbf{S}}) = \operatorname{vec}\left(\mathbf{S} \bigtimes_{\mu=1}^{d} \widetilde{U}_\mu^\mathsf{T} U_\mu \right)^\mathsf{T} \operatorname{vec}(\widetilde{\mathbf{S}}) \\ &= \left\langle \mathbf{S} \bigtimes_{\mu=1}^{d} \widetilde{U}_\mu^\mathsf{T} U_\mu, \, \widetilde{\mathbf{S}} \right\rangle. \end{aligned} \tag{3.9}$$

Thus, we first calculate the d products $\widetilde{U}_\mu^\mathsf{T} U_\mu$ for $\mu = 1, \dots, d$ using $O(dr\widetilde{r}n)$ flops. Then, we explicitly form the small tensor $\mathbf{S} \times_1 \widetilde{U}_1^\mathsf{T} U_1 \cdots \times_d \widetilde{U}_d^\mathsf{T} U_d$ of size $\widetilde{r}_1 \times \cdots \times \widetilde{r}_d$. If we assume $r \geq \widetilde{r}$, then this requires $O(d\widetilde{r}r^d)$ operations. It remains to calculate the inner product of two small tensors of size $\widetilde{r}_1 \times \cdots \times \widetilde{r}_d$ using $O(\widetilde{r}^d)$ multiplications and additions. If we furthermore assume that $r \approx \widetilde{r}$, we end up with a total computational complexity of

$$O(dr^2 n + dr^{d+1} + r^d).$$

Computation of the norm is particularly simple, as we assume that the factor matrices have orthonormal columns and thus the product $U_\mu^\mathsf{T} U_\mu$ is the identity,

$$\|\mathbf{X}\| = \langle \mathbf{X}, \mathbf{X} \rangle = \operatorname{vec}\left(\mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T} U_\mu \right)^\mathsf{T} \operatorname{vec}(\mathbf{S}) = \operatorname{vec}(\mathbf{S})^T \operatorname{vec}(\mathbf{S}) = \|\mathbf{S}\|.$$

Thus, the calculation of the norm can be performed in $O(r^d)$ operations.

3.3.4. Hadamard product

The *Hadamard* or *elementwise product* $\mathbf{Z} := \mathbf{X} \star \mathbf{Y}$ of two Tucker tensors $\mathbf{X}$ and $\mathbf{Y}$ of the form (3.8) is defined as

$$\mathbf{Z}(i_1, \ldots, i_d) = \mathbf{X}(i_1, \ldots, i_d)\mathbf{Y}(i_1, \ldots, i_d), \quad i_\mu = 1, \ldots, n_\mu.$$

Although direct access to the elements of $\mathbf{X}$ and $\mathbf{Y}$ is not available in the Tucker format, this elementwise operation can still be performed efficiently without constructing the full tensor. The result $\mathbf{Z} = \mathbf{R} \times_1 V_1 \times_2 V_2 \cdots \times_d V_d$ is a Tucker tensor of multilinear rank $(r_1\widetilde{r}_1, \ldots, r_d\widetilde{r}_d)$ with core tensor

$$\mathbf{R} = \mathbf{S} \otimes \widetilde{\mathbf{S}},$$

obtained by a tensor Kronecker product (3.3), and factor matrices

$$V_\mu = U_\mu \odot^\mathsf{T} \widetilde{U}_\mu, \quad \mu = 1, \ldots, d.$$

Here, $\odot^\mathsf{T}$ denotes the *transposed Khatri-Rao* product, see [KT14], a row-wise Kronecker product defined for two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times k}$ as

$$A \odot^\mathsf{T} B := \begin{bmatrix} A(1,:) \otimes B(1,:) \\ \vdots \\ A(n,:) \otimes B(n,:) \end{bmatrix} \in \mathbb{R}^{n \times mk}.$$

3.3.5. Rank truncation and higher-order SVD

To obtain a multilinear rank-$\mathbf{r}$ approximation $\widetilde{\mathbf{X}}$ of a tensor $\mathbf{X}$, we can make use of a generalization of the singular value decomposition (SVD), the higher-order SVD (HOSVD). The truncated HOSVD procedure, shown in Algorithm 3.1, can be described by the successive application of best rank-r_μ approximations $\mathrm{P}^\mu_{r_\mu}$ in each mode $\mu = 1, \ldots, d$:

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

The application of a single projector $\mathrm{P}^\mu_{r_\mu}$ is given explicitly as $(\mathrm{P}^\mu_{r_\mu} \mathbf{X})_{(\mu)} = QQ^\mathsf{T}\mathbf{X}_{(\mu)}$, where $Q \in \mathbb{R}^{n_\mu \times r_\mu}$ contains the first r_μ left singular vectors of $\mathbf{X}_{(\mu)}$.

While the truncated HOSVD does not yield the best rank-$\mathbf{r}$ approximation like the truncated SVD, it fulfills a quasi-best approximation property [LMV00, Hac12].

Theorem 3.4.

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

where $\mathrm{P}_\mathbf{r} \mathbf{X}$ *is the projection yielding any best rank-*$\mathbf{r}$ *approximation of* $\mathbf{X}$ *in the norm* $\| \cdot \|$.

Note that such a best rank-$\mathbf{r}$ approximation $\mathrm{P}_\mathbf{r} \mathbf{X}$ is guaranteed to exist, see [Usc13, Cor. 7.2].

The HOSVD inherits the smoothness of low-rank matrix approximations.

Algorithm 3.1 Truncated HOSVD: Successive higher-order SVD

Input: Tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $\mathbf{r}$.
Output: Multilinear rank-$\mathbf{r}$ approximation $\widetilde{\mathbf{X}}$ of $\mathbf{X}$ in the Tucker format,

$$\widetilde{\mathbf{X}} = \mathbf{S} \times U_1 \times U_2 \times \cdots \times U_d.$$

```
$\mathbf{S} \leftarrow \mathbf{X}$
for $\mu = 1, \dots, d$ do
    $[U, \Sigma, V] \leftarrow \mathrm{svd}(\mathbf{S}_{(\mu)})$
    $U_\mu \leftarrow U(:, 1 : r_\mu)$
    $\mathbf{S}_{(\mu)} \leftarrow \Sigma(1 : r_\mu, 1 : r_\mu) V(:, 1 : r_\mu)^\mathsf{T}$
end for
```

Proposition 3.5 (Smoothness of truncated HOSVD)**.** *Let* $\mathbf{X} \in \mathcal{M}_\mathbf{r}$. *Then there exists a neighborhood* $\mathcal{U} \subset \mathbb{R}^{n_1 \times \cdots \times n_d}$ *of* $\mathbf{X}$ *such that* $\mathrm{P}^{\mathrm{HO}}_\mathbf{r} : \mathcal{U} \to \mathcal{M}_\mathbf{r}$ *is smooth.*

Proof. Let $\mathcal{U}_\mu$ denote the open set of tensors whose μth matricization has a nonzero gap between the r_μth and the $(r_\mu + 1)$th singular values. It was shown in [BGBMN91, CD00] that in some neighborhood, the factors of an SVD decomposition depend smoothly on the original matrix if we additionally allow for negative singular values and a change in the order of the singular values. Therefore, we can assume that the matrix Q containing the first r_μ left singular vectors of a rank-r_μ matrix $\mathbf{X}_{(\mu)}$ depends smoothly on $\mathbf{X}_{(\mu)}$. Hence the projector $(\mathrm{P}^\mu_{r_\mu} \mathbf{X})_{(\mu)} = QQ^\mathsf{T} \mathbf{X}_{(\mu)}$ is also smooth and well-defined on $\mathcal{U}_\mu$. Clearly, $\mathbf{X} \in \mathcal{M}_\mathbf{r}$ is contained in all $\mathcal{U}_\mu$ and is a fixpoint of every $\mathrm{P}^\mu_{r_\mu}$:

$$\mathrm{P}^\mu_{r_\mu} \mathbf{X} = \mathbf{X} \quad \forall \mu = 1, \dots, d \quad \Rightarrow \quad \mathrm{P}^\mu_{r_\mu} \circ \cdots \circ \mathrm{P}^1_{r_1} \mathbf{X} = \mathbf{X}.$$

Thus, it is possible to construct an open neighborhood $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of $\mathbf{X}$ such that $\mathrm{P}^\mu_{r_\mu} \circ \cdots \circ \mathrm{P}^1_{r_1} \mathcal{U} \subseteq \mathcal{U}_\mu$ for all μ. Hence, the chain rule yields the smoothness of the operator $\mathrm{P}^{\mathrm{HO}}_\mathbf{r}$ on $\mathcal{U}$. □

The HOSVD is a main ingredient of most of the algorithms presented in this thesis. If $\mathbf{X}$ is a general, unstructured d-dimensional tensor, the computation of the HOSVD can become very expensive as the tensor sizes increase, since we have to form the singular value decomposition of each matricization. In practical algorithms which exploit the low-rank Tucker structure, we can often assume that the tensor is already in the Tucker format, but its rank $\mathbf{r}$ is higher than the desired target rank $\widetilde{\mathbf{r}}$. In this setting, we can use Algorithm 3.2, which implements a low-rank recompression. Note that it is crucial for this algorithm that the factor matrices U_μ of the input tensor $\mathbf{X}$ have orthonormal columns. If this is not the case, they can be orthonormalized beforehand by a QR-procedure, see Section 3.3.1.

Algorithm 3.2 Low-rank recompression using the HOSVD

Input: Rank-$\mathbf{r}$ Tucker tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ represented by $\mathbf{X} = \mathbf{S} \times U_1 \times U_2 \times \cdots \times U_d$. Desired target rank $\widetilde{\mathbf{r}}$ with $\widetilde{r}_\mu \leq r_\mu$, $\mu = 1, \ldots, d$.

Output: Multilinear rank-$\widetilde{\mathbf{r}}$ approximation $\widetilde{\mathbf{X}}$ of $\mathbf{X}$ in the Tucker format,

$$\widetilde{\mathbf{X}} = \widetilde{\mathbf{S}} \times \widetilde{U}_1 \times \widetilde{U}_2 \times \cdots \times \widetilde{U}_d.$$

```
$[\widetilde{\mathbf{S}}, V_1, \ldots, V_d] \leftarrow \mathrm{hosvd}(\mathbf{S}, \widetilde{\mathbf{r}})$
for $\mu = 1, \ldots, d$ do
    $\widetilde{U}_\mu \leftarrow U_\mu V_\mu$
end for
```

3.4. Manifold structure

Let us assume that we have a high-dimensional optimization problem where the solution $\mathbf{X}$ has low multilinear rank and can therefore be represented efficiently in the Tucker format. To apply the Riemannian optimization techniques introduced in Chapter 2 to this problem, we first have to investigate in a suitable manifold structure. The following theorem yields that the set of Tucker tensors of fixed multilinear rank is indeed a smooth embedded submanifold of Euclidean space. This result was already used in [KL10, Hac12], with a formal proof presented in [Usc13] for the more general case of Hilbert spaces. There, the proof is based on identifying the non-uniqueness of the Tucker format and defining a corresponding Lie group action whose orbits yield equivalence classes. Here, we present a constructive approach which explicitly uses the characterization of an embedded submanifold as the level set of a submersion and thus uses only basic differential geometric concepts.

Theorem 3.6. *The set of Tucker tensors of fixed multilinear rank,*

$$\mathcal{M}_{\mathbf{r}} := \{\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \mid \mathrm{rank}_{\mathrm{ML}}(\mathbf{X}) = \mathbf{r}\},$$

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

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

Proof. We first note again that the multilinear rank $\mathbf{r}$ cannot be chosen arbitrarily and has to fulfill the condition of Remark 3.3.

The proof proceeds inductively by fitting the rank in each matricization to reach the target rank $\mathbf{r}$. First, we will show that the set

$$\mathcal{M}^1_{r_1} = \{\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \mid \mathrm{rank}(\mathbf{X}_{(1)}) = r_1\}.$$

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

$$\left(n_1 + \prod_{\mu=2}^{d} n_\mu\right) r_1 - r_1^2.$$

Then, we show that the recursively defined set, $\mu > 1$,

$$\mathcal{M}^{1,\ldots,\mu}_{r_1,\ldots,r_\mu} = \{\mathbf{X} \in \mathcal{M}^{1,\ldots,\mu-1}_{r_1,\ldots,r_{\mu-1}} \mid \operatorname{rank}(\mathbf{X}_{(\mu)}) = r_\mu\},$$

forms a smooth embedded submanifold of $\mathcal{M}^{1,\ldots,\mu-1}_{r_1,\ldots,r_{\mu-1}}$ of dimension

$$\left(\prod_{\nu=1}^{\mu} r_\nu\right)\left(\prod_{\nu=\mu+1}^{d} n_\nu\right) + \sum_{\nu=1}^{\mu}(r_\nu n_\nu - r_\nu^2).$$

From this we obtain for $\mu = d$ the set $\mathcal{M}^{1,\ldots,d}_{r_1,\ldots,r_d}$, which is identical to $\mathcal{M}_{\mathbf{r}}$. Hence, $\mathcal{M}_{\mathbf{r}}$ is also a smooth embedded submanifold of $\mathbb{R}^{n_1\times\cdots\times n_d}$ of the same dimension as $\mathcal{M}^{1,\ldots,d}_{r_1,\ldots,r_d}$, which will prove the theorem.

Base case, $\mu = 1$. We show that the set

$$\mathcal{M}^{1}_{r_1} = \{\mathbf{X} \in \mathbb{R}^{n_1\times\cdots\times n_d} \mid \operatorname{rank}(\mathbf{X}_{(1)}) = r_1\}$$

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

For this base case, we follow the proof that the set of fixed-rank matrices forms a smooth embedded submanifold [Lee03, Example 8.14] and apply it to the first matricization. Using row- and column permutations $P_{r,1}$ and $P_{c,1}$, we can partition $\mathbf{X}_{(1)}$ into blocks

$$\mathbf{X}_{(1)} = P_{r,1}\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} P_{c,1}, \tag{3.10}$$

where $A_1 \in \mathbb{R}^{r_1\times r_1}$ is non-singular and $D_1 \in \mathbb{R}^{(n_1-r_1)\times(n_2\cdots n_d-r_1)}$. We define a subset $\mathcal{S}_1$ containing $\mathbf{X}$ by

$$\mathcal{S}_1 = \left\{\mathbf{X} \in \mathbb{R}^{n_1\times\cdots\times n_d} \;\middle|\; \mathbf{X}_{(1)} = P_{r,1}\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} P_{c,1},\ \det(A_1) \neq 0\right\}.$$

As the row- and column permutations are linear isomorphisms and the determinant function is continuous, $\mathcal{S}_1$ is an open subset of $\mathbb{R}^{n_1\times\cdots\times n_d}$.

Using this block structure, we will now show that a tensor $\mathbf{X} \in \mathcal{S}_1$ fulfills $\operatorname{rank}(\mathbf{X}_{(1)}) = r_1$ if and only if the Schur complement is a matrix of all zeros,

$$D_1 - C_1 A_1^{-1} B_1 = 0.$$

For this, we consider the invertible matrix $R \in \mathbb{R}^{n_2\cdots n_d \times n_2\cdots n_d}$ given by

$$R = P_{c,1}^{-1}\begin{bmatrix} A_1^{-1} & -A_1^{-1}B_1 \\ 0 & I_{n_2\cdots n_d - r_1} \end{bmatrix}.$$

As R and the permutations are invertible, we have $\operatorname{rank}(\mathbf{X}_{(1)}) = \operatorname{rank}(P_{r,1}^{-1}\mathbf{X}_{(1)}R)$ and hence

$$\begin{aligned} P_{r,1}^{-1}\mathbf{X}_{(1)}R &= P_{r,1}^{-1}P_{r,1}\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} P_{c,1}P_{c,1}^{-1}\begin{bmatrix} A_1^{-1} & -A_1^{-1}B_1 \\ 0 & I_{n_2\cdots n_d - r_1} \end{bmatrix} \\ &= \begin{bmatrix} I_{r_1} & 0 \\ C_1A_1^{-1} & D_1 - C_1A_1^{-1}B_1 \end{bmatrix}, \end{aligned}$$

from which we can see that $P_{r,1}^{-1}\mathbf{X}_{(1)}R$ has rank r_1 if and only if its lower right part $D_1 - C_1 A_1^{-1} B_1$ is zero.

Therefore, we can describe all tensors in $\mathcal{S}_1 \cap \mathcal{M}_{r_1}^1$ by the level set of the function

$$\Phi_1 : \mathcal{S}_1 \to \mathbb{R}^{(n_1-r_1)\times(n_2\cdots n_d - r_1)}, \quad \Phi_1(\mathbf{X}) = D_1 - C_1 A_1^{-1} B_1.$$

This function is smooth, as it only involves the inversion of a non-singular matrix and basic matrix operations. To see that Φ_1 is a submersion, see Def. 2.6, we show that $D\Phi_1(\mathbf{X}) : T_{\mathbf{X}}\mathcal{S}_1 \simeq \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{(n_1-r_1)\times(n_2\cdots n_d-r_1)}$ is surjective at any $\mathbf{X} \in \mathcal{S}_1$. Consider the path

$$\gamma : \mathbb{R} \to \mathcal{S}_1, \quad \gamma(t)_{(1)} = P_{r,1}\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 + tE_1 \end{bmatrix} P_{c,1}$$

with an arbitrary matrix $E_1 \in \mathbb{R}^{(n_1-r_1)\times(n_2\cdots n_d-r_1)}$. Then, we have

$$D\Phi_1(\mathbf{X})[\gamma'(0)] = (\Phi_1 \circ \gamma)'(t)\Big|_{t=0} = \frac{d}{dt}\Big|_{t=0}(D_1 + tE_1 - C_1 A_1^{-1} B_1) = E_1$$

Thus, every point $\mathbf{X} \in \mathcal{M}_{r_1}^1$ has a neighborhood $\mathcal{S}_1 \subset \mathbb{R}^{n_1\times\cdots\times n_d}$ such that $\mathcal{S}_1 \cap \mathcal{M}_{r_1}^1$ is the level set of a submersion Φ_1 and, thus, by Proposition 2.10, $\mathcal{M}_{r_1}^1$ is an embedded submanifold of $\mathbb{R}^{n_1\times\cdots\times n_d}$ with dimension

$$\begin{aligned}\dim \mathcal{M}_{r_1}^1 &= \dim\ker\Phi_1 = \dim(\mathbb{R}^{n_1\times\cdots\times n_d}) - \dim(\mathbb{R}^{(n_1-r_1)\times(n_2\cdots n_d-r_1)}) \\ &= \left(\prod_{\mu=1}^{d} n_\mu\right) - (n_1 - r_1)\left(\prod_{\mu=2}^{d} n_\mu - r_1\right) = \left(n_1 + \prod_{\mu=2}^{d} n_\mu\right) r_1 - r_1^2.\end{aligned}$$

Inductive step, $\mu - 1 \to \mu$. Assume that $\mathcal{M}_{r_1,\ldots,r_{\mu-1}}^{1,\ldots,\mu-1}$ is a smooth embedded submanifold of $\mathcal{M}_{r_1,\ldots,r_{\mu-2}}^{1,\ldots,\mu-2}$. Then we want to show that $\mathcal{M}_{r_1,\ldots,r_\mu}^{1,\ldots,\mu}$ is a smooth embedded submanifold of $\mathcal{M}_{r_1,\ldots,r_{\mu-1}}^{1,\ldots,\mu-1}$.

We will proceed in a similar way as for the base case. We can represent every $\mathbf{X} \in \mathcal{M}_{r_1,\ldots,r_{\mu-1}}^{1,\ldots,\mu-1}$ as

$$\begin{aligned} \mathbf{X}_{(\mu-1)} &= U_{\mu-1}\mathbf{S}_{(\mu-1)}(I_{n_\mu\cdots n_d} \otimes U_{\mu-2}^{\mathsf{T}} \otimes \cdots \otimes U_1^{\mathsf{T}}) \\ \Leftrightarrow \quad \mathbf{X} &= \mathbf{S} \times_1 U_1 \times_2 U_2 \cdots \times_{\mu-1} U_{\mu-1}. \end{aligned}$$

where the factor matrices $U_\nu \in \mathbb{R}^{n_1\times r_1}$ are matrices with full column rank and $\mathbf{S} \in \mathbb{R}^{r_1\times\cdots\times r_{\mu-1}\times n_\mu\times\cdots\times n_d}$. Such a representation of the $(\mu-1)$th matricization can be obtained by, e.g., a QR-like-decomposition of $\mathbf{X}_{(\mu-1)}$. As shown in Corollary A.2, the factors of such a decomposition depend smoothly on the original matrix $\mathbf{X}_{(\mu-1)}$. In the μth matricization, $\mathbf{X}$ is then given by

$$\mathbf{X}_{(\mu)} = \mathbf{S}_{(\mu)}(I_{n_{\mu+1}\cdots n_d} \otimes U_{\mu-1}^{\mathsf{T}} \otimes \cdots \otimes U_1^{\mathsf{T}}).$$

As $I_{n_{\mu+1}\cdots n_d} \otimes U_{\mu-1}^{\mathsf{T}} \otimes \cdots \otimes U_1^{\mathsf{T}}$ has full row rank $(\prod_{\nu=1}^{\mu-1} r_\nu)(\prod_{\nu=\mu+1}^{d} n_\nu) \geq r_\mu$, it suffices to look at the rank of $\mathbf{S}_{(\mu)}$. Analogous to the base case, we can introduce row- and column permutations $P_{r,\mu}$ and $P_{c,\mu}$ to reveal the block structure of $\mathbf{S}_{(\mu)}$,

$$\mathbf{X}_{(\mu)} = P_{r,\mu}\begin{bmatrix} A_\mu & B_\mu \\ C_\mu & D_\mu \end{bmatrix} P_{c,\mu}(I_{n_{\mu+1}\cdots n_d} \otimes U_{\mu-1}^{\mathsf{T}} \otimes \cdots \otimes U_1^{\mathsf{T}}). \tag{3.11}$$

where $A_\mu \in \mathbb{R}^{r_\mu \times r_\mu}$ is a non-singular matrix and $D_\mu \in \mathbb{R}^{(n_\mu - r_\mu)\times(r_1\cdots r_{\mu-1}n_{\mu+1}\cdots n_d - r_\mu)}$. Thus, let us consider the relatively open set

$$\begin{aligned}\mathcal{S}_\mu = \Big\{\mathbf{X} \in \mathcal{M}^{1,\dots,\mu-1}_{r_1,\dots,r_{\mu-1}} \;\Big|\; &\\ \mathbf{X}_{(\mu)} = P_{r,\mu}\begin{bmatrix} A_\mu & B_\mu \\ C_\mu & D_\mu \end{bmatrix} P_{c,\mu}(I_{n_{\mu+1}\cdots n_d} \otimes U_{\mu-1}^\mathsf{T} \otimes \cdots \otimes U_1^\mathsf{T}), \det(A_\mu) \neq 0\Big\}.&\end{aligned}$$

Using the block structure of $\mathbf{S}_{(\mu)}$, we will now show that a tensor $\mathbf{X} \in \mathcal{S}_\mu$ fulfills $\operatorname{rank}(\mathbf{X}_{(\mu)}) = \operatorname{rank}(\mathbf{S}_{(\mu)}) = r_\mu$ if and only if the Schur complement is a matrix of all zeros,

$$D_\mu - C_\mu A_\mu^{-1} B_\mu = 0.$$

To this end, we consider the invertible matrix $R \in \mathbb{R}^{m\times m}$, $m = r_1 \cdots r_{\mu-1} n_{\mu+1} \cdots n_d$, given by

$$R = P_{c,\mu}^{-1}\begin{bmatrix} A_\mu^{-1} & -A_\mu^{-1}B_\mu \\ 0 & I_{m-r_\mu} \end{bmatrix}.$$

As R and the permutations are invertible, we have $\operatorname{rank}(\mathbf{S}_{(\mu)}) = \operatorname{rank}(P_{r,\mu}^{-1}\mathbf{S}_{(\mu)}R)$ and hence

$$\begin{aligned} P_{r,\mu}^{-1}\mathbf{S}_{(\mu)}R &= P_{r,\mu}^{-1}P_{r,\mu}\begin{bmatrix} A_\mu & B_\mu \\ C_\mu & D_\mu \end{bmatrix} P_{c,\mu}P_{c,\mu}^{-1}\begin{bmatrix} A_\mu^{-1} & -A_\mu^{-1}B_\mu \\ 0 & I_{m-r_\mu} \end{bmatrix} \\ &= \begin{bmatrix} I_{r_\mu} & 0 \\ C_\mu A_\mu^{-1} & D_\mu - C_\mu A_\mu^{-1}B_\mu \end{bmatrix}, \end{aligned}$$

from which we can see that $P_{r,\mu}^{-1}\mathbf{S}_{(\mu)}R$ has rank r_μ if and only if its lower right part $D_\mu - C_\mu A_\mu^{-1}B_\mu$ is zero.

Thus, we can describe all tensors in $\mathcal{S}_\mu \cap \mathcal{M}^{1,\dots,\mu}_{r_1,\dots,r_\mu}$ by the level set of

$$\Phi_\mu : \mathcal{S}_\mu \to \mathbb{R}^{(n_\mu - r_\mu)\times(r_1\cdots r_{\mu-1}n_{\mu+1}\cdots n_d - r_\mu)}, \quad \Phi_\mu(\mathbf{X}) = D_\mu - C_\mu A_\mu^{-1}B_\mu.$$

Again, we can check its submersion property by considering the path

$$\gamma : \mathbb{R} \to \mathcal{S}_\mu, \quad \gamma(t)_{(\mu)} = P_{r,\mu}\begin{bmatrix} A_\mu & B_\mu \\ C_\mu & D_\mu + tE_\mu \end{bmatrix} P_{c,\mu}(I_{n_{\mu+1}\cdots n_d} \otimes U_{\mu-1}^\mathsf{T} \otimes \cdots \otimes U_1^\mathsf{T})$$

with arbitrary matrix $E_\mu \in \mathbb{R}^{(n_\mu - r_\mu)\times(r_1\cdots r_{\mu-1}n_{\mu+1}\cdots n_d - r_\mu)}$ and hence

$$D\Phi_\mu(\mathbf{X})[\gamma'(0)] = (\Phi_\mu \circ \gamma)'(t)\Big|_{t=0} = \frac{d}{dt}\Big|_{t=0}(D_\mu + tE_\mu - C_\mu A_\mu^{-1}B_\mu) = E_\mu.$$

Using that $\mathcal{S}_\mu$ is a relatively open subset of $\mathcal{M}^{1,\dots,\mu-1}_{r_1,\dots,r_{\mu-1}}$ and thus $\dim(\mathcal{S}_\mu) = \dim(\mathcal{M}^{1,\dots,\mu-1}_{r_1,\dots,r_{\mu-1}})$

we obtain that $\mathcal{M}^{1,\ldots,\mu}_{r_1,\ldots,r_\mu}$ is an embedded submanifold of $\mathcal{M}^{1,\ldots,\mu-1}_{r_1,\ldots,r_{\mu-1}}$ of dimension

$$\begin{aligned}
\dim \mathcal{M}^{1,\ldots,\mu}_{r_1,\ldots,r_\mu} &= \dim \ker \Phi_\mu \\
&= \dim(\mathcal{S}_\mu) - \dim(\mathbb{R}^{(n_\mu - r_\mu)\times(r_1\cdots r_{\mu-1} n_{\mu+1}\cdots n_d - r_\mu)}) \\
&= \left(\prod_{\nu=1}^{\mu-1} r_\nu\right)\left(\prod_{\nu=\mu}^{d} n_\nu\right) + \sum_{\nu=1}^{\mu-1} r_\nu n_\nu - r_\nu^2. \\
&\quad - \left(\prod_{\nu=1}^{\mu-1} r_\nu\right)\left(\prod_{\nu=\mu}^{d} n_\nu\right) + \left(\prod_{\nu=1}^{\mu} r_\nu\right)\left(\prod_{\nu=\mu+1}^{d} n_\nu\right) + n_\mu r_\mu - r_\mu^2 \\
&= \left(\prod_{\nu=1}^{\mu} r_\nu\right)\left(\prod_{\nu=\mu+1}^{d} n_\nu\right) + \sum_{\nu=1}^{\mu}(r_\nu n_\nu - r_\nu^2).
\end{aligned}$$

As $\mathcal{M}^{1,\ldots,\mu}_{r_1,\ldots,r_\mu}$ is identical to $\mathcal{M}_{\mathbf{r}}$, we obtain that $\mathcal{M}_{\mathbf{r}}$ is an embedded submanifold of $\mathcal{M}^{1,\ldots,\mu-1}_{r_1,\ldots,r_{\mu-1}}$ and thus also of $\mathbb{R}^{n_1\times\cdots\times n_d}$ of the same dimension

$$\prod_{\mu=1}^{d} r_\mu + \sum_{\mu=1}^{d}(n_\mu r_\mu - r_\mu^2).$$

$\square$

Note that the embedded submanifold property does not follow directly from the intersection of the d rank-r_μ matrix manifolds $\mathcal{M}^{\mu}_{r_\mu}$ corresponding to the matricizations $\mathbf{X}_{(\mu)}$, $\mu = 1,\ldots,d$. This follows from that fact that they do not *intersect transversally*, as the dimension of the intersection of their tangent spaces $T_{\mathbf{X}}\mathcal{M}^{\mu}_{r_\mu}$ is smaller than the dimension of the full space $\mathbb{R}^{n_1\times\cdots\times n_d}$. Indeed, the μth tangent space has dimensionality

$$\dim(T_{\mathbf{X}}\mathcal{M}^{\mu}_{r_\mu}) = r_\mu(n_\mu + \prod_{\substack{\nu=1\\ \nu\neq\mu}}^{d} n_\nu) - r_\mu^2,$$

from which we can conclude that

$$\dim(T_{\mathbf{X}}\mathcal{M}^{\mu}_{r_\mu} \cap\cdots\cap T_{\mathbf{X}}\mathcal{M}^{\mu}_{r_\mu}) \leq \sum_{\mu=1}^{d} r_\mu(n_\mu + \prod_{\substack{\nu=1\\ \nu\neq\mu}}^{d} n_\nu) - r_\mu^2 < \prod_{\mu=1}^{d} n_\mu = \dim(\mathbb{R}^{n_1\times\cdots\times n_d}),$$

if we assume that $r_\mu < n_\mu$.

Furthermore, we note that $\mathcal{M}_{\mathbf{r}}$ can also be seen as a quotient manifold. Let $\mathrm{St}(n,r)$ denote the Stiefel manifold of $n\times r$ matrices with orthonormal columns. We set $\mathcal{M}_{\mathbf{r}} = \mathcal{N}_{\mathbf{r}}/\sim$ with the total space $\mathcal{N}_{\mathbf{r}}$ given as the product

$$\mathcal{N}_{\mathbf{r}} = \mathbb{R}^{r_1\times\cdots\times r_d} \times \mathrm{St}(n_1,r_1)\times\cdots\times\mathrm{St}(n_d,r_d),$$

and $\sim$ represents the equivalence relation due to the non-uniqueness of the Tucker decomposition, see Remark 3.2. This viewpoint was investigated recently by Kasai and Mishra [KM15].

3.5. The tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$

To find a suitable parametrization of the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$, we choose an arbitrary smooth curve γ in the manifold passing through $\mathbf{X} = \mathbf{S} \times_1 U_1 \cdots \times_d U_d$. As $\gamma(t) \in \mathcal{M}_{\mathbf{r}}$, we can describe it by smoothly varying components $\mathbf{S}(t)$ and $U_\mu(t)$ in the Tucker format,

$$\begin{aligned}\gamma : \mathbb{R} \to \mathcal{M}_{\mathbf{r}}, \quad & \gamma(t) = \mathbf{S}(t) \times_1 U_1(t) \times_2 U_2(t) \cdots \times_d U_d(t),\\ & \gamma(0) = \mathbf{X}.\end{aligned}$$

The tangent vector realized by this curve is then obtained by the product rule,

$$\begin{aligned}\gamma'(0) &= \mathbf{S}'(0) \times_1 U_1(0) \times_2 U_2(0) \cdots \times_d U_d(0)\\ &\quad + \mathbf{S}(0) \times_1 U_1'(0) \times_2 U_2(0) \cdots \times_d U_d(0)\\ &\quad + \cdots + \mathbf{S}(0) \times_1 U_1(0) \times_2 U_2(0) \cdots \times_d U_d'(0)\\ &= \mathbf{S}'(0) \times_1 U_1 \times_2 U_2 \cdots \times_d U_d\\ &\quad + \mathbf{S} \times_1 U_1'(0) \times_2 U_2 \cdots \times_d U_d\\ &\quad + \cdots + \mathbf{S} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d'(0),\end{aligned}$$

where we have used that $\mathbf{S}(0) = \mathbf{S}$ and $U_\mu(0) = U_\mu$ due to the initial condition $\gamma(0) = \mathbf{X}$. Thus, any tangent vector $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be written as

$$\begin{aligned}\xi &= \mathbf{S} \times_1 \delta U_1 \times_2 U_2 \ \cdots \times_d U_d \ + \ \mathbf{S} \times_1 U_1 \times_2 \delta U_2 \ \cdots \times_d U_d \ + \ \cdots\\ &\quad + \ \mathbf{S} \times_1 U_1 \times_2 U_2 \ \cdots \times_d \delta U_d \ + \ \delta\mathbf{S} \times_1 U_1 \times_2 U_2 \ \cdots \times_d U_d\\ &= \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1\\ \nu\neq\mu}}^{d} U_\nu + \delta\mathbf{S} \bigtimes_{\nu=1}^{d} U_\nu,\end{aligned} \tag{3.12}$$

with arbitrary first-order variations $\delta U_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ and $\delta\mathbf{S} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$. From Proposition 2.4, we know that $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is a vector space of the same dimension as $\mathcal{M}_{\mathbf{r}}$. By counting dimensions, we see that the representation (3.12) has

$$\prod_{\mu=1}^{d} r_\mu + \sum_{\mu=1}^{d} n_\mu r_\mu > \dim(\mathcal{M}_{\mathbf{r}})$$

degrees of freedom. Introducing additional orthogonality constraints, the so-called *gauge conditions*,

$$\delta U_\mu^{\mathsf{T}} U_\mu = 0 \quad \forall \mu = 1, \ldots, d$$

introduces r_μ^2 constraints per factor matrix U_μ, such that the number of degrees of freedom matches $\dim(\mathcal{M}_{\mathbf{r}})$. Indeed, this representation of tangent vectors allows us to decompose the tangent space into orthogonal subspaces, as the next proposition shows.

Proposition 3.7. *The tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be orthogonally decomposed as*

$$T_{\mathbf{X}}\mathcal{M} = \mathcal{V}_1 \oplus \mathcal{V}_2 \oplus \cdots \oplus \mathcal{V}_d \oplus \mathcal{V}_{d+1}, \quad \textit{with } \mathcal{V}_\mu \perp \mathcal{V}_\nu \ \forall\, \mu \neq \nu, \tag{3.13}$$

where the subspaces $\mathcal{V}_\mu$ are given by

$$\mathcal{V}_\mu = \left\{ \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1\\ \nu\neq\mu}}^{d} U_\nu \;\middle|\; \delta U_\mu \in \mathbb{R}^{n_\mu \times r_\mu},\ \delta U_\mu^\mathsf{T} U_\mu = 0 \right\}, \qquad \mu = 1, \dots, d, \tag{3.14}$$

and

$$\mathcal{V}_{d+1} = \left\{ \delta\mathbf{S} \bigtimes_{\nu=1}^{d} U_\nu \;\middle|\; \delta\mathbf{S} \in \mathbb{R}^{r_1 \times \cdots \times r_d} \right\}.$$

Proof. To prove the orthogonal decomposition, we have to show for any pair $\mathcal{V}_\mu$, $\mathcal{V}_\nu$ with $\mu, \nu \in \{1, \dots, d\}$, $\mu \neq \nu$, it has to hold that $\langle \mathbf{X}, \mathbf{Y} \rangle = 0$ for all $\mathbf{X} \in \mathcal{V}_\mu$, $\mathbf{Y} \in \mathcal{V}_\nu$. Thus we calculate

$$\begin{aligned}
\langle \mathbf{X}, \mathbf{Y} \rangle &= \left\langle \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\kappa=1\\ \kappa\neq\mu}}^{d} U_\kappa,\ \mathbf{S} \times_\nu \delta U_\nu \bigtimes_{\substack{\tau=1\\ \tau\neq\nu}}^{d} U_\tau \right\rangle \\
&= \left\langle \mathbf{S} \times_\mu U_\mu^\mathsf{T} \delta U_\mu \times_\nu \delta U_\nu^\mathsf{T} U_\nu \bigtimes_{\substack{\kappa=1\\ \kappa\neq\mu\\ \kappa\neq\nu}}^{d} U_\kappa^\mathsf{T} U_\kappa,\ \mathbf{S} \right\rangle \\
&= \left\langle \mathbf{S} \times_\mu U_\mu^\mathsf{T} \delta U_\mu \times_\nu \delta U_\nu^\mathsf{T} U_\nu,\ \mathbf{S} \right\rangle = 0,
\end{aligned}$$

where we used the rules for the inner product, see Section 3.3.3 and the orthogonality conditions of the factor matrices. Analogously, we obtain for $\mathbf{X} \in \mathcal{V}_\mu$, $\mathbf{Y} \in \mathcal{V}_{d+1}$

$$\begin{aligned}
\langle \mathbf{X}, \mathbf{Y} \rangle &= \left\langle \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\kappa=1\\ \kappa\neq\mu}}^{d} U_\kappa,\ \delta\mathbf{S} \bigtimes_{\nu=1}^{d} U_\nu \right\rangle \\
&= \left\langle \mathbf{S} \times_\mu U_\mu^\mathsf{T} \delta U_\mu,\ \delta\mathbf{S} \right\rangle = 0,
\end{aligned}$$

from which we obtain that also $\mathcal{V}_\mu \perp \mathcal{V}_{d+1}$ for all $\mu = 1, \dots, d$, which proves the proposition. ☐

In particular, this orthogonal decomposition shows that, given the core tensor $\mathbf{S}$ and factor matrices U_μ of $\mathbf{X}$, the tangent vector ξ is *uniquely* represented in terms of $\delta\mathbf{S}$ and the *gauged* δU_μ, in contrast to the non-unique representation in the Tucker format.

3.5.1. Projection onto the tangent space

An explicit expression for the projection onto the tangent space $T_\mathbf{X}\mathcal{M}_\mathbf{r}$ at a point $\mathbf{X} \in \mathcal{M}_\mathbf{r}$ was derived by Koch and Lubich, [KL10, Eq.(2.7)] and is shown in the following proposition.

Proposition 3.8. *For an arbitrary* $\mathbf{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$*, the orthogonal projection onto the tangent space at* $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}, \mathbf{X} = \mathbf{S} \times_1 U_1 \cdots \times_d U_d$*, with* $U_\mu^\mathsf{T} U_\mu = I_{r_\mu}$*, is given by*

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}, \quad \mathbf{Z} \mapsto \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu + \delta \mathbf{S} \bigtimes_{\nu=1}^{d} U_\nu,$$

where the components δU_μ *and* $\delta \mathbf{S}$ *are determined by*

$$\delta \mathbf{S} = \mathbf{Z} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T},$$

$$\delta U_\mu = (I_{n_\mu} - U_\mu U_\mu^\mathsf{T}) \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^\mathsf{T} \Big]_{(\mu)} \mathbf{S}_{(\mu)}^\dagger,$$

where $\mathbf{S}_{(\mu)}^\dagger = \mathbf{S}_{(\mu)}^\mathsf{T} \big(\mathbf{S}_{(\mu)} \mathbf{S}_{(\mu)}^\mathsf{T} \big)^{-1}$ *is the Moore–Penrose pseudo-inverse of* $\mathbf{S}_{(\mu)}$*.*

Proof. The orthogonal projection operator has to fulfill

$$\langle \mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} \mathbf{Z}, \xi \rangle = \langle \mathbf{Z}, \xi \rangle, \quad \forall \xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}.$$

To simplify this expression, we make use of the orthogonal decomposition of the tangent space into the subspaces $\mathcal{V}_\mu$, see Prop. 3.7 to consider each projection $\mathrm{P}_{\mathcal{V}_\mu}$, $\mu = 1, \ldots, d+1$ individually:

$$\langle \mathrm{P}_{\mathcal{V}_\mu} \mathbf{Z}, \xi_\mu \rangle = \langle \mathbf{Z}, \xi_\mu \rangle, \quad \forall \xi_\mu \in \mathcal{V}_\mu.$$

Let us first consider the easiest case, $\mu = d+1$, the projection onto $\mathcal{V}_{d+1}$. We can write any $\xi_{d+1} \in \mathcal{V}_{d+1}$ as $\xi_{d+1} = \mathbf{R} \times_1 \mathbf{U}_1 \cdots \times_d \mathbf{U}_d$ with arbitrary $\mathbf{R} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$. Thus, it has to hold

$$\begin{aligned}
\langle \mathrm{P}_{\mathcal{V}_{d+1}} \mathbf{Z}, \xi_{d+1} \rangle &= \langle \mathbf{Z}, \xi_{d+1} \rangle \\
\Leftrightarrow \quad \Big\langle \delta \mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu,\ \mathbf{R} \bigtimes_{\mu=1}^{d} U_\mu \Big\rangle &= \Big\langle \mathbf{Z},\ \mathbf{R} \bigtimes_{\mu=1}^{d} U_\mu \Big\rangle \\
\Leftrightarrow \quad \Big\langle \delta \mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T} U_\mu,\ \mathbf{R} \Big\rangle &= \Big\langle \mathbf{Z} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T},\ \mathbf{R} \Big\rangle \\
\Leftrightarrow \quad \langle \delta \mathbf{S},\ \mathbf{R} \rangle &= \Big\langle \mathbf{Z} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T},\ \mathbf{R} \Big\rangle
\end{aligned}$$

for all $\mathbf{R}$, from which we conclude that $\delta \mathbf{S} = \mathbf{Z} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T}$. For the remaining subspaces $\mathcal{V}_\mu$ with $\mu = 1, \ldots, d$ we test with

$$\xi_\mu = \mathbf{S} \times_\mu \mathrm{P}_{U_\mu}^\perp V \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu \in \mathcal{V}_\mu,$$

where we have added a projection $\mathrm{P}_{U_\mu}^\perp := I_{n_\mu} - U_\mu U_\mu^\mathsf{T}$ into the orthogonal complement of U_μ, such that the gauge condition of the μth factor matrix is automatically fulfilled for arbitrary choices of $V \in \mathbb{R}^{n_\mu \times r_\mu}$. With this ansatz, we want to determine δU_μ such that

$$\begin{aligned} \langle \mathrm{P}_{\mathcal{V}_\mu} \mathbf{Z}, \xi_\mu \rangle &= \langle \mathbf{Z}, \xi_\mu \rangle \\ \Leftrightarrow \quad \Big\langle \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu,\ \mathbf{S} \times_\mu \mathrm{P}_{U_\mu}^{\perp} V \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu \Big\rangle &= \Big\langle \mathbf{Z},\ \mathbf{S} \times_\mu \mathrm{P}_{U_\mu}^{\perp} V \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu \Big\rangle \\ \Leftrightarrow \quad \langle \mathbf{S} \times_\mu \delta U_\mu,\ \mathbf{S} \times_\mu V \rangle &= \Big\langle \mathbf{Z} \times_\mu \mathrm{P}_{U_\mu}^{\perp} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}},\ \mathbf{S} \times_\mu V \Big\rangle \end{aligned}$$

holds for all $V \in \mathbb{R}^{n_\mu \times r_\mu}$ To get rid of the core tensor $\mathbf{S}$, we look at the μth matricization *(observing that* $\langle \mathbf{X}, \mathbf{Y} \rangle = \langle \mathbf{X}_{(\mu)}, \mathbf{Y}_{(\mu)} \rangle = \mathrm{trace}(\mathbf{X}_{(\mu)}^{\mathsf{T}} \mathbf{Y}_{(\mu)})$ *holds for the Euclidean inner product)*,

$$\begin{aligned} \Big\langle \delta U_\mu \mathbf{S}_{(\mu)},\ V \mathbf{S}_{(\mu)} \Big\rangle &= \Big\langle \mathrm{P}_{U_\mu}^{\perp} \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}} \Big]_{(\mu)},\ V \mathbf{S}_{(\mu)} \Big\rangle \\ \Leftrightarrow \quad \Big\langle \delta U_\mu \mathbf{S}_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}},\ V \Big\rangle &= \Big\langle \mathrm{P}_{U_\mu}^{\perp} \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}} \Big]_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}},\ V \Big\rangle \end{aligned}$$

from which we conclude that

$$\delta U_\mu \mathbf{S}_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}} = \mathrm{P}_{U_\mu}^{\perp} \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}} \Big]_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}}.$$

Multiplying from the right with $(\mathbf{S}_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}})^{-1}$ and using that $\mathbf{S}_{(\mu)}^{\mathsf{T}} (\mathbf{S}_{(\mu)} \mathbf{S}_{(\mu)}^{\mathsf{T}})^{-1} = \mathbf{S}_{(\mu)}^{\dagger}$, we obtain

$$\delta U_\mu = \mathrm{P}_{U_\mu}^{\perp} \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}} \Big]_{(\mu)} \mathbf{S}_{(\mu)}^{\dagger} = (I - U_\mu U_\mu^{\mathsf{T}}) \Big[\mathbf{Z} \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu^{\mathsf{T}} \Big]_{(\mu)} \mathbf{S}_{(\mu)}^{\dagger},$$

which proves the proposition. □

The projection of a Tucker tensor of multilinear rank $\tilde{\mathbf{r}}$ onto $T_{\mathbf{X}} \mathcal{M}_{\mathbf{r}}$ can be performed in $O(dn\tilde{r}r^{d-1} + \tilde{r}^d r)$ operations, where we assume $\tilde{r} \geq r$.

3.5.2. Tucker representation of tangent vectors

Every tangent vector $\xi \in T_{\mathbf{X}} \mathcal{M}_{\mathbf{r}}$

$$\xi = \delta \mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu + \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1 \\ \nu \neq \mu}}^{d} U_\nu,$$

can also be represented as a Tucker tensor

$$\xi = \mathbf{R} \times_1 V_1 \cdots \times_d V_d \tag{3.15}$$

of rank at most $2\mathbf{r}$, whose (non-orthogonalized) factor matrices are given by

$$V_\mu = \begin{bmatrix} U_\mu & \delta U_\mu \end{bmatrix} \in \mathbb{R}^{n_\mu \times 2r_\mu}$$

and the core tensor $\mathbf{R} \in \mathbb{R}^{2r_1 \times \cdots \times 2r_d}$ is composed of $\mathbf{S}$ and $\delta\mathbf{S}$ with a special block structure illustrated in Figure 3.5 for the case $d = 3$. This rank-$2\mathbf{r}$ representation of tangent vectors will be used when calculating the retraction and is also convenient for implementation purposes, as it allows us to reuse the implementation of operations acting on Tucker tensors.

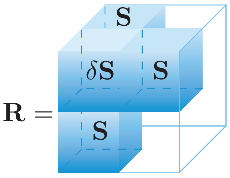


Figure 3.5: Block structure of the core tensor $\mathbf{R}$ *in the compact representation* (3.15) *of a tangent tensor* $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ *for the case* $d = 3$.

3.5.3. Inner product of two tangent vectors

Let $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$ with $\mathbf{X} = \mathbf{S} \times_1 U_1 \cdots \times_d U_d$. The inner product of two tangent tensors ξ, ν in $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$,

$$\xi = \delta\mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu + \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\nu\neq\mu} U_\nu,$$
$$\eta = \widetilde{\delta\mathbf{S}} \bigtimes_{\mu=1}^{d} U_\mu + \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \widetilde{\delta U}_\mu \bigtimes_{\nu\neq\mu} U_\nu,$$

can be evaluated efficiently by using the orthogonal decomposition, Prop. 3.7, to split the inner product into $(d+1)$ parts corresponding to the subspaces $\mathcal{V}_\mu$,

$$\begin{aligned}\langle \xi, \eta \rangle &= \left\langle \delta\mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu,\ \widetilde{\delta\mathbf{S}} \bigtimes_{\mu=1}^{d} U_\mu \right\rangle + \sum_{\mu=1}^{d} \left\langle \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\nu\neq\mu} U_\nu,\ \mathbf{S} \times_\mu \widetilde{\delta U}_\mu \bigtimes_{\nu\neq\mu} U_\nu \right\rangle \\ &= \langle \delta\mathbf{S}, \widetilde{\delta\mathbf{S}} \rangle + \sum_{\mu=1}^{d} \langle \mathbf{S} \times_\mu \widetilde{\delta U}_\mu^\mathsf{T} \delta U_\mu,\ \mathbf{S} \rangle.\end{aligned}$$

Thus, after evaluating the products $\widetilde{\delta U}_\mu^\mathsf{T} \delta U_\mu$ and multiplying the result along the μth mode of $\mathbf{S}$, $\mu = 1, \ldots, d$, we only have to compute $(d+1)$ small inner products of size $r_1 r_2 \cdots r^d$, for a total cost of $O(dr^2 n + r^{d+1} + (d+1)r^d)$ floating point operations.

3.6. Retraction

In the matrix case, the metric projection (2.4) can be easily computed from the truncated SVD. However, according to Theorem 3.4, the HOSVD is only quasi-optimal and thus cannot be used to compute the orthogonal projection.

Nevertheless, it still possesses all necessary properties of a retraction in the sense of Definition 2.16.

Proposition 3.9. *The map*

$$R : T\mathcal{M}_{\mathbf{r}} \to \mathcal{M}_{\mathbf{r}}, \quad (\mathbf{X}, \xi) \mapsto \mathrm{P}_{\mathbf{r}}^{\mathrm{HO}}(\mathbf{X} + \xi) \tag{3.16}$$

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

Proof. The map R defined in (3.16) can be written as the composition

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

where the smooth map $F : T\mathcal{M}_{\mathbf{r}} \to \mathbb{R}^{n_1 \times \cdots \times n_d}$ is defined as $F(\mathbf{X}, \xi) := \mathbf{X} + \xi$. By Proposition 3.5, $\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}}$ is smooth on the image on F as long as ξ is sufficiently small. Hence, R defines a locally smooth map in a neighborhood $\mathcal{U} \subset T\mathcal{M}_{\mathbf{r}}$ around $(\mathbf{X}, 0_{\mathbf{X}})$.

Definition 2.16 (b) follows from the fact that the application of the HOSVD to elements in $\mathcal{M}_{\mathbf{r}}$ leaves them unchanged.

It remains to check Definition 2.16 (c), the local rigidity condition. Because the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is a first order approximation of $\mathcal{M}_{\mathbf{r}}$ around ξ, we have that $\|(\mathbf{X} + t\xi) - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi)\| = O(t^2)$ for $t \to 0$. Thus, using Theorem 3.4:

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

Let us now write R in terms of the best approximation $\mathrm{P}_{\mathcal{M}_{\mathbf{r}}}$, see (2.4), from which we know that it constitutes a retraction:

$$R(\mathbf{X}, t\xi) = \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi) + E(\mathbf{X} + t\xi),$$

where E is the error term due to the non-optimality of the HOSVD, with

$$\begin{aligned}
\|E(\mathbf{X} + t\xi)\| &= \| \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi) - R(\mathbf{X}, t\xi)\| \\
&= \|(\mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi) - \mathbf{X} + t\xi) + (\mathbf{X} + t\xi - R(\mathbf{X}, t\xi))\| \\
&\leq \|\mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi) - \mathbf{X} + t\xi\| + \|\mathbf{X} + t\xi - R(\mathbf{X}, t\xi)\| \\
&= O(t^2).
\end{aligned}$$

Hence, the derivative fulfills

$$\begin{aligned}
\frac{d}{dt}R(\mathbf{X}, t\xi)\Big|_{t=0} &= \frac{d}{dt}\,\mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi)\Big|_{t=0} + \frac{d}{dt}E(\mathbf{X} + t\xi)\Big|_{t=0} \\
&= \xi + 0 = \xi,
\end{aligned}$$

and therefore $DR(\mathbf{X}, \cdot)(0_{\mathbf{X}}) = \mathrm{id}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}$, which completes the proof. □

To calculate the retraction, we make use of the rank-$2\mathbf{r}$ Tucker representation of a tangent vector $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ discussed in Section 3.5.2. Let $\alpha \in \mathbb{R}$ be the step size in one iteration of an optimization algorithm. The sum $\mathbf{Z} = \mathbf{X} + \alpha\xi$, interpreted as the usual sum in Euclidean space, can again be written in a compact way as a Tucker tensor with representation

$$\mathbf{Z} = \mathbf{R} \times_1 V_1 \times_2 V_2 \cdots \times_d V_d$$

of rank at most $2\mathbf{r}$, whose (non-orthogonalized) factor matrices are given by

$$V_\mu = \begin{bmatrix} U_\mu & \delta U_\mu \end{bmatrix} \in \mathbb{R}^{n_\mu \times 2r_\mu}$$

and the core tensor $\mathbf{R} \in \mathbb{R}^{2r_1 \times \cdots \times 2r_d}$ is composed of $\mathbf{S}$ and $\delta\mathbf{S}$ with the block structure illustrated in Figure 3.6 for the case $d = 3$. After orthogonalizing $\mathbf{Z}$ such that its factor matrices have orthonormal columns, we can calculate the retraction by truncating the rank of this compact representation from rank at most $2\mathbf{r}$ to $\mathbf{r}$ using the HOSVD procedure described in Algorithm 3.2. In total, performing the retraction $R(\mathbf{X}, \alpha\xi)$ can be done in $O(dr^2n + r^{d+1})$ operations.

Alternative retractions. In [AO14], Absil and Oseledets have investigated different retractions for the matrix case, i.e. the manifold of matrices of fixed rank. There, the orthogonal projection computed from the truncated SVD procedure is compared to other approaches, such as calculating the exponential map (using a Runge-Kutta scheme) or an orthographic projection, where we choose $R(\mathbf{X}, \xi)$ to be the point from

$$\mathbf{X} + \xi + N_\mathbf{X}\mathcal{M}_\mathbf{r} \cap \mathcal{M}_\mathbf{r}$$

closest to $\mathbf{X} + \xi$, see also [AM12]. This can be solved explicitly in the matrix case to obtain a retraction which does not require the calculation of the SVD but a matrix inversion instead. Obtaining a similar explicit expression in the tensor case is not straight-forward, as the structure of the normal space $N_\mathbf{X}\mathcal{M}_\mathbf{r}$ is more complicated. A different approach formulates the retraction map as a dynamic low-rank approximation problem, given as

$$\frac{d}{dt}\mathbf{X}_{k+1} = \mathbf{X}_k + t\xi, \quad \mathbf{X}_{k+1} \in \mathcal{M}_\mathbf{r} \quad \forall t \in [0, 1].$$

An efficient way to solve such an ODE is given by the KSL projector-splitting scheme [LO14], which has also been extended to the Tucker [Lub15] and TT case [LOV15]. As the comparison in [AO14] showed, the SVD-based retraction yields very good results and is thus our method of choice. Nevertheless, it would be interesting to search for a retraction which yields similar performance while removing the need for expensive HOSVDs.

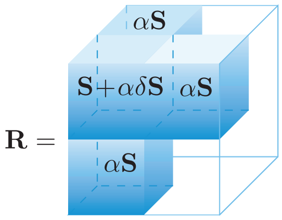


Figure 3.6: Block structure of the core tensor $\mathbf{R}$ *obtained from the addition* $\mathbf{X} + \alpha\xi$ *(seen as addition in Euclidean space) with* $\mathbf{X} \in \mathcal{M}_\mathbf{r}$ *and* $\xi \in T_\mathbf{X}\mathcal{M}_\mathbf{r}$ *for the case* $d = 3$.

3.7. Vector transport

For the conjugate gradient algorithm, we need to compare the current gradient ξ_i with the previous search direction η_{i-1}, living in two different tangent spaces. As described in Section 2.7, we can transport a tangent vector $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ to $T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}$ by means of the vector transport

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

Hence, we have to apply the projection operator $\mathrm{P}_{T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}}$ to a tangent vector $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$. Again, we can make use of the compact Tucker representation of ξ, see Section 3.5.2, and apply the projection operator to this Tucker tensor of multilinear rank $\widetilde{\mathbf{r}} \leq 2\mathbf{r}$ to obtain the transported tangent vector $\mathcal{T}_{\mathbf{X}\to\mathbf{Y}}\xi$. The total cost of this procedure is $O(dnr^d + r^{d+1})$ operations.

3.8. Linear operators acting on Tucker tensors

Many applications, such as the solution of linear systems, see Chapter 6, require the action of a linear operator $\mathcal{A}$ on a tensor $\mathbf{X}$. If $\mathbf{X}$ is given in a low-rank format, a natural question is: *in which cases can we preserve the low-rank structure of* $\mathbf{X}$ *after application of* $\mathcal{A}$*?*

A common case are linear operators $\mathcal{A} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{n_1\times\cdots\times n_d}$ which possess a Kronecker-structured matrix representation $A \in \mathbb{R}^{n_1 n_2\cdots n_d \times n_1 n_2 \cdots n_d}$,

$$\begin{aligned} \mathbf{Y} = \mathcal{A}\mathbf{X} \quad &\Leftrightarrow \mathrm{vec}(\mathbf{Y}) = A\,\mathrm{vec}(\mathbf{X}), \\ A = \sum_{i=1}^{R} &A_{d,i} \otimes A_{d-1,i} \otimes \cdots \otimes A_{1,i}, \end{aligned} \tag{3.17}$$

where each $A_{\mu,i}$ is a matrix of size $n_\mu \times n_\mu$ for $i = 1,\ldots,R$. The application of such a Kronecker structured operator A to a rank-$\mathbf{r}$ Tucker tensor $\mathbf{X} = \mathbf{S} \times_1 U_1 \cdots \times_d U_d$ is straightforward,

$$\mathbf{Y} = \sum_{i=1}^{R} \mathbf{S} \times_1 A_{1,i}U_1 \times_2 A_{2,i}U_2 \cdots \times_d A_{d,i}U_i.$$

As it is a sum of R rank-$\mathbf{r}$ tensors, the multilinear rank of $\mathbf{Y}$ is at most $R\mathbf{r}$.

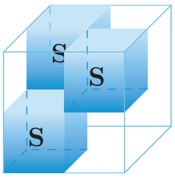
$\widetilde{\mathbf{S}} =$ 

Figure 3.7: Block structure of the core tensor $\widetilde{\mathbf{S}}$ *of* $\mathbf{Y} = \mathcal{A}\mathbf{X}$*, where* $\mathcal{A}$ *is the Laplace operator* (3.18)*, for the case* $d = 3$.

As an example, the d-dimensional Laplace operator discretized on a uniform tensor grid of mesh width h exhibits the form (3.17):

$$A = \sum_{\mu=1}^{d} I_{n_d} \otimes \cdots \otimes I_{n_{\mu+1}} \otimes L_\mu \otimes I_{n_{\mu-1}} \otimes \cdots \otimes I_{n_1}, \tag{3.18}$$

where $L_\mu = \frac{1}{h^2}\,\mathrm{tridiag}(-1, 2, -1)$ is the one-dimensional finite difference matrix. Making use of its special structure, we can show that applying the Laplace operator does not yield a tensor of rank $R\mathbf{r}$ but of rank $2\mathbf{r}$, independent of the number of dimensions: The factor matrices $\widetilde{U}_\mu$ of $\mathbf{Y}$ are determined by

$$\widetilde{U}_\mu = \begin{bmatrix} U_\mu & L_\mu U_\mu \end{bmatrix}$$

and the resulting core tensor $\widetilde{\mathbf{S}}$ of $\mathbf{Y}$ has a block structure, shown in Figure 3.7 for the case $d = 3$.

More complicated examples of such linear operators with Kronecker product structure will be discussed in the numerical experiments of Chapter 6.

Chapter 4

▶ Tensors of Fixed Tensor Train Rank

In many cases, the number of dimensions d of the solution tensor $\mathbf{X} \in \mathbb{R}^{n^d}$ can become quite large. Indeed, when studying linear systems or eigenvalue problems stemming from spin systems in quantum physics, d corresponds to the number of particles, so we can easily reach cases of $d > 100$. As we have seen in the previous Chapter 3, the dimensionality of the manifold of Tucker tensors of fixed multilinear rank $\mathbf{r} = (r, \ldots, r)$ scales like

$$O\big(r^d + d(nr^2 - r^2)\big).$$

While this is clearly a big improvement over the n^d degrees of freedom of the full tensor if $r \ll n$, we still have an exponential dependence on d: The term r^d quickly becomes too large to handle even for very small ranks r. Thus, the usefulness of the Tucker format is limited to only moderately high-dimensional cases with, lets say, $d < 5$. In this chapter, we will see a different notion of the tensor rank leading to the *Tensor Train* (TT) format as introduced by Oseledets and Tyrtyshnikov [OT09, Ose11c]. In the physics community, this format was already used earlier under the name *Matrix Product States* (MPS) [AKLT87, AKLT88, HWSH13, FNW92, OR95, Sch11, Whi92].

We proceed in a similar way as in the previous Chapter 3: We first introduce the TT format and basic operations on tensors in this representation. Then, we discuss the manifold property of the set of tensors of fixed TT rank, derive the corresponding tangent space and construct expressions for a retraction map and vector transport needed for Riemannian optimization.

4.1. Tensor train rank and format

In Section 3.1.1, we have seen the μth-mode matricization as a way to reshape a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ into a matrix. Of course, this is not the only way to matricize a tensor. In this chapter, we introduce another approach which we call the *μ-mode unfolding* to distinguish it from the μth-mode matricization. Note that in the literature, it is often also called μth matricization and the correct reshaping has to be deduced from the context.

Definition 4.1. *The* μth unfolding *of* $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ *is obtained by arranging the entries in a matrix*

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

where the corresponding index map is given by

$$\iota : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \cdots n_\mu} \times \mathbb{R}^{n_{\mu+1} \cdots n_d}, \quad \iota(i_1, \dots, i_d) = (i_{\text{row}}, i_{\text{col}}),$$

$$i_{\text{row}} = 1 + \sum_{\nu=1}^{\mu} (i_\nu - 1) \prod_{\tau=1}^{\nu-1} n_\tau, \quad i_{\text{col}} = 1 + \sum_{\nu=\mu+1}^{d} (i_\nu - 1) \prod_{\tau=\mu+1}^{\nu-1} n_\tau.$$

The new indices i_{row} *and* i_{col} *form a colexicographic ordering of* $\mathbb{R}^{n_1 \cdots n_\mu}$ *and* $\mathbb{R}^{n_{\mu+1} \cdots n_d}$*, respectively.*

We note that the μth matricization and the μth unfolding of a tensor $\mathbf{X}$ are only related for the first and last mode, $\mu = 1$ and $\mu = d$, as we have

$$\mathbf{X}_{(1)} = \mathbf{X}^{<1>}, \qquad \mathbf{X}_{(d)} = (\mathbf{X}^{<d-1>})^{\mathsf{T}}.$$

Remark 4.2. *The* μ*th unfolding is compatible with the column-major storage order usually used to represent matrices. If the tensor* $\mathbf{X}$ *is stored in memory as a long vector* $\operatorname{vec}(\mathbf{X}) \in \mathbb{R}^{n_1 \cdots n_d}$*, then unfolding the tensor into a matrix does not require moving entries in memory. Indeed, we have*

$$\operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{X}^{<\mu>}), \quad \mu = 1, \dots, d-1.$$

which can be exploited in an efficient implementation. Note that for the μ*th matricization, this is only true for the first mode,* $\operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{X}_{(1)})$*.*

The unfoldings allow us to define the *tensor train rank.*

Definition 4.3. *The* tensor train rank *or* TT rank *of a tensor tensor* $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *is defined as the following* $(d+1)$*-tuple of ranks,*

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

where we have set $r_0 = r_d = 1$*.*

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

$$\mathbf{X}(i_1, i_2, \dots, i_d) = U_1(i_1) U_2(i_2) \cdots U_d(i_d), \tag{4.1}$$

where each $U_\mu(i_\mu)$ is a matrix of size $r_{\mu-1} \times r_\mu$ for $i_\mu = 1, 2, \dots, n_\mu$. We can view the matrices $U_\mu(i_\mu)$ as slices of a third order tensor $\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$ along the second dimension,

$$\mathbf{U}_\mu(:, i_\mu, :) = U_\mu(i_\mu), \quad i_\mu = 1, 2, \dots, n_\mu.$$

The $\mathbf{U}_\mu$ are called the *cores* of the TT format. In terms of these core tensors, equation (4.1) can also be written as

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

To be able to exploit the product structure of (4.1), we now focus on ways to operate on parts of the cores and in particular, access and manipulate individual cores. We define the *left* and the *right unfoldings*,

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

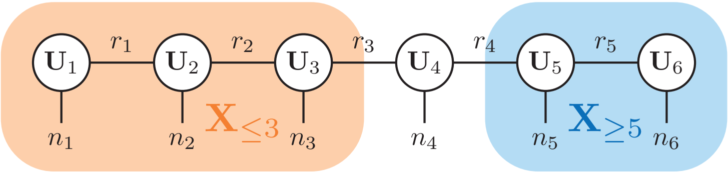


Figure 4.1: Tensor network diagram representation of a TT tensor of order $d = 6$. As $r_0 = r_d = 1$, they are usually omitted. The two interface matrices $\mathbf{X}_{\leq 3}$ and $\mathbf{X}_{\geq 5}$ are shown.

by reshaping $\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1}\times n_\mu \times r_\mu}$ into matrices of size $r_{\mu-1}n_\mu \times r_\mu$ and $r_{\mu-1} \times n_\mu r_\mu$, respectively. To rigorously define the index mapping, we can express the left and right unfoldings in terms of the matricization operations,

$$\mathbf{U}_\mu^{\mathsf{L}} = (\mathbf{U}_\mu)_{(3)}^{\mathsf{T}} = \mathbf{U}_\mu^{<2>}, \qquad \mathbf{U}_\mu^{\mathsf{R}} = (\mathbf{U}_\mu)_{(1)} = \mathbf{U}_\mu^{<1>}.$$

Additionally, we can split the tensor $\mathbf{X}$ into left and right parts, the *interface matrices*

$$\mathbf{X}_{\leq\mu}(i_1,\dots,i_\mu) = U_1(i_1)U_2(i_2)\cdots U_\mu(i_\mu), \quad \mathbf{X}_{\leq\mu} \in \mathbb{R}^{n_1 n_2\cdots n_\mu \times r_\mu},$$
$$\mathbf{X}_{\geq\mu}(i_\mu,\dots,i_d) = \left[U_\mu(i_\mu)U_{\mu+1}(i_{\mu+1})\cdots U_d(i_d)\right]^{\mathsf{T}}, \quad \mathbf{X}_{\geq\mu} \in \mathbb{R}^{n_\mu n_{\mu+1}\cdots n_d \times r_{\mu-1}}.$$

Note that here, we slightly abuse the notation, as $\mathbf{X}_{\leq\mu}(i_1,\dots,i_\mu)$ results in a row vector of length r_μ and not a single element of $\mathbf{X}_{\leq\mu}$. Analogously, $\mathbf{X}_{\geq\mu}(i_\mu,\dots,i_d)$ yields a row vector of length $r_{\mu-1}$. Figure 4.1 depicts a TT tensor of order 6 as a *tensor network diagram*, where we have marked the interface matrices $\mathbf{X}_{\leq 3}$ and $\mathbf{X}_{\geq 5}$. The interface matrices can be constructed recursively:

$$\mathbf{X}_{\leq\mu} = (I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1})\mathbf{U}_\mu^{\mathsf{L}}, \quad \text{and} \quad \mathbf{X}_{\geq\mu}^{\mathsf{T}} = \mathbf{U}_\mu^{\mathsf{R}}(\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \otimes I_{n_\mu}), \tag{4.2}$$

where we define $\mathbf{X}_{\leq 0} = \mathbf{X}_{\geq d+1} = 1$ such that for the first and last interface matrix it holds

$$\mathbf{X}_{\leq 1} = \mathbf{U}_1^{\mathsf{L}}, \quad \mathbf{X}_{\geq d} = (\mathbf{U}_\mu^{\mathsf{R}})^{\mathsf{T}}.$$

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

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

Inserting (4.2) and vectorizing, we can isolate the μth core:

$$\begin{aligned}
\operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{X}^{<\mu-1>}) &= \operatorname{vec}(\mathbf{X}_{\leq\mu-1}\mathbf{X}_{\geq\mu}^{\mathsf{T}}) \\
&= \operatorname{vec}\left(\mathbf{X}_{\leq\mu-1}\mathbf{U}_\mu^{\mathsf{R}}(\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \otimes I_{n_\mu})\right) \\
&= \operatorname{vec}\left(\mathbf{X}_{\leq\mu-1}(\mathbf{U}_\mu)_{(1)}(\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \otimes I_{n_\mu})\right) \\
&= \operatorname{vec}(\mathbf{U}_\mu \times_1 \mathbf{X}_{\leq\mu-1} \times_3 \mathbf{X}_{\geq\mu+1}) \\
&= (\mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1}) \operatorname{vec}(\mathbf{U}_\mu).
\end{aligned}$$

Finally, introducing the shorthand notation

$$\mathbf{X}_{\neq\mu} = \mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1} \in \mathbb{R}^{n_1 n_2\cdots n_d \times r_{\mu-1}n_\mu r_\mu}, \tag{4.3}$$

we obtain the handy expression

$$\operatorname{vec}(\mathbf{X}) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu). \tag{4.4}$$

4.2. Operations on TT tensors

4.2.1. Orthogonalization

We call $\mathbf{X}$ μ-orthogonal if

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

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

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

Thus, starting from the left, we compute a QR decomposition of the first core, $\mathbf{U}_1^{\mathsf{L}} = \mathbf{Q}_1^{\mathsf{L}} R_1$. The orthogonal factor is kept as the new, left-orthogonal first core and the non-orthogonal part is shifted to the second core. Then, a QR decomposition of this updated second core is performed, and so on and so forth. A similar relation holds for the right-orthogonal part. The resulting scheme is shown in Algorithm 4.1. Moving from a μ-orthogonal tensor to a $(\mu+1)$-orthogonal one only requires a QR decomposition of $\mathbf{U}_\mu^{\mathsf{L}}$. An important consequence that we will use in Section 4.4 is that changing the orthogonalization of a tensor only affects its internal representation and does not change the tensor itself. The computation of such an orthogonalized representation from an arbitrary rank-$\mathbf{r}$ TT tensor $\mathbf{X}$ can be performed in $O(dnr^3)$ operations.

Algorithm 4.1 μ-orthogonalization of a TT tensor

```
Input: Index $\mu \in \{1, 2, \dots, d\}$, TT Tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ represented as
  $\mathbf{X}(i_1, \dots, i_d) = U_1(i_1) U_2(i_2) \cdots U_d(i_d)$,
Output: $\mu$-orthogonal $\mathbf{X}$.
  for $\nu = 1, 2, \dots, \mu-1$ do
    $[Q, R] \leftarrow \mathrm{qr}(\mathbf{U}_\nu^{\mathsf{L}})$          % Left-orthogonalize up to $\mu-1$
    $\mathbf{U}_\nu^{\mathsf{L}} \leftarrow Q$
    $\mathbf{U}_{\nu+1} \leftarrow \mathbf{U}_{\nu+1} \times_1 R$
  end for
  for $\nu = d, d-1, \dots, \mu+1$ do
    $[Q, R] \leftarrow \mathrm{qr}((\mathbf{U}_\nu^{\mathsf{R}})^{\mathsf{T}})$          % Right-orthogonalize down to $\mu+1$
    $\mathbf{U}_\nu^{\mathsf{R}} \leftarrow Q^{\mathsf{T}}$
    $\mathbf{U}_{\nu-1} \leftarrow \mathbf{U}_{\nu-1} \times_3 R$
  end for
```

4.2.2. Addition

Consider the two tensors $\mathbf{X}$, $\mathbf{Y}$, with $\operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) = \mathbf{r}$, $\operatorname{rank}_{\mathrm{TT}}(\mathbf{Y}) = \widetilde{\mathbf{r}}$ represented in the TT format as

$$\begin{aligned}\mathbf{X}(i_1, i_2, \dots, i_d) &= U_1(i_1) U_2(i_2) \cdots U_d(i_d), \\ \mathbf{Y}(i_1, i_2, \dots, i_d) &= \widetilde{U}_1(i_1) \widetilde{U}_2(i_2) \cdots \widetilde{U}_d(i_d).\end{aligned} \tag{4.5}$$

Then, $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ can be represented as a TT tensor of rank $\mathbf{r} + \widetilde{\mathbf{r}}$,

$$\mathbf{Z}(i_1, i_2, \dots, i_d) = V_1(i_1) V_2(i_2) \cdots V_d(i_d),$$

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

$$V_1(i_1) = \begin{bmatrix} U_1(i_1) & \widetilde{U}_1(i_1) \end{bmatrix}, \; V_\mu(i_\mu) = \begin{bmatrix} U_\mu(i_\mu) & 0 \\ 0 & \widetilde{U}_\mu(i_\mu) \end{bmatrix}, \; V_d(i_d) = \begin{bmatrix} U_d(i_d) \\ \widetilde{U}_d(i_d) \end{bmatrix}.$$

This identity can be easily verified by multiplying out the matrix product:

$$\begin{aligned}\mathbf{Z}(i_1, i_2, \dots, i_d) &= V_1(i_1) V_2(i_2) \cdots V_d(i_d) \\ &= \begin{bmatrix} U_1(i_1) & \widetilde{U}_1(i_1) \end{bmatrix} \begin{bmatrix} U_2(i_2) & 0 \\ 0 & \widetilde{U}_2(i_2) \end{bmatrix} \cdots \begin{bmatrix} U_{d-1}(i_{d-1}) & 0 \\ 0 & \widetilde{U}_{d-1}(i_{d-1}) \end{bmatrix} \begin{bmatrix} U_d(i_d) \\ \widetilde{U}_d(i_d) \end{bmatrix} \\ &= U_1(i_1) U_2(i_2) \cdots U_d(i_d) + \widetilde{U}_1(i_1) \widetilde{U}_2(i_2) \cdots \widetilde{U}_d(i_d) \\ &= \mathbf{X}(i_1, i_2, \dots, i_d) + \mathbf{Y}(i_1, i_2, \dots, i_d).\end{aligned}$$

Thus, adding two TT tensors does not require any arithmetic operations, but increases the rank.

4.2.3. Inner product and norm

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

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \langle \operatorname{vec}(\mathbf{X}), \operatorname{vec}(\mathbf{Y}) \rangle. \tag{4.6}$$

To compute the inner product efficiently, we first note the identity

$$\begin{aligned}\operatorname{vec}(\mathbf{X}) = \mathbf{X}_{\neq 1} \operatorname{vec}(\mathbf{U}_1) &= (\mathbf{X}_{\geq 2} \otimes I_{n_1} \otimes \mathbf{X}_{\leq 0}) \operatorname{vec}(\mathbf{U}_1) \\ &= (\mathbf{X}_{\geq 2} \otimes I_{n_1}) \operatorname{vec}(\mathbf{U}_1) = \operatorname{vec}(\mathbf{U}_1 \times_3 \mathbf{X}_{\geq 2}).\end{aligned}$$

Then, we can reformulate the inner product as

$$\begin{aligned}\langle \operatorname{vec}(\mathbf{X}), \operatorname{vec}(\mathbf{Y}) \rangle &= \operatorname{vec}(\mathbf{U}_2)^{\mathsf{T}} \mathbf{X}_{\neq 2}^{\mathsf{T}} \mathbf{Y}_{\neq 2} \operatorname{vec}(\mathbf{V}_2) \\ &= \operatorname{vec}(\mathbf{U}_2)^{\mathsf{T}} \big(\mathbf{X}_{\geq 3}^{\mathsf{T}} \otimes I_{n_2} \otimes I_{r_1}\big) \big(\mathbf{Y}_{\geq 3} \otimes I_{n_2} \otimes \mathbf{X}_{\leq 1}^{\mathsf{T}} \mathbf{Y}_{\leq 1}\big) \operatorname{vec}(\mathbf{V}_2) \\ &= \operatorname{vec}\left(\mathbf{U}_2 \times_3 \mathbf{X}_{\geq 3}\right)^{\mathsf{T}} \operatorname{vec}\left(\mathbf{V}_2 \times_1 \mathbf{X}_{\leq 1}^{\mathsf{T}} \mathbf{Y}_{\leq 1} \times_3 \mathbf{Y}_{\geq 3}\right) \\ &= \operatorname{vec}\big(\widetilde{\mathbf{X}}\big)^{\mathsf{T}} \operatorname{vec}\big(\widetilde{\mathbf{Y}}\big) = \langle \widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}} \rangle,\end{aligned}$$

where $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ are the $(d-1)$ dimensional tensors obtained by removing the first core of $\mathbf{X}$ and $\mathbf{Y}$ (with modified second core):

$$\begin{aligned}\widetilde{\mathbf{X}}(i_2,\dots,i_d) &= U_2(i_2)U_3(i_3)\cdots U_d(i_d) = \mathbf{X}_{\geq 2}(i_2,\dots,i_d)^{\mathsf{T}},\\ \widetilde{\mathbf{Y}}(i_2,\dots,i_d) &= \widetilde{V}_2(i_2)V_3(i_3)\cdots V_d(i_d), \quad \widetilde{\mathbf{V}}_2 = \mathbf{V}_2 \times_1 \mathbf{X}_{\leq 1}^{\mathsf{T}}\mathbf{Y}_{\leq 1} = \mathbf{V}_2 \times_1 (\mathbf{U}_1^{\mathsf{L}})^{\mathsf{T}}\mathbf{V}_1^{\mathsf{L}}.\end{aligned}$$

Thus, we can calculate the inner product by moving from the left to right, $\mu = 1,\dots,d$, calculating the small $r_\mu \times \widetilde{r}_\mu$ matrices $(\mathbf{U}_\mu^{\mathsf{L}})^{\mathsf{T}}\mathbf{V}_\mu^{\mathsf{L}}$ and multiplying the result to the next core $\mathbf{V}_{\mu+1}$, see Algorithm 4.2. In total, the computation of the inner product requires $O(d(r^2\widetilde{r}n + r\widetilde{r}^2 n))$ operations. Note that one can analogously derive a right-to-left procedure involving the matrices $\mathbf{U}_\mu^{\mathsf{R}}(\mathbf{V}_\mu^{\mathsf{R}})^{\mathsf{T}}$ for $\mu = d, d-1,\dots,1$.

Algorithm 4.2 Inner product of two TT tensors (left-to-right procedure)

```
Input: Tensors $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$, $\mathbf{Y} \in \mathcal{M}_{\widetilde{\mathbf{r}}}$ with $\mathbf{X}(i_1,\dots,i_d) = U_1(i_1)U_2(i_2)\cdots U_d(i_d)$,
  $\mathbf{Y}(i_1,\dots,i_d) = V_1(i_1)V_2(i_2)\cdots V_d(i_d)$
Output: Inner product $p = \langle \mathbf{X}, \mathbf{Y} \rangle$.
  $p \leftarrow (\mathbf{U}_1^{\mathsf{L}})^{\mathsf{T}}\mathbf{V}_1^{\mathsf{L}}$
  for $\mu = 2,\dots,d$ do
    $\mathbf{V}_\mu \leftarrow \mathbf{V}_\mu \times_1 p$
    $p \leftarrow (\mathbf{U}_\mu^{\mathsf{L}})^{\mathsf{T}}\mathbf{V}_\mu^{\mathsf{L}}$
  end for
```

Remark 4.4. *Due to this iterative process of "reducing" the number of dimensions, this process is also known as a* tensor contraction*, especially when only performed partially, that is, only for certain modes instead of all* $\mu = 1,\dots,d$.

The norm of a TT tensor is induced by the inner product. If $\mathbf{X}$ is μ-orthogonal, then $\mathbf{X}_{\neq\mu}^{\mathsf{T}}\mathbf{X}_{\neq\mu} = I_{r_{\mu-1}n_\mu r_\mu}$ and we obtain

$$\begin{aligned}\|\mathbf{X}\| = \sqrt{\langle \mathbf{X}, \mathbf{X}\rangle} &= \sqrt{\operatorname{vec}(\mathbf{U}_\mu)^{\mathsf{T}}\mathbf{X}_{\neq\mu}^{\mathsf{T}}\mathbf{X}_{\neq\mu}\operatorname{vec}(\mathbf{U}_\mu)}\\ &= \sqrt{\operatorname{vec}(\mathbf{U}_\mu)^{\mathsf{T}}\operatorname{vec}(\mathbf{U}_\mu)} = \|\mathbf{U}_\mu\|,\end{aligned}$$

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

4.2.4. Hadamard product

As shown in [Ose11c], the *Hadamard* or *element-wise product* $\mathbf{Z} := \mathbf{X} \star \mathbf{Y}$ of two TT tensors $\mathbf{X}$ and $\mathbf{Y}$ of the form (4.5) can also be performed in an efficient way by exploiting the structure of the TT format. We have

$$\begin{aligned}\mathbf{Z}(i_1,\dots,i_d) &= \mathbf{X}(i_1,\dots,i_d)\mathbf{Y}(i_1,\dots,i_d)\\ &= \mathbf{X}(i_1,\dots,i_d) \otimes \mathbf{Y}(i_1,\dots,i_d)\\ &= (U_1(i_1)U_2(i_2)\cdots U_d(i_d)) \otimes (\widetilde{U}_1(i_1)\widetilde{U}_2(i_2)\cdots \widetilde{U}_d(i_d))\\ &= \left(U_1(i_1)\otimes \widetilde{U}_1(i_1)\right)\left(U_2(i_2)\otimes \widetilde{U}_2(i_2)\right)\cdots\left(U_d(i_d)\otimes \widetilde{U}_d(i_d)\right),\end{aligned}$$

where we first used that the Kronecker product is just the normal multiplication when applied to the scalar quantities $\mathbf{X}(i_1,\dots,i_d)$ and $\mathbf{Y}(i_1,\dots,i_d)$ and then distributed the Kronecker product to each factor of the TT representation. Hence, $\mathbf{Z}$ is a TT tensor with ranks at most $r_\mu \widetilde{r}_\mu$, $\mu = 1, \dots, d-1$, and cores

$$V_\mu(i_\mu) = U_\mu(i_\mu) \otimes \widetilde{U}_\mu(i_\mu), \quad \mu = 1, \dots, d.$$

As a result, the Hadamard product of $\mathbf{X}$ and $\mathbf{Y}$ can be computed in $O(dn(r\widetilde{r})^2)$ operations.

4.2.5. Rank truncation and TT-SVD

The *TT-SVD* [Ose11c] uses successive rank-r_μ truncations, $\mu = 1, \dots, d-1$, of a given tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ to obtain a TT representation of $\mathbf{X}$. This is similar to the HOSVD procedure 3.3.5, but instead of performing truncated SVDs in the μth *matricization*, we apply them to the μth *unfoldings* of $\mathbf{X}$. This yields

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

where each projector $\widetilde{\mathrm{P}}^{\mu}_{r_\mu}$ yields the best rank-r_μ approximation and is given by $(\widetilde{\mathrm{P}}^{\mu}_{r_\mu}\mathbf{X})^{<\mu>} = QQ^{\mathsf{T}}\mathbf{X}^{<\mu>}$, where $Q \in \mathbb{R}^{n_\mu \times r_\mu}$ consists of the first r_μ left singular vectors of $\mathbf{X}^{<\mu>}$. Again, just as the HOSVD, the TT-SVD yields a quasi-optimal approximation but with a different constant.

Theorem 4.5 (c.f. [Ose11c, Cor. 2.4])**.** *The truncated $\mathrm{P}^{TT}_{\mathbf{r}}\mathbf{X}$ fulfills a quasi-best approximation property,*

$$\|\mathbf{X} - \mathrm{P}^{\mathrm{TT}}_{\mathbf{r}}\mathbf{X}\| \leq \sqrt{d-1}\|\mathbf{X} - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X})\|, \tag{4.7}$$

where $\mathrm{P}_{\mathbf{r}}(\mathbf{X})$ is the projection yielding any best approximation to $\mathbf{X}$ within the set of rank-$\mathbf{r}$ TT tensors.

The existence of a best rank-$\mathbf{r}$ approximation $\mathrm{P}_{\mathbf{r}}\mathbf{X}$ is guaranteed, see [Usc13, Cor. 7.2].

The TT-SVD inherits the smoothness of low-rank matrix approximations.

Proposition 4.6 (Smoothness of truncated TT-SVD)**.** *Let $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$. Then there exists a neighborhood $\mathcal{U} \subset \mathbb{R}^{n_1 \times \cdots \times n_d}$ of $\mathbf{X}$ such that $\mathrm{P}^{\mathrm{TT}}_{\mathbf{r}} : \mathcal{U} \to \mathcal{M}_{\mathbf{r}}$ is smooth.*

Proof. The proof is analogous to the HOSVD case, see Proposition 3.5, but with the projectors $\widetilde{\mathrm{P}}_{r_\mu}$ instead of P_{r_μ}. ☐

If $\mathbf{X}$ is given in the TT-format with $\mathrm{rank}_{\mathrm{TT}}(\mathbf{X}) = \mathbf{r}$, then the TT-SVD truncation to a prescribed target rank $\mathbf{r}$ can be performed efficiently using the recursive relation (4.2). Let $\mathbf{X}$ be d-orthogonal. By taking the SVD of $\mathbf{U}^{\mathsf{R}}_d = QSV^{\mathsf{T}}$ and splitting the product to the adjacent cores:

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

the tensor is now $(d-1)$-orthogonalized. By keeping only $\widetilde{r}_{d-1}$ singular values, the rank r_{d-1} between $\mathbf{U}_{d-1}$ and $\mathbf{U}_d$ is reduced to $\widetilde{r}_{d-1}$. Note that due to the orthogonality of the other cores, the truncated SVD of $\mathbf{U}^{\mathsf{R}}_d$ corresponds to a truncated SVD of $\mathbf{X}^{<d>}$. Repeating this procedure till core $\mathbf{U}_1$ allows us to truncate the rank $\mathbf{r}$ of $\mathbf{X}$ to any prescribed value $\tilde{\mathbf{r}}$, with $\tilde{r}_\mu \leq r_\mu$ for all $\mu = 1, \dots, d$.

4.3. Manifold of TT tensors of fixed rank

In Section 3.4 we investigated the smooth manifold structure of the set of Tucker tensors of fixed multilinear rank. An analogous result for the TT case is available [HRS12b, UV13]. As the proof is rather technical, it will be omitted here.

Theorem 4.7. *The set of TT tensors of fixed TT rank,*

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

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

$$\dim \mathcal{M}_{\mathbf{r}} = \sum_{\mu=1}^{d} r_{\mu-1} n_\mu r_\mu - \sum_{\mu=1}^{d-1} r_\mu^2. \tag{4.8}$$

The inner product of Section 4.2.3 induces an Euclidean metric on the embedding space $\mathbb{R}^{n_1 \times \cdots \times n_d}$. When equipped with this metric, $\mathcal{M}_{\mathbf{r}}$ is a Riemannian manifold. A similar result is also available for the more general hierarchical Tucker format [UV13].

4.4. The tangent space

As we have seen in Subsection 4.2.1, the internal representation of a TT tensor is not unique. Thus, we can assume that $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is represented as a d-orthogonal TT tensor

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

that is, with left-orthogonal cores $\mathbf{U}_\mu$, $\left(\mathbf{U}_\mu^{\mathsf{L}}\right)^{\mathsf{T}} \mathbf{U}_\mu^{\mathsf{L}} = I_{r_\mu}$, for all $\mu = 1, \ldots, d-1$. To parametrize the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ at this point $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$, we take an arbitrary curve γ in $\mathcal{M}_{\mathbf{r}}$ through $\mathbf{X}$,

$$\begin{aligned} \gamma : \mathbb{R} \to \mathcal{M}_{\mathbf{r}}, \quad \gamma(t)(i_1, i_2, \ldots, i_d) &= U_1(t)(i_1) U_2(t)(i_2) \cdots U_d(t)(i_d), \\ \gamma(0)(i_1, i_2, \ldots, i_d) &= \mathbf{X}. \end{aligned}$$

By the product rule, this curve realizes the tangent vector

$$\begin{aligned} \gamma'(0)(i_1, i_2, \ldots, i_d) = {} & \delta U_1(t)(i_1) U_2(i_2) U_3(i_3) \cdots U_d(i_d) \\ & + U_1(i_1) \delta U_2(t)(i_2) U_3(i_3) \cdots U_d(i_d) \\ & + \cdots + U_1(i_1) U_2(i_2) U_3(i_3) \cdots \delta U_d(t)(i_d), \end{aligned}$$

where we have used that $\mathbf{U}_\mu(0) = \mathbf{U}_\mu$ for all $\mu = 1, \ldots, d$ due to the initial condition $\gamma(0) = \mathbf{X}$. Hence, any element of the tangent space $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be represented as

$$\begin{aligned} \xi(i_1, i_2, \ldots, i_d) = {} & \delta U_1(i_1) U_2(i_2) U_3(i_3) \cdots U_d(i_d) \\ & + U_1(i_1) \delta U_2(i_2) U_3(i_3) \cdots U_d(i_d) \\ & + \cdots + U_1(i_1) U_2(i_2) U_3(i_3) \cdots \delta U_d(i_d), \qquad (4.9) \\ \Leftrightarrow \quad \operatorname{vec}(\xi) = {} & \sum_{\mu=1}^{d} \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta \mathbf{U}_\mu), \end{aligned}$$

where the first-order variations $\delta\mathbf{U}_\mu$ are arbitrary tensors of size $r_\mu \times n_\mu \times r_\mu$. Counting dimensions, notice that this representation has

$$\sum_{\mu=1}^{d} r_{\mu-1} n_\mu r_\mu > \dim(\mathcal{M}_\mathbf{r})$$

degrees of freedom. Thus, analogous to the Tucker case, we introduce a *gauge condition* to remove this overparametrization. In particular, the orthogonality constraints

$$(\mathbf{U}_\mu^\mathsf{L})^\mathsf{T} \delta\mathbf{U}_\mu^\mathsf{L} = 0 \quad \forall \mu = 1, \ldots, d-1$$

enforce r_μ^2 equations that have to hold for each core $\mathbf{U}_\mu$. The following proposition uses this structure to orthogonally decompose $T_\mathbf{X}\mathcal{M}_\mathbf{r}$ from which we can follow that any tangent vector ξ is *uniquely* represented in terms of the *gauged cores* δU_μ.

Proposition 4.8. *The tangent space* $T_\mathbf{X}\mathcal{M}_\mathbf{r}$ *can be orthogonally decomposed as*

$$T_\mathbf{X}\mathcal{M}_\mathbf{r} = \mathcal{V}_1 \oplus \mathcal{V}_2 \oplus \cdots \oplus \mathcal{V}_d, \quad \text{with } \mathcal{V}_\mu \perp \mathcal{V}_\nu \ \forall\, \mu \neq \nu, \tag{4.10}$$

where the subspaces $\mathcal{V}_\mu$ *are given for d-orthogonal* $\mathbf{X}$ *by*

$$\begin{aligned} \mathcal{V}_\mu &= \Big\{ \xi_\mu \ \Big|\ \operatorname{vec}(\xi_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu),\ \delta\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1}\times n_\mu \times r_\mu},\ \big(\mathbf{U}_\mu^\mathsf{L}\big)^\mathsf{T} \delta\mathbf{U}_\mu^\mathsf{L} = 0 \Big\}, \\ \mathcal{V}_d &= \Big\{ \xi_d \ \Big|\ \operatorname{vec}(\xi_d) = \mathbf{X}_{\neq d} \operatorname{vec}(\delta\mathbf{U}_d),\ \delta\mathbf{U}_d \in \mathbb{R}^{r_{d-1}\times n_d \times r_d} \Big\}. \end{aligned} \tag{4.11}$$

Proof. Choose $\mu, \nu \in \{1, \ldots, d\}$, $\mu < \nu$ and arbitrary $\xi_\mu \in \mathcal{V}_\mu, \xi_\nu \in \mathcal{V}_\nu$, given in TT representation by

$$\begin{aligned} \xi_\mu &= U_1(i_1) \cdots U_{\mu-1}(i_{\mu-1}) \delta U_\mu(i_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d), \\ \xi_\nu &= U_1(i_1) \cdots U_{\nu-1}(i_{\nu-1}) \delta U_\nu(i_\nu) U_{\nu+1}(i_{\nu+1}) \cdots U_d(i_d). \end{aligned}$$

Then, the orthogonality of the spaces $\mathcal{V}_\mu$ and $\mathcal{V}_\nu$ is equivalent to $\langle \xi_\mu, \xi_\nu \rangle = 0$. The calculation of the inner product as explained in Section 4.2.3 consecutively computes the product of the left unfolding of the cores, $(\mathbf{U}_\tau^\mathsf{L})^\mathsf{T}\mathbf{U}_\tau^\mathsf{L} = I_{r_{\tau-1}n_\tau}$ for $\tau = 1, \ldots, \mu-1$. As the μth core is reached, we calculate $(\delta\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}\mathbf{U}_\mu^\mathsf{L} = 0$ due to the gauge condition. Thus, the inner product between ξ_μ and ξ_ν is zero for all $\mu \neq \nu$. ☐

To summarize, we can write any tangent vector $\xi \in T_\mathbf{X}\mathcal{M}_\mathbf{r}$ in the convenient form

$$\operatorname{vec}(\xi) = \sum_{\mu=1}^{d} \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu) \ \in \mathbb{R}^{n_1\times\cdots\times n_d} \qquad \text{s.t. } (\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}\delta\mathbf{U}_\mu^\mathsf{L} = 0, \quad \forall \mu \neq d. \tag{4.12}$$

4.4.1. Projection onto the tangent space

For the orthogonal projection $\mathrm{P}_{T_\mathbf{X}\mathcal{M}_\mathbf{r}} : \mathbb{R}^{n_1\times\cdots\times n_d} \to T_\mathbf{X}\mathcal{M}_\mathbf{r}$ of an arbitrary tensor $\mathbf{Z} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, Lubich et al. [LOV15, p. 924] derived an explicit expression for the first order variations $\delta\mathbf{U}_\mu$.

Proposition 4.9. *For an arbitrary* $\mathbf{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$*, the orthogonal projection onto the tangent space at* $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$*, where* $\mathbf{X}(i_1, \ldots, i_d) = U_1(i_1) \cdots U_d(i_d)$ *is d-orthogonal, is given by*

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}, \quad \mathrm{vec}(\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} \mathbf{Z}) = \sum_{\mu=1}^{d} \mathbf{X}_{\neq \mu} \mathrm{vec}(\delta \mathbf{U}_\mu).$$

For $\mu = 1, \ldots, d-1$*, the components* $\delta \mathbf{U}_\mu$ *in this expression are given by*

$$\delta \mathbf{U}_\mu^{\mathsf{L}} = \left(I_{n_\mu r_{\mu-1}} - \mathbf{U}_\mu^{\mathsf{L}} (\mathbf{U}_\mu^{\mathsf{L}})^{\mathsf{T}}\right) (I_{n_\mu} \otimes \mathbf{X}_{\leq \mu-1}^{\mathsf{T}}) \mathbf{Z}^{<\mu>} \mathbf{X}_{\geq \mu+1} \left(\mathbf{X}_{\geq \mu+1}^{\mathsf{T}} \mathbf{X}_{\geq \mu+1}\right)^{-1}. \quad (4.13)$$

For $\mu = d$*, we have*

$$\delta \mathbf{U}_d^{\mathsf{L}} = (I_{n_d} \otimes \mathbf{X}_{\leq d-1}^{\mathsf{T}}) \mathbf{Z}^{<d>}. \quad (4.14)$$

Note that the explicit expressions are stated for the left unfolding $\delta \mathbf{U}_\mu^{\mathsf{L}}$*. From this,* $\delta \mathbf{U}_\mu$ *can be directly obtained by reshaping.*

Proof. The orthogonal projection operator has to fulfill

$$\langle \mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} \mathbf{Z}, \xi \rangle = \langle \mathbf{Z}, \xi \rangle, \quad \forall \xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}.$$

To simplify this equation, the orthogonal projection $\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}$ onto the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be decomposed in accordance with (4.10):

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}} = \mathrm{P}_{\mathcal{V}_1} + \mathrm{P}_{\mathcal{V}_2} + \cdots + \mathrm{P}_{\mathcal{V}_d},$$

where $\mathrm{P}_{\mathcal{V}_\mu}$ are orthogonal projections onto $\mathcal{V}_\mu$. Then the projection can be written as

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}(\mathbf{Z}) = \sum_{\mu=1}^{d} \mathrm{P}_{\mathcal{V}_\mu}(\mathbf{Z}) \quad \text{where} \quad \mathrm{vec}\left(\mathrm{P}_{\mathcal{V}_\mu}(\mathbf{Z})\right) = \mathbf{X}_{\neq \mu} \mathrm{vec}(\delta \mathbf{U}_\mu). \quad (4.15)$$

Thus, we obtain d conditions

$$\langle \mathrm{P}_{\mathcal{V}_\mu} \mathbf{Z}, \xi_\mu \rangle = \langle \mathbf{Z}, \xi_\mu \rangle, \quad \forall \xi_\mu \in \mathcal{V}_\mu.$$

Let us first consider the easiest case, $\mu = d$, the projection onto $\mathcal{V}_d$. We can write any $\xi_d \in \mathcal{V}_d$ as $\mathrm{vec}(\xi_d) = \mathbf{X}_{\neq d} \mathrm{vec}(\mathbf{V})$ with arbitrary $\mathbf{V} \in \mathbb{R}^{r_{d-1} \times n_d \times r_d}$. Thus, it has to hold

$$\begin{aligned}
&\langle \mathrm{vec}(\mathrm{P}_{\mathcal{V}_d} \mathbf{Z}), \mathrm{vec}(\xi_d) \rangle = \langle \mathrm{vec}(\mathbf{Z}), \mathrm{vec}(\xi_d) \rangle \\
\Leftrightarrow \quad &\left\langle \mathbf{X}_{\neq d} \mathrm{vec}(\delta \mathbf{U}_d),\ \mathbf{X}_{\neq d} \mathrm{vec}(\mathbf{V}) \right\rangle = \left\langle \mathrm{vec}(\mathbf{Z}),\ \mathbf{X}_{\neq d} \mathrm{vec}(\mathbf{V}) \right\rangle \\
\Leftrightarrow \quad &\left\langle \mathbf{X}_{\neq d}^{\mathsf{T}} \mathbf{X}_{\neq d} \mathrm{vec}(\delta \mathbf{U}_d),\ \mathrm{vec}(\mathbf{V}) \right\rangle = \left\langle \mathbf{X}_{\neq d}^{\mathsf{T}} \mathrm{vec}(\mathbf{Z}),\ \mathrm{vec}(\mathbf{V}) \right\rangle \\
\Leftrightarrow \quad &\left\langle \mathrm{vec}(\delta \mathbf{U}_d),\ \mathrm{vec}(\mathbf{V}) \right\rangle = \left\langle (I_{n_d} \otimes \mathbf{X}_{\leq d-1}^{\mathsf{T}}) \mathrm{vec}(\mathbf{Z}),\ \mathrm{vec}(\mathbf{V}) \right\rangle.
\end{aligned}$$

From which we obtain $\mathrm{vec}(\delta \mathbf{U}_d) = (I_{n_d} \otimes \mathbf{X}_{\leq d-1}^{\mathsf{T}}) \mathrm{vec}(\mathbf{Z})$ which is equivalent to

$$\delta \mathbf{U}_d^{\mathsf{L}} = (I_{n_d} \otimes \mathbf{X}_{\leq d-1}^{\mathsf{T}}) \mathbf{Z}^{<d>},$$

as $\mathbf{Z}^{<d>} = \mathrm{vec}(\mathbf{Z})$ and $\delta \mathbf{U}_d^{\mathsf{L}} = \mathrm{vec}(\delta \mathbf{U}_d)$ due to $r_d = 1$.

For $\mu = 1, \ldots, d-1$ we test with

$$\mathrm{vec}(\xi_\mu) = \mathbf{X}_{\neq\mu}\,\mathrm{vec}(\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}}) \in \mathcal{V}_\mu,$$

where we added the projection $\mathrm{P}^{\perp}_{\mu,\mathsf{L}} := I_{n_\mu r_{\mu-1}} - \mathbf{U}^{\mathsf{L}}_\mu(\mathbf{U}^{\mathsf{L}}_\mu)^{\mathsf{T}}$ onto the orthogonal complement of $\mathbf{U}^{\mathsf{L}}_\mu$, such that the gauge condition of the μth core tensor $\mathbf{U}_\mu$ is automatically fulfilled for arbitrary choices of $\mathbf{V} \in \mathbb{R}^{r_{\mu-1}\times n_\mu \times r_\mu}$. We determine $\delta\mathbf{U}_\mu$ such that

$$\begin{aligned}
&\Big\langle \mathbf{X}_{\neq\mu}\,\mathrm{vec}(\delta\mathbf{U}_\mu),\ \mathbf{X}_{\neq\mu}\,\mathrm{vec}(\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}})\Big\rangle = \Big\langle\,\mathrm{vec}(\mathbf{Z}),\ \mathbf{X}_{\neq\mu}\,\mathrm{vec}(\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}})\Big\rangle \\
&\quad\Leftrightarrow \Big\langle (\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes I_{r_{\mu}-1})\,\mathrm{vec}(\delta\mathbf{U}_\mu),\ \mathrm{vec}(\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}})\Big\rangle \\
&\qquad\qquad\qquad\qquad\qquad = \Big\langle (\mathbf{X}^{\mathsf{T}}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{X}^{\mathsf{T}}_{\leq\mu-1})\,\mathrm{vec}(\mathbf{Z}),\ \mathrm{vec}(\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}})\Big\rangle \\
&\quad\Leftrightarrow \Big\langle \delta\mathbf{U}^{\mathsf{L}}_\mu(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1}),\ \mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}}\Big\rangle = \Big\langle (I_{n_\mu} \otimes \mathbf{X}^{\mathsf{T}}_{\leq\mu-1})\mathbf{Z}^{<\mu>}\mathbf{X}_{\geq\mu+1},\ \mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}}\Big\rangle.
\end{aligned}$$

As $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$, $\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1} \in \mathbb{R}^{r_\mu \times r_\mu}$ is non-singular, we can introduce the constructive identity $(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})^{-1}\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1}$ and shift using the cyclic invariance of the trace inner product,

$$\begin{aligned}
&\Big\langle \delta\mathbf{U}^{\mathsf{L}}_\mu,\ \mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})\Big\rangle \\
&\qquad = \Big\langle (I_{n_\mu} \otimes \mathbf{X}^{\mathsf{T}}_{\leq\mu-1})\mathbf{Z}^{<\mu>}\mathbf{X}_{\geq\mu+1}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})^{-1},\ \mathrm{P}^{\perp}_{\mu,\mathsf{L}}\,\mathbf{V}^{\mathsf{L}}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})\Big\rangle.
\end{aligned}$$

Defining $\widetilde{\mathbf{V}}^{\mathsf{L}} := \mathbf{V}^{\mathsf{L}}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})$ and moving $\mathrm{P}^{\perp}_{\mu,\mathsf{L}}$ we obtain

$$\Leftrightarrow \quad \Big\langle \delta\mathbf{U}^{\mathsf{L}}_\mu,\ \widetilde{\mathbf{V}}^{\mathsf{L}}\Big\rangle = \Big\langle\,\mathrm{P}^{\perp}_{\mu,\mathsf{L}}(I_{n_\mu} \otimes \mathbf{X}^{\mathsf{T}}_{\leq\mu-1})\mathbf{Z}^{<\mu>}\mathbf{X}_{\geq\mu+1}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})^{-1},\ \widetilde{\mathbf{V}}^{\mathsf{L}}\Big\rangle.$$

As this equality has to hold for all $\mathbf{V} \in \mathbb{R}^{r_{\mu-1}\times n_\mu \times r_\mu}$ and thus for all $\widetilde{\mathbf{V}}^{\mathsf{L}} \in \mathbb{R}^{r_{\mu-1}n_\mu \times r_\mu}$, we obtain that

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

which proves the statement. □

Expression (4.13) is relatively straight-forward to implement using tensor contractions along the modes $1, \ldots, \mu-1, \mu+1, \ldots, d$. The projection of a tensor of TT rank $\tilde{\mathbf{r}}$ into $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can be performed in $O(dnr\tilde{r}^2)$ operations, where we assume $\tilde{r} \geq r$.

4.4.2. A different parametrization of the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$

Note that the inverse $(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})^{-1}$ appearing in (4.13) potentially leads to numerical instabilities or may not even be well-defined: For example, $\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1}$ is singular in case the right unfoldings $\mathbf{U}^{\mathsf{R}}_{\mu+1}, \ldots, \mathbf{U}^{\mathsf{R}}_d$ do not have full rank. To circumvent this ill-conditioning, we will work with a different parametrization of the tangent space first proposed in [KOS12]: Observe that in the representation (4.9), each individual summand is a TT tensor which *can be orthogonalized individually* without changing the overall tangent tensor. Hence, to obtain the μth summand, we can μ-orthogonalize $\mathbf{X}$,

$$\mathbf{X} = U_1(i_1)\cdots U_{\mu-1}(i_{\mu-1})\widetilde{U}_\mu(i_\mu)V_{\mu+1}(i_{\mu+1})\cdots V_d(i_d)$$

where the cores denoted by the letter U are left-orthogonal, V are right-orthogonal and $\widetilde{U}$ are neither left- nor right-orthogonal. Then, the resulting first-order variation is given by

$$U_1(i_1)\cdots U_{\mu-1}(i_{\mu-1})\delta\widetilde{U}_\mu(i_\mu)V_{\mu+1}(i_{\mu+1})\cdots V_d(i_d). \tag{4.16}$$

The first order variations $\delta\widetilde{\mathbf{U}}_\mu$ still fulfill the gauge condition $(\mathbf{U}^{\mathsf{L}}_\mu)^{\mathsf{T}}\delta\widetilde{\mathbf{U}}^{\mathsf{L}}_\mu = 0, \mu = 1,\ldots,d-1$, as we can write $\delta\mathbf{U}^{\mathsf{L}}_\mu = \delta\widetilde{\mathbf{U}}^{\mathsf{L}}_\mu R$, where $R \in \mathbb{R}^{r_\mu \times r_\mu}$ is the matrix corresponding to the non-orthogonal part left over from the reorthogonalization of $\mathbf{X}_{\geq\mu+1}$. As $\mathbf{X}_{\mu+1}$ is now right-orthogonal, the expression (4.13) for $\delta\mathbf{U}_\mu$ simplifies to

$$\delta\widetilde{\mathbf{U}}^{\mathsf{L}}_\mu = \left(I_{r_{\mu-1}n_\mu} - \mathbf{U}^{\mathsf{L}}_\mu(\mathbf{U}^{\mathsf{L}}_\mu)^{\mathsf{T}}\right)\left(I_{n_\mu}\otimes\mathbf{X}_{\leq\mu-1}\right)^{\mathsf{T}}\mathbf{Z}^{<\mu>}\mathbf{X}_{\geq\mu+1}, \tag{4.17}$$

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

$$\begin{aligned}\mathbf{Y}(i_1,\ldots,i_d) &= \delta\widetilde{U}_1(i_1)V_2(i_2)\cdots V_d(i_d)\\ &\quad + U_1(i_1)\delta\widetilde{U}_2(i_2)\cdots V_d(i_d)\\ &\quad + \cdots\\ &\quad + U_1(i_1)U_2(i_2)\cdots\delta\widetilde{U}_d(i_d).\end{aligned} \tag{4.18}$$

Thus, to compute the projection $\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}$ in this modified parametrization, we first d-orthogonalize $\mathbf{X}$ to obtain the left-orthogonal cores $\mathbf{U}_1,\ldots,\mathbf{U}_{d-1}$. Then we 1-orthogonalize $\mathbf{X}$ to obtain the right-orthogonal cores $\mathbf{V}_2,\ldots,\mathbf{V}_d$. Finally, we use (4.17) to obtain the modified first-order variations $\delta\widetilde{\mathbf{U}}_\mu$, $\mu = 1,\ldots,d-1$ and (4.14) to obtain $\delta\widetilde{\mathbf{U}}_d$. In algorithms, it is often possible to store both a 1- and d-orthogonal version of $\mathbf{X}$, such that this parametrization of the tangent space does not involve any extra costs. In the worst case, it requires an additional orthogonalization of $\mathbf{X}$, which can be performed in $O(dnr^3)$ operations.

4.4.3. TT representation of tangent vectors

Making use of the structure of the sum (4.18), we observe that a tangent vector in $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ can also be represented as a TT tensor:

Remark 4.10. *Let $\mathbf{Y} \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ be given in the representation (4.18). Then, it can be identified with a TT tensor of TT-rank at most $(1, 2r_1, 2r_2, \ldots, 2r_{d-1}, 1)$:*

$$\mathbf{Y}(i_1,\ldots,i_d) = W_1(i_1)W_2(i_2)\cdots W_d(i_d), \tag{4.19}$$

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

$$W_1(i_1) = \begin{bmatrix}\delta\widetilde{U}_1(i_1) & U_1(i_1)\end{bmatrix},\; W_\mu(i_\mu) = \begin{bmatrix}V_\mu(i_\mu) & 0\\ \delta\widetilde{U}_\mu(i_\mu) & U_\mu(i_\mu)\end{bmatrix},\; W_d(i_d) = \begin{bmatrix}V_d(i_d)\\ \delta\widetilde{U}_d(i_d)\end{bmatrix}.$$

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

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

4.4.4. Inner product of two tangent vectors

Due to the orthogonality of the $\mathcal{V}_\mu$ spaces, see Proposition 4.8, the inner product of two tangent tensors $\xi, \eta \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ decomposes into d inner products. Let ξ be represented by first-order variations $\delta\mathbf{U}_\mu$ and η by $\delta\mathbf{V}_\mu$, respectively. Then,

$$\begin{aligned}\langle \xi, \eta \rangle &= \Big\langle \sum_{\mu=1}^{d} \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu), \sum_{\nu=1}^{d} \mathbf{X}_{\neq\nu} \operatorname{vec}(\delta\mathbf{V}_\nu) \Big\rangle \\ &= \sum_{\mu=1}^{d} \big\langle \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu), \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{V}_\mu) \big\rangle.\end{aligned}$$

Let ξ and η now be in the μ-orthogonalized parametrization introduced in Section 4.4.2 with first order variations $\delta\widetilde{\mathbf{U}}_\mu$ and $\delta\widetilde{\mathbf{V}}_\mu$. Then these d inner products are particularly simple, as we have

$$\mathbf{X}_{\neq\mu}^{\mathsf{T}}\mathbf{X}_{\neq\mu} = (\mathbf{X}_{\geq\mu+1}^{\mathsf{T}}\mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{X}_{\geq\mu+1}^{\mathsf{T}}\mathbf{X}_{\geq\mu+1}) = I_{r_{\mu-1}n_\mu r_\mu}$$

and thus

$$\langle \xi, \eta \rangle = \sum_{\mu=1}^{d} \langle \delta\widetilde{\mathbf{U}}_\mu, \delta\widetilde{\mathbf{V}}_\mu \rangle.$$

which can be performed in $O(dnr^2)$ operations.

If ξ and η are not in the μ-orthogonalized parametrization, then they can be either reorthogonalized to move to this representation, or alternatively written as TT tensors of TT-rank $2\mathbf{r}$, see Section 4.4.3. The latter case is followed by the application of the inner product to TT tensors, see Section 4.2.3, with a total cost of $O(dnr^3)$.

4.5. Retraction

For $\mathcal{M}_{\mathbf{r}}$, a retraction maps the new point $\mathbf{X} + \xi \in \mathbb{R}^{n_1\times\cdots\times n_d}$ to a tensor of TT-rank $\mathbf{r}$. A computationally efficient method is given by the TT-SVD procedure, see Section 4.2.5 which satisfies the quasi-best approximation property (4.7). The proof of the following proposition is very similar to the Tucker case discussed in Section 3.6, where we showed that the HOSVD (which possesses an analogous quasi-best approximation property, Thm. 3.4) fulfills indeed all necessary properties of a retraction map.

Proposition 4.11. *Let $\mathrm{P}_{\mathbf{r}}^{\mathrm{TT}} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathcal{M}_{\mathbf{r}}$ denote the projection operator performing rank truncation using the TT-SVD. Then,*

$$R : T\mathcal{M}_{\mathbf{r}} \to \mathcal{M}_{\mathbf{r}}, \quad (\mathbf{X}, \xi) \mapsto \mathrm{P}_{\mathbf{r}}^{\mathrm{TT}}(\mathbf{X} + \xi) \tag{4.20}$$

defines a retraction on $\mathcal{M}_{\mathbf{r}}$ around $\mathbf{X}$ according to Def. 2.16.

Proof. We check the conditions *(a),(b),(c)* of Def. 2.16 which have to hold for a retraction.
(a) From Prop. 4.6, we obtain a small neighborhood of $\mathcal{U} \subset \mathbb{R}^{n_1 \times \cdots n_d}$ of $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$ such that the TT-SVD truncation operator $\mathrm{P}^{\mathrm{TT}}_{\mathbf{r}} : \mathcal{U} \to \mathcal{M}_{\mathbf{r}}$ is smooth. Thus, the retraction (4.20) is locally smooth around $(\mathbf{X}, 0_{\mathbf{X}})$, since it can be written as $R(\mathbf{X}, \xi) = \mathrm{P}^{\mathrm{TT}}_{\mathbf{r}} \circ p(\mathbf{X}, \xi)$ with the smooth function $p : T\mathcal{M}_{\mathbf{r}} \to \mathbb{R}^{n_1 \times \cdots \times n_d}, p(\mathbf{X}, \xi) = \mathbf{X} + \xi$.
(b) We have the obvious identity $\mathrm{P}^{\mathrm{TT}}_{\mathbf{r}}(\mathbf{X}) = \mathbf{X}$ for all $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$.
(c) We use the quasi-optimality (4.7). We have that $\|(\mathbf{X}+t\xi) - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X}+t\xi)\| = O(t^2)$ for $t \to 0$, as $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is a first-order approximation of $\mathcal{M}_{\mathbf{r}}$ around ξ. Hence,

$$\|(\mathbf{X} + t\xi) - R(\mathbf{X}, t\xi)\| \leq \sqrt{d-1}\|(\mathbf{X} + t\xi) - \mathrm{P}_{\mathcal{M}_{\mathbf{r}}}(\mathbf{X} + t\xi)\| = O(t^2).$$

Therefore, $R(\mathbf{X}, t\xi) = (\mathbf{X} + t\xi) + O(t^2)$ and $\frac{d}{dt} R(\mathbf{X}, t\xi)\Big|_{t=0} = \xi$, which is equivalent to $DR(\mathbf{X}, \cdot)(0_{\mathbf{X}}) = \mathrm{id}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}$. □

To efficiently apply the TT-SVD procedure to $\mathbf{X} + \xi$, we exploit the representation (4.18) and notice that the new point, obtained by a step $\mathbf{Y} \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ of length α from the point $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$, can be written as

$$\begin{aligned}\mathbf{X}(i_1, \ldots, i_d) + \alpha \mathbf{Y}(i_1, \ldots, i_d) &= \begin{bmatrix} \alpha\delta\widetilde{U}_1(i_1) & U_1(i_1) \end{bmatrix} \begin{bmatrix} V_2(i_2) & 0 \\ \alpha\delta\widetilde{U}_2(i_2) & U_2(i_2) \end{bmatrix} \cdots \\ &\cdots \begin{bmatrix} V_{d-1}(i_{d-1}) & 0 \\ \alpha\delta\widetilde{U}_{d-1}(i_{d-1}) & U_{d-1}(i_{d-1}) \end{bmatrix} \begin{bmatrix} V_d(i_d) \\ U_d(i_d) + \alpha\delta\widetilde{U}_d(i_d) \end{bmatrix}.\end{aligned}$$

Hence, retracting back to the rank-$\mathbf{r}$ manifold $\mathcal{M}$ is equivalent to truncating a rank-$2\mathbf{r}$ TT tensor to rank $\mathbf{r}$, for a total cost of approximately $O(dnr^3)$. Furthermore, the orthogonality relations between $\mathbf{U}_\mu$ and $\delta\mathbf{V}_\mu$ can be used to save some work when reorthogonalizing.

4.6. Vector transport

To compute the vector transport $\tau_{\mathbf{X}\to\mathbf{Y}}\mathbf{Z} = \mathrm{P}_{T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}} \mathbf{Z}$ of a certain tangent tensor $\mathbf{Z} \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ into $T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}$, we make use of the identity (4.19) to represent $\mathbf{Z}$ in the TT format,

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

Thus, calculation of the vector transport is reduced to the projection of a TT tensor of rank $2\mathbf{r}$ onto $T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}$. We have to compute the first order variations $\widetilde{\delta\mathbf{U}}$ of $\mathrm{P}_{T_{\mathbf{Y}}\mathcal{M}_{\mathbf{r}}}$ according to (4.17), where $\mathbf{U}_\mu$ denote the left- and $\mathbf{V}_\mu$ the right-orthogonal cores of $\mathbf{Y}$ as in Section 4.4.2:

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

which can be efficiently evaluated as a tensor contraction along dimensions $1, \dots d$ except mode μ. This can be checked using the recursive relations (4.2),

$$\begin{aligned}\mathbf{Z}_{\geq\mu+1}^{\mathsf{T}}\mathbf{Y}_{\geq\mu+1} &= \mathbf{W}_{\mu+1}^{\mathsf{R}}(\mathbf{Z}_{\geq\mu+2}^{\mathsf{T}} \otimes I_{n_{\mu+1}})(\mathbf{Y}_{\geq\mu+2} \otimes I_{n_{\mu+1}})(\mathbf{V}_{\mu+1}^{\mathsf{R}})^{\mathsf{T}} \\ &= \mathbf{W}_{\mu+1}^{\mathsf{R}}(\mathbf{Z}_{\geq\mu+2}^{\mathsf{T}}\mathbf{Y}_{\geq\mu+2} \otimes I_{n_{\mu+1}})(\mathbf{V}_{\mu+1}^{\mathsf{R}})^{\mathsf{T}}\end{aligned}$$

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

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

4.7. Linear operators acting on TT tensors

In the solution of linear systems, see Chapter 6, and eigenvalue problems, see Chapter 7, a linear operator $\mathcal{A} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{m_1\times\cdots\times m_d}$ is applied to a tensor $\mathbf{X}$ in the tensor train format.

In Section 3.8, we have seen the efficient application of Kronecker-structured linear operators to tensors in the Tucker format, preserving the low-rank structure. In the tensor train case, the so-called *operator TT format* or *TT-matrix* introduced in [Ose11b] allows for a convenient TT-like representation of linear operators.

As in (3.17), let A denote the matrix representation of $\mathcal{A}$, that is,

$$\mathbf{Y} = \mathcal{A}\mathbf{X} \quad \Leftrightarrow \operatorname{vec}(\mathbf{Y}) = A\operatorname{vec}(\mathbf{X}).$$

Analogous to the TT format, we then represent each entry of $A \in \mathbb{R}^{m_1m_2\cdots m_d\times n_1n_2\cdots n_d}$ as

$$A(\iota(i_1, \dots, i_d), \iota(j_1, \dots, j_d)) = A_1(i_1, j_1)A_2(i_2, j_2)\cdots A_d(i_d, j_d),$$

where ι maps the multiindex $(i_1, \dots, i_d)$ to a single index according to the standard vectorization index ordering, see (3.1). Each $A_\mu(i_\mu, j_\mu)$ is a matrix of size $R_{\mu-1} \times R_\mu$, where the R_μ denote the rank of the TT operator. Alternatively, we can also view the $A_\mu(i_\mu, j_\mu)$ as the slices of the 4-dimensional operator cores,

$$\mathbf{A}_\mu(:, i_\mu, j_\mu, :) = A_\mu(i_\mu, j_\mu), \quad \mathbf{A}_\mu \in \mathbb{R}^{R_{\mu-1}\times m_\mu\times n_\mu\times R_\mu}.$$

As a tensor network diagram, the linear operator $\mathcal{A}$ can be represented as a tensor train with an additional “leg” at each node, see Figure 4.2.

In principle, any sum of Kronecker products

$$A = \sum_{i=1}^{R} L_d^{(i)} \otimes \cdots \otimes L_2^{(i)} \otimes L_1^{(i)}$$

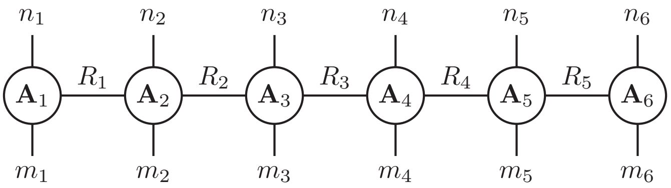


Figure 4.2: Representation of a TT operator of order $d = 6$. As $R_0 = R_d = 1$, they are usually omitted.

can be represented in the operator TT format with ranks at most $R_1 = \cdots = R_{d-1} = R$ by setting

$$A_1(i_1, j_1) = \begin{bmatrix} L_1^{(1)}(i_1, j_1) & L_1^{(2)}(i_1, j_1) & \dots & L_1^{(R)}(i_1, j_1) \end{bmatrix}, \quad A_d(i_d, j_d) = \begin{bmatrix} L_d^{(1)}(i_d, j_d) \\ L_d^{(2)}(i_d, j_d) \\ \vdots \\ L_d^{(R)}(i_d, j_d) \end{bmatrix},$$

and

$$A_\mu(i_\mu, j_\mu) = \begin{bmatrix} L_\mu^{(1)}(i_\mu, j_\mu) & & \\ & \ddots & \\ & & L_\mu^{(R)}(i_\mu, j_\mu) \end{bmatrix}, \quad \mu = 2, \dots, d-1.$$

In particular, if $A = L_d \otimes \cdots \otimes L_2 \otimes L_1$, the cores are given by the coefficients L_μ themselves and the operator TT ranks are all equal to one.

Application to a TT tensor. The tensor $\mathbf{Y}$ resulting from a “matrix-vector product” $\mathbf{Y} = \mathcal{A}\mathbf{X}$, with $\mathbf{X}$ in TT format and $\mathcal{A}$ in operator TT format, is again in TT format. Each element $\mathbf{Y}(i_1, \dots, i_d)$ is given by [Ose11c]

$$\begin{aligned} \mathbf{Y}(i_1, \dots, i_d) &= \sum_{j_1=1}^{n_1} \cdots \sum_{j_d=1}^{n_d} \mathcal{A}(i_1, \dots, i_d, j_1, \dots, j_d) \mathbf{X}(j_1, \dots, j_d) \\ &= \sum_{j_1=1}^{n_1} \cdots \sum_{j_d=1}^{n_d} A_1(i_1, j_1) A_2(i_2, j_2) \cdots A_d(i_d, j_d) U_1(j_1) U_2(j_2) \cdots U_d(j_d) \\ &= \sum_{j_1=1}^{n_1} \cdots \sum_{j_d=1}^{n_d} \Big(A_1(i_1, j_1) \otimes U_1(j_1) \Big) \cdots \Big(A_d(i_d, j_d) \otimes U_d(j_d) \Big). \end{aligned}$$

Thus, the cores $\mathbf{V}_\mu$ of $\mathbf{Y}$ can be computed entry-wise by

$$V_\mu(i_\mu) = \sum_{j_\mu=1}^{n_\mu} A_\mu(i_\mu, j_\mu) \otimes U_\mu(j_\mu).$$

For an efficient implementation, we do not form this Kronecker product for each entry of V_μ. Instead, we perform the summation over $j_k = 1, \dots, n_\mu$ by a matrix product of suitable matricizations,

$$Z_\mu = (\mathbf{A}_\mu)_{(3)}^\mathsf{T} (\mathbf{U}_\mu)_{(2)}, \quad \text{where } (\mathbf{A}_\mu)_{(3)}^\mathsf{T} \in \mathbb{R}^{R_{\mu-1} m_\mu R_\mu \times n_\mu}, \ (\mathbf{U}_\mu)_{(2)} \in \mathbb{R}^{n_\mu \times r_{\mu-1} r_\mu}.$$

The core $\mathbf{V}_\mu$ of $\mathbf{Y}$ is then obtained by reordering the $R_{\mu-1}m_\mu R_\mu \times r_{\mu-1}r_\mu$ matrix Z_μ into a tensor of size $R_{\mu-1}r_{\mu-1} \times m_\mu \times R_\mu r_\mu$. Thus, the TT ranks of $\mathbf{Y}$ are bounded by $R_\mu r_\mu$, $\mu = 1, \ldots, d-1$.

Chapter 5

▶ Application 1: Tensor Completion

In the previous chapters, we have introduced all necessary ingredients to apply Riemannian techniques to optimization problems with low-rank tensor structure. As a first application, we consider the completion of a d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in which the vast majority of entries are unknown. Depending on the source of the data set represented by $\mathbf{A}$, assuming an underlying low-rank structure is a sensible choice. In particular, this is the case for discretizations of functions that are well-approximated by a short sum of separable functions. Low-rank models have been shown to be effective for various applications, such as electronic structure calculations and approximate solutions of stochastic and parametric PDEs, see [GKT13] for an overview. With this assumption, the tensor completion problem can be cast into the optimization problem

$$\begin{aligned} \min_{\mathbf{X}} \quad & f(\mathbf{X}) := \frac{1}{2}\| \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A} \|^2 \\ \text{subject to} \quad & \mathbf{X} \in \mathcal{M}_{\mathbf{r}} := \left\{ \mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \;\middle|\; \operatorname{rank}(\mathbf{X}) = \mathbf{r} \right\}, \end{aligned} \tag{5.1}$$

where the definition of $\operatorname{rank}(\mathbf{X})$ depends on the employed tensor format, such as the multilinear (Tucker) rank introduced in Chapter 3 or the tensor train rank, see Chapter 4. We denote with P_Ω the projection onto the *sampling set* $\Omega \subset \{1, 2, \ldots, n_1\} \times \cdots \times \{1, 2, \ldots, n_d\}$ corresponding to the indices of the known entries of $\mathbf{A}$,

$$\mathrm{P}_\Omega \mathbf{X} := \begin{cases} \mathbf{X}(i_1, i_2, \ldots, i_d) & \text{if } (i_1, i_2, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

In the two-dimensional case $d = 2$, tensor completion (5.1) reduces to the extensively studied *matrix completion* for which convergence theory and efficient algorithms have been derived in the last years, see [MWG$^+$] for an overview. The number of entries in the optimization variable $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ scales exponentially with d, the so-called *curse of dimensionality*. Hence, the naive approach of reshaping the given data $\mathbf{A}$ into a large matrix and applying existing algorithms for matrix completion is computationally infeasible for all but very small problem sizes.

Assuming low-rank structure in the different tensor formats along with their notion of tensor rank allows us to reduce the degrees of freedom significantly. The most popular choice, the *Tucker* tensor format, reduces the storage complexity from $O(n^d)$ to $O(dnr + r^d)$ and thus yields good results for not too high-dimensional problems, say $d \leq 5$. Algorithms developed for the Tucker format range from alternating minimization [LMWY09, LS13] and convex optimization, using various generalizations of the nuclear norm [LMWY09, LS13, GRY11, SLS10, STL$^+$14, SPM$^+$11] to Riemannian optimization techniques [KM15]. As an instance of the last category, we present a tensor completion algorithm called GeomCG [KSV14] based on the manifold structure introduced in Section 3.4. In Section 5.2, we discuss the implementation details for the Tucker case and show the effectiveness of the proposed algorithm, for the three-dimensional case.

If higher dimensional problems are considered, the Tucker format can be replaced by the *hierarchical Tucker* (HT) or the TT format. In the TT format, the number of unknowns in a tensor with fixed rank r is scaling like $O(dnr^2 - dr^2)$, see Section 4.3, which potentially allows for the efficient treatment of problems with a much higher number of dimensions d than the Tucker format.

To tackle these higher dimensional problems, we extend our Riemannian optimization scheme to the TT case in Section 5.4. We obtain an algorithm called RTTC [Ste15] scaling linearly in the number of dimensions d, in the tensor size n and in the size of the sampling set $|\Omega|$ and scaling polynomially in the tensor train rank r.

For the HT format, Da Silva and Herrmann [DSH13, DSH15] have derived HTOpt, a closely related Riemannian approach. By exploiting the simpler structure of the TT format, RTTC exhibits a better computational complexity of $O(d|\Omega|r^2)$ per iteration instead of $O(d|\Omega|r^3)$ for HTOpt. Furthermore, our implementation is specifically tuned for very small sampling set sizes. In the TT format Grasedyck *et al.* [GKK13] presented an alternating direction method with similar computational complexity of $O(d|\Omega|r^2)$. We will compare these two methods to our algorithm RTTC and to a simple alternating least squares approach.

Note that for tensor completion, the index set Ω of given entries is fixed. If we relax this constraint and instead try to reconstruct the original tensor $\mathbf{A}$ from as few entries as possible, black-box approximation techniques such as *cross-approximation* are viable alternatives. In the matrix case, cross approximation was first developed for the approximation of integral operators [Beb00, BR03, BG05] and as the so-called *Skeleton decomposition* [GTZ97, Tyr00]. Extensions to high dimensional tensors were developed by Oseledets *et al.* [OT10, SO11] for the TT format and by Ballani *et al.* [BGK13, BG14] for the HT format. In these techniques, the sampling points are chosen adaptively during the runtime of the algorithm. In Section 5.5.5 we will compare them to our proposed tensor completion algorithm and show that choosing randomly distributed sampling points beforehand is competitive in certain applications.

The algorithms presented in this chapter are generalizations of the Riemannian matrix completion approach of Vandereycken [Van13] to a higher-dimensional setting using the Tucker and TT format. The content of this chapter is based on two reports for the Tucker [KSV14] and the TT format [Ste15], respectively. Additional material on rank adaptation schemes is provided in Section 5.3.4.

5.1. A Riemannian tensor completion algorithm

We apply a Riemannian nonlinear CG scheme, see Algorithm 2.3 in Section 2.8.3, to solve the tensor completion problem (5.1). This results in Algorithm 5.1 which either assumes a fixed multilinear *(GeomCG)* or tensor train rank *(RTTC)* of the underlying data. In this section, we discuss basic ingredients of the Riemannian tensor completion algorithms that are common to both the Tucker and TT case and discuss their general convergence properties.

Algorithm 5.1 Riemannian Nonlinear CG for Tensor Completion: GEOMCG (Tucker case) and RTTC (Tensor Train case)

```
Input: $\mathrm{P}_\Omega \mathbf{A}$, $\mathrm{P}_{\Omega_C} \mathbf{A}$, initial guess $\mathbf{X}_0 \in \mathcal{M}$, tolerance $\delta$.
Output: Sequence of iterates $(\mathbf{X}_k)$.
  $\mathbf{G}_0 \leftarrow \mathrm{P}_\Omega \mathbf{A} - \mathrm{P}_\Omega \mathbf{X}_0$                      % compute Euclidean gradient
  $\xi_0 \leftarrow \operatorname{grad} f(\mathbf{X}_0) = \mathrm{P}_{T_{\mathbf{X}_0}\mathcal{M}_\mathbf{r}} \mathbf{G}_0$       % compute Riemannian gradient
  $\eta_0 \leftarrow -\xi_0$                                        % first step is steepest descent
  $\alpha_0 \leftarrow -\langle \mathrm{P}_\Omega \eta_0, \mathbf{G}_0 \rangle / \| \mathrm{P}_\Omega \eta_0 \|^2$              % step size by linearized line-search
  $x_1 \leftarrow R(x_0, \alpha_0 \eta_0)$                              % obtain next iterate by retraction

  for $k = 1, 2, \dots$ maxiter do
    $\mathbf{G}_k \leftarrow \mathrm{P}_\Omega \mathbf{A} - \mathrm{P}_\Omega \mathbf{X}_k$                    % compute Euclidean gradient
    $\xi_0 \leftarrow \operatorname{grad} f(\mathbf{X}_k) = \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}} \mathbf{G}_k$     % compute Riemannian gradient
    $\eta_k \leftarrow -\xi_k + \beta_k \tau_{x_{k-1} \to x_k}(\eta_{k-1})$           % conjugate direction by update rule
    $\alpha_k \leftarrow -\langle \mathrm{P}_\Omega \eta_k, \mathbf{G}_k \rangle / \| \mathrm{P}_\Omega \eta_k \|^2$            % step size by linearized line-search
    $x_{k+1} \leftarrow R(x_k, \alpha_k \eta_k)$                         % obtain next iterate by retraction
    if $\varepsilon_\Omega < \delta$ or $\varepsilon_{\Omega_C} < \delta$ then
      Converged. Return.
    end if
    if Stagnation criteria (5.7) are fulfilled then
      Not converged. Return.
    end if
  end for
```

To obtain the descent direction in our optimization scheme, we need the Riemannian gradient of the objective function $f(\mathbf{X}) = \frac{1}{2}\|\mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}\|^2$. According to Section 2.4, the Riemannian gradient is obtained by projecting the Euclidean gradient

$$\operatorname{Grad} f(\mathbf{X}) = \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}$$

into the tangent space:

$$\operatorname{grad} f(\mathbf{X}) = \mathrm{P}_{T_\mathbf{X}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}). \tag{5.3}$$

As the Euclidean gradient $\mathbf{Z} = \mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a very large but sparse tensor, the projection into the tangent space has to be handled with care. The efficient implementation is specific to the employed tensor format and is discussed in Sections 5.2.1 and 5.4.1, respectively.

5.1.1. Linearized line search

We can derive a linearized line-search for the cost function f as described in Section 2.8.1. According to (2.7), the linearized step length α_k is obtained by

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}}\, f(\mathbf{X}_k + \alpha\eta_k),$$

which, due to the simple quadratic form of the cost function, can be calculated analytically by finding the root of the first derivative,

$$\frac{d}{d\alpha}\frac{1}{2}\| \mathrm{P}_\Omega(\mathbf{X} + \alpha\eta_k) - \mathrm{P}_\Omega \mathbf{A}\|^2 \overset{!}{=} 0,$$

from which we obtain

$$\alpha_k = \frac{\langle \mathrm{P}_\Omega \eta_k, \mathrm{P}_\Omega(\mathbf{A} - \mathbf{X}_k)\rangle}{\| \mathrm{P}_\Omega \eta_k\|^2} = -\frac{\langle \mathrm{P}_\Omega \eta_k, \operatorname{Grad} f(\mathbf{X}_k)\rangle}{\| \mathrm{P}_\Omega \eta_k\|^2}. \tag{5.4}$$

If needed, a standard Armijo backtracking scheme can be added to control the step sizes, using the result of this linearized line search procedure as an initial guess. Nevertheless, in all numerical experiments, the linearized line search performed so well that a backtracking step was never necessary and we propose to omit it to save some computational cost.

5.1.2. Stopping criteria

During the course of the optimization procedure, we monitor the current relative error on the sampling set,

$$\varepsilon_\Omega(\mathbf{X}_k) = \frac{\| \mathrm{P}_\Omega \mathbf{X}_k - \mathrm{P}_\Omega \mathbf{A}\|}{\| \mathrm{P}_\Omega \mathbf{A}\|}. \tag{5.5}$$

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

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

$$\varepsilon_{\Omega_C}(\mathbf{X}_k) = \frac{\| \mathrm{P}_{\Omega_C} \mathbf{X}_k - \mathrm{P}_{\Omega_C} \mathbf{A}\|}{\| \mathrm{P}_{\Omega_C} \mathbf{A}\|}. \tag{5.6}$$

This control set is either given or can be obtained by random subsampling of the given data. Numerical experiments show that it usually suffices to choose only few control samples, say, $|\Omega_C| = 100$. Algorithm 5.1 is run until either the prescribed tolerance is attained for $\varepsilon_\Omega(\mathbf{X}_k)$ and $\varepsilon_{\Omega_C}(\mathbf{X}_k)$ or the maximum number of iterations is reached. Furthermore, to detect stagnation in the optimization procedure, we check if the relative change in ε_Ω and ε_{Ω_C} between two iterates is below a certain threshold δ:

$$\frac{|\varepsilon_\Omega(\mathbf{X}_k) - \varepsilon_\Omega(\mathbf{X}_{k+1})|}{|\varepsilon_\Omega(\mathbf{X}_k)|} < \delta, \quad \frac{|\varepsilon_{\Omega_C}(\mathbf{X}_k) - \varepsilon_{\Omega_C}(\mathbf{X}_{k+1})|}{|\varepsilon_{\Omega_C}(\mathbf{X}_k)|} < \delta. \tag{5.7}$$

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

5.1.3. Convergence analysis

The convergence analysis of Algorithm 5.1 follows the same steps as the analysis employed in the matrix case [Van13]. First, to fulfill the assumptions of standard convergence results for Riemannian line search methods, we have to safeguard the heuristic linearized line search with an Armijo-type backtracking scheme. Let us assume that $\mathcal{M}_\mathbf{r}$ is either the manifold of tensors of fixed multilinear or TT rank. Then, we can apply Theorem 2.24 to obtain

Proposition 5.1. *Let* $(\mathbf{X}_k)_{k\in\mathbb{N}}$ *be an infinite sequence of iterates generated by Algorithm 5.1 with backtracking line search. Then, every accumulation point* $\mathbf{X}_* \in \mathcal{M}_\mathbf{r}$ *of* $(\mathbf{X}_k)$ *is a critical point of the cost function* f *and hence satisfies* $\operatorname{grad} f(\mathbf{X}_k) = 0$, *that is,* $\mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega\, \mathbf{X}_*) = \mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega\, \mathbf{A})$.

This result shows us that in the tangent space, reconstruction is achieved for all limit points of Algorithm 5.1. Unfortunately, as the set $\mathcal{M}_\mathbf{r}$ is not closed, $\mathbf{X}_*$ is not necessarily an element of $\mathcal{M}_\mathbf{r}$ anymore — and hence not a valid solution of problem (5.1). To enforce that $\mathbf{X}_* \in \mathcal{M}_\mathbf{r}$, we regularize the cost function in such a way that the iterates $\mathbf{X}_k$ stay inside a compact subset of $\mathcal{M}_\mathbf{r}$ so that we can apply Corollary 2.25. We define the modified cost function g with regularization parameter λ as

$$g : \mathcal{M}_\mathbf{r} \to \mathbb{R}, \quad \mathbf{X} \mapsto f(\mathbf{X}) + \lambda^2 \sum_{\mu=1}^{d} \left(\|\mathbf{X}_{(\mu)}\|_F^2 + \|(\mathbf{X}_{(\mu)})^\dagger\|_F^2 \right), \quad \lambda > 0, \tag{5.8}$$

if $\mathcal{M}_\mathbf{r}$ is the manifold of fixed multilinear rank and

$$g : \mathcal{M}_\mathbf{r} \to \mathbb{R}, \quad \mathbf{X} \mapsto f(\mathbf{X}) + \lambda^2 \sum_{\mu=1}^{d-1} \left(\|\mathbf{X}^{<\mu>}\|_F^2 + \|(\mathbf{X}^{<\mu>})^\dagger\|_F^2 \right), \quad \lambda > 0, \tag{5.9}$$

if $\mathcal{M}_\mathbf{r}$ is the manifold of fixed TT rank. With this modification, i.e. regularizing the singular values of the matricizations or unfoldings of $\mathbf{X}$, we obtain the following result:

Proposition 5.2. *Let* $(\mathbf{X}_k)_{k\in\mathbb{N}}$ *be an infinite sequence of iterates generated by Algorithm 5.1 but with the modified cost function* g *defined in* (5.8) *or* (5.9). *Then* $\lim_{k\to\infty} \|\operatorname{grad} g(\mathbf{X}_k)\| = 0$.

Proof. Let us first assume that $\mathcal{M}_\mathbf{r}$ is the manifold of fixed TT rank. The cost function is guaranteed to decrease due to the line search, $g(\mathbf{X}_k) \leq g(\mathbf{X}_0) =: C_0^2$, which yields $\lambda^2 \sum_{\mu=1}^d \|\mathbf{X}_k^{<\mu>}\|_F^2 \leq C_0^2$ and $\lambda^2 \sum_{\mu=1}^d \|(\mathbf{X}_k^{<\mu>})^\dagger\|_F^2 \leq C_0^2$. From this, we can deduce upper and lower bounds for the largest and smallest singular values of the matricizations,

$$\sigma_{\max}(\mathbf{X}_k^{<\mu>}) \leq \|\mathbf{X}_k^{<\mu>}\| \leq C_0\lambda^{-1}, \quad \sigma_{\min}^{-1}(\mathbf{X}_k^{<\mu>}) \leq \|(\mathbf{X}_k^{<\mu>})^\dagger\| \leq C_0\lambda^{-1}.$$

Therefore, all iterates are guaranteed to stay inside the compact set

$$B := \Big\{\mathbf{X} \in \mathcal{M}_\mathbf{r} \,\Big|\, \sigma_{\max}(\mathbf{X}^{<\mu>}) \leq C_0\lambda^{-1},\ \sigma_{\min}(\mathbf{X}^{<\mu>}) \geq C_0^{-1}\lambda,\ \mu = 1, \ldots, d\Big\}.$$

Now suppose, conversely to the statement of the proposition, that $\|\operatorname{grad} g(\mathbf{X}_k)\|$ does not converge to zero. Then there is a $\delta > 0$ and a subsequence of $\{\mathbf{X}_k\}$ such that $\|\operatorname{grad} g(\mathbf{X}_k)\| > \delta$ for all elements of the subsequence. Since $\mathbf{X}_k \in B$, it follows that this subsequence has an accumulation point $\mathbf{X}_*$ for which also $\|\operatorname{grad} g(\mathbf{X}_*)\| > \delta$. However, this contradicts Proposition 5.1 which states that every accumulation point is a critical point of g.

If $\mathcal{M}_\mathbf{r}$ is the manifold of fixed multilinear rank, then the proof is analogous, only replacing the unfolding $\mathbf{X}^{<\mu>}$ with the matricizations $\mathbf{X}_{(\mu)}$. □

Note that the regularization parameter λ can be chosen arbitrarily small. If the core unfoldings are always of full rank during the optimization procedure (even when $\lambda \to 0$), then the accumulation points $\mathbf{X}_*$ are guaranteed to stay inside $\mathcal{M}_\mathbf{r}$ and $\operatorname{grad} f(\mathbf{X}_*) = \mathrm{P}_{T_{\mathbf{X}_*}\mathcal{M}_\mathbf{r}}(\mathrm{P}_\Omega \mathbf{X}_* - \mathrm{P}_\Omega \mathbf{A}) \to 0$ as $\lambda \to 0$. In this case, optimizing the modified cost function (5.8) or (5.9) is equivalent to the original cost function.

Asymptotic convergence rates are, unfortunately, not available for Riemannian conjugate gradient schemes. For steepest descent, linear convergence can be shown, with a constant depending on the eigenvalues of the Riemannian Hessian at the critical point, see Theorem 2.27.

A very different approach which avoids the need for regularization has been discussed in a recent work by Schneider and Uschmajew [SU15], extending Riemannian optimization techniques to the algebraic variety $\mathcal{M}_{\leq \mathbf{r}} := \{X \in \mathbb{R}^{n\times n} |\, \operatorname{rank}(X) \leq \mathbf{r}\}$, the closure of $\mathcal{M}_\mathbf{r}$. Convergence results for the therein proposed gradient scheme follow from Łojasiewicz inequality. Extension of this work to the tensor case remains an open question.

Rauhut, Schneider and Stojanac [RSS13, RSS15] discuss ways to extend hard thresholding techniques from matrix completion to the tensor case and obtain a local convergence result under the assumption of a tensor *restricted isometry property* (RIP). Unfortunately, this RIP is not fulfilled for the entry sampling operator of tensor completion, but only for gaussian measurements.

5.2. Tucker case

In the following sections, we will provide algorithmic details on the individual steps of Algorithm 5.1 in the Tucker case and discuss their computational complexity. The resulting algorithm is called *GeomCG* and was published in [KSV14]. To simplify the expressions for the computational complexity, we assume that $n := n_1 = \ldots = n_d$ and $r := r_1 = \ldots = r_d$.

5.2.1. Evaluation of the cost function and the gradient

Cost function. The calculation of the cost function and its Riemannian gradient (5.3) requires the explicit computation of $|\Omega|$ individual entries of a tensor $\mathbf{X}$ from its Tucker decomposition:

$$\mathbf{X}(i_1, i_2, \ldots, i_d) = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \cdots \sum_{j_d=1}^{r_d} \mathbf{S}(j_1, j_2, \ldots, j_d) U_1(i_1, j_1) U_2(i_2, j_2) \cdots U_d(i_d, j_d).$$

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

Gradient. The projection of $\mathbf{G} := \operatorname{Grad} f(\mathbf{X}_k) = \mathrm{P}_\Omega \mathbf{X}_k - \mathrm{P}_\Omega \mathbf{A}$ onto the tangent space gives the Riemannian gradient $\xi_k := \operatorname{grad} f(\mathbf{X}_k)$, which will be stored in factorized form as

$$\xi_k = \delta\mathbf{S} \bigtimes_{\mu=1}^{d} U_\mu + \sum_{\mu=1}^{d} \mathbf{S} \times_\mu \delta U_\mu \bigtimes_{\substack{\nu=1 \\ \nu\neq\mu}}^{d} U_\nu \,,$$

where

$$\delta\mathbf{S} := \mathbf{G} \bigtimes_{\mu=1}^{d} U_\mu^\mathsf{T}, \qquad \delta U_\mu := (I_{n_\mu} - U_\mu U_\mu^\mathsf{T}) \left[\mathbf{G} \bigtimes_{\nu\neq\mu} U_\nu^\mathsf{T} \right]_{(\mu)} \mathbf{S}_{(\mu)}^\dagger,$$

see Proposition 3.8. By exploiting the sparsity of $\mathbf{G}$, the computation of $\delta\mathbf{S}$ and $\mathbf{U}_\mu$, $\mu = 1, \ldots, d$, requires $O\Big(dr^d(|\Omega| + n) + r^{d+1}\Big)$ operations. This makes the calculation of the gradient the most time-consuming part of our optimization scheme.

5.2.2. Calculation of the new iterate

Search direction. To calculate the new search direction, we use the Polak-Ribière+ update formula (2.9), where the iterates are now tensors.

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

The calculation of β_k requires the evaluation of the vector transport described in Section 3.7 together with inner products of tangent tensors, Section 3.5.3.

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

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

where $\eta_{k-1} \in T_{\mathbf{X}_{k-1}}\mathcal{M}_\mathbf{r}$ is the previous conjugate direction. The vector transport is performed exactly in the same way as above. Due to linearity, the addition of two tensors in the same tangent space is performed by simply adding their first-order variations. Thus, the computation of the new search direction η_k can be performed in $O(dnr^d + r^{d+1})$ operations.

Line search. After computing the new search direction, the linearized line-search is computed according to (5.4). Note that due to the projection operator P_Ω, all involved quantities are sparse. Thus, each of the inner products requires only $O(|\Omega|)$ operations. The costly part is due to the term $\mathrm{P}_\Omega\, \eta_k$, which requires the computation of $|\Omega|$ entries of η_k for a total of $O(|\Omega|(d+1)r^d)$ operations.

Retraction. With the line-search parameter α_k at hand, the next iterate is given by $\mathbf{X}_{k+1} = R(\mathbf{X}_k, \alpha_k \eta_k)$, computed by means of the HOSVD procedure, see Section 3.6, in $O(dr^2 n + r^{d+1})$ operations.

5.2.3. Computational complexity

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

$$O(dr^d(n + |\Omega|)) \tag{5.10}$$

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

5.3. Numerical experiments for the Tucker case

Algorithm 5.1, geomCG, was implemented in MATLAB version 2012a, using the Tensor Toolbox version 2.5 [BK+12] for handling some of the tensor operations. However, to attain reasonable performance, it was important to implement operations with sparse tensors in C and call them via MEX interfaces. In particular, this was done for the evaluation of the objective function, the computation of the Euclidean gradient and its projection onto the tangent space, as well as for the linearized line search, see 5.2.1. For simplicity, we restricted the implementation to the case $d = 3$. The source code is freely available under a BSD license and can be downloaded from `http://anchp.epfl.ch/geomCG`.

To measure the convergence during the iteration, Algorithm 5.1 computes the relative residuals ε_Ω and ε_{Ω_C}, see (5.5) and (5.6). However, to investigate the reconstruction quality of the algorithm, measuring the relative residual on the *sampling set* Ω is not sufficient. For this purpose, we also measure the relative error $\| \mathrm{P}_\Gamma \mathbf{X} - \mathrm{P}_\Gamma \mathbf{A} \| / \| \mathrm{P}_\Gamma \mathbf{A} \|$ on a random *test set* Γ of the same cardinality as Ω. In contrast to Ω and Ω_C, Γ is only used for the evaluation of the algorithm and is not available to the algorithm as additional information for the completion problem.

The initial guess is always taken a random rank-$\mathbf{r}$ Tucker tensor, that is, with all entries of $\mathbf{S}$ and $\mathbf{U}_\mu$ sampled from a uniform random distribution on $[0, 1]$ followed by an orthogonalization of the factor matrices $\mathbf{U}_\mu$.

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

5.3.1. Scaling of the algorithm

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

As a first test, we check that the implementation of Algorithm 5.1 exhibits the same scaling behaviour per iteration as predicted by (5.10). To measure the scaling with regard to the tensor size n, we fix the multilinear rank to $\mathbf{r} = (10, 10, 10)$ and scale the size of the sampling set linearly with the tensor size, $|\Omega| = 10n$. We perform 10 iterations of our algorithm and repeat the process 10 times for different randomly chosen datasets. Analogously, we measure the dependence on the tensor rank by setting the tensor size to $n = 300$ and fixing the sampling set to 0.1% of the full tensor.

The results are shown in Figure 5.1. We observe that our algorithm scales indeed linearly in the tensor size over a large interval $n \in [100, 3000]$. Even for such large tensors, the time per iteration step is very low. Plotting the results for the scaling with regard to the tensor rank, we observe an $O(r^3)$-dependence, in agreement with (5.10).

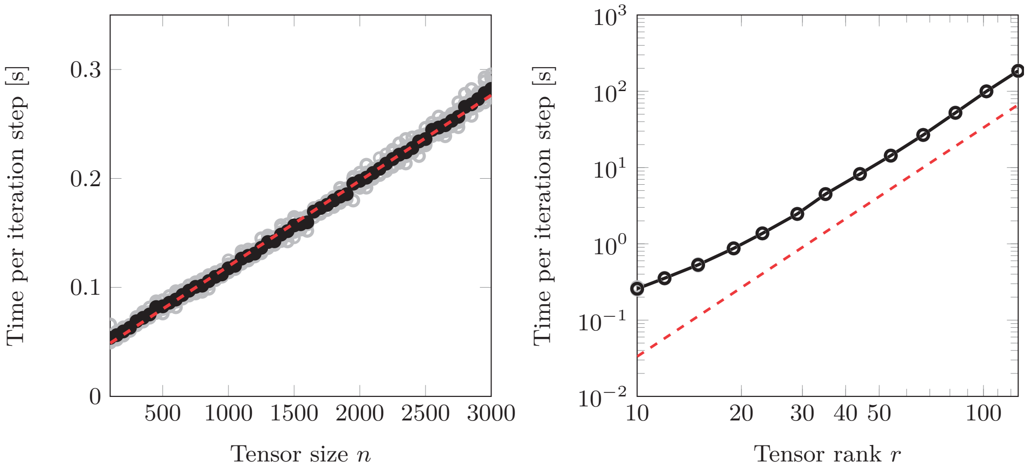


Figure 5.1: Time needed per iteration step for various problem sizes. ***Left:*** *Runtime with fixed rank* $\mathbf{r} = (10, 10, 10)$ *and varying tensor size* $n = n_1 = n_2 = n_3 \in \{100, 150, \ldots, 3000\}$. *The size of the sampling set scales linearly with* n, $|\Omega| = 10n$. ***Right:*** *Runtime with fixed tensor size* $n = (300, 300, 300)$ *and varying tensor rank* $r = r_1 = r_2 = r_3 \in \{20, 25, \ldots, 100\}$. *Size of sampling set: 0.1% of the full tensor. The red dashed line shows a scaling behaviour* $O(r^3)$.

5.3.2. Reconstruction of synthetic data sets

We compare the reconstruction performance of our algorithm with the so-called *hard completion* algorithm by Signoretto et al. [STL$^+$14, Alg. 3], based on the so called *inexact splitting method* applied to the convex non-smooth optimization problem

$$\min_{\mathbf{X}} \sum_{\mu=1}^{3} \lambda_\mu \|\mathbf{X}_{(\mu)}\|_* \quad \text{subject to } \mathrm{P}_\Omega \mathbf{X} = \mathrm{P}_\Omega \mathbf{A},$$

where $\|A\|_*$ denotes the *nuclear norm* of a matrix A, the sum of its singular values. The algorithm depends on certain parameters discussed in [STL$^+$14]. We choose the proximity parameter $\tau = 10$ and the nuclear norm weights $\lambda_1 = \lambda_2 = \lambda_3 = 1$, which corresponds to the settings used in the supplied test routine.

In Figure 5.2 we present the convergence behaviour of the algorithms for varying sizes of the sampling set, in terms of the error on the test set Γ. The sampling set sizes are denoted by a percentage p of the full tensor, $|\Omega| = pN^3$. We use a relative residual of 10^{-12} and a maximum number of 300 iterations as stopping criterions. Both algorithms need more iterations when the number of missing entries increases, but the effect is more strongly pronounced for hard completion. Our algorithm performs better both when measuring the performance with respect to time or the number of iterations.

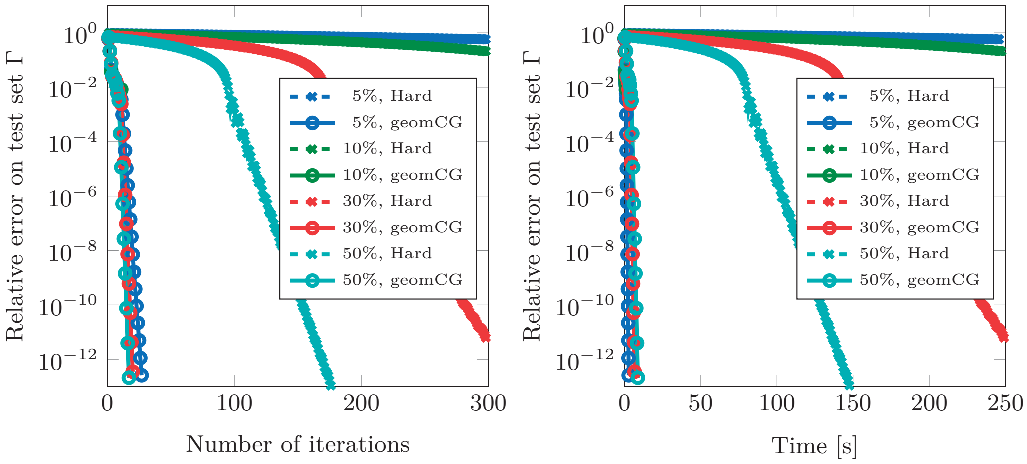


Figure 5.2: Convergence curves for different sampling set sizes as functions of iterations and time for our proposed algorithm (geomCG) and the hard completion algorithm by Signoretto et al. [STL$^+$14]. Tensor size and multilinear rank fixed to $n = 100$ and $\mathbf{r} = (5, 5, 5)$, respectively.

Reconstruction of noisy data. In this part, we investigate the convergence properties of Algorithm 5.1 in the presence of noise. The known entries of $\mathbf{A}$ are perturbed by rescaled Gaussian noise $\mathbf{E}$, such that $\|\mathrm{P}_\Omega \mathbf{E}\| = \epsilon_0 \|\mathrm{P}_\Omega \mathbf{A}\|$ for a prescribed noise level ϵ_0. Ideally, Algorithm 5.1 should return an approximation $\mathbf{X}^*$ at the level of ϵ_0, that is,

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

To test that the noise does not lead to a misidentification of the rank of the underlying problem, we compare the case where we take the initial guess on the correct manifold to an uninformed rank-$(1, 1, 1)$ guess. There, we employ a heuristic rank adaptation strategy discussed in Section 5.3.3. We show in Figure 5.3 that in both cases we can indeed recover the original data up to the given noise level.

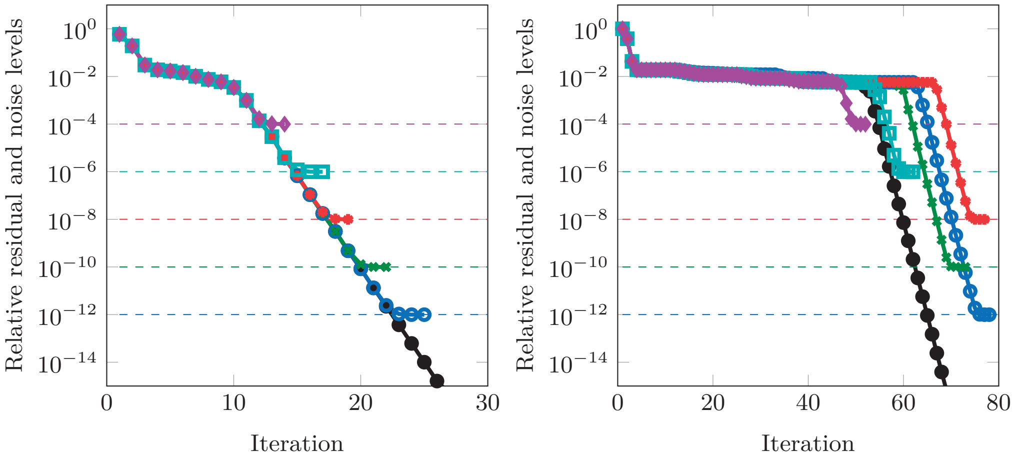


Figure 5.3: Tensor completion from noisy measurements with $n = 100$, $\mathbf{r} = (6, 6, 6)$*. The relative size of the sampling set was fixed to* 10%*. The black line corresponds to the noise-free case. The different colors correspond to the noise levels* $\epsilon_0 \in \{10^{-4}, 10^{-6}, \ldots, 10^{-12}\}$*.* ***Left:*** *Results when the underlying rank* $\mathbf{r}$ *is known.* ***Right:*** *Results for the case of unknown rank of the underlying problem. Due to the rank adaptation procedure, more iterations are necessary.*

Size of the sampling set. It is well known that in the matrix case, the number of random samples needed to exactly recover the original matrix is at least $O(nr \log n)$ under standard assumptions; see e.g. [CT09, KMO10]. Up to this date, these results could not be extended to the tensor case, see e.g. [RSS15] for a discussion. A straightforward solution of the tensor completion problem by applying matrix completion to one matricization, e.g. $\mathbf{X}_{(1)} \in \mathbb{R}^{n \times n^2}$, therefore needs at least $O(n^2 r \log n)$ entries. This approach neglects the existing structure between the modes. Hence one can expect an algorithm that is specially designed for the tensor case to perform better than this bound by exploiting its inherent structure. A slightly improved result was obtained in [MHWG14], where it was proven that $O(r^{\lfloor \frac{d}{2} \rfloor} n^{\lceil \frac{d}{2} \rceil})$ samples suffice.

In the left plot of Figure 5.4, we present numerical experiments suggesting that a similar scaling as in the matrix case may also hold for the three-dimensional tensor case. The algorithm is declared converged (and hence yields perfect reconstruction) if the relative residual drops below 10^{-6} within 100 iterations.

The right plot of Figure 5.4 displays a phase transition of the measured convergence speed of Algorithm 5.1, computed from

$$\rho = \left(\frac{\| \mathrm{P}_\Gamma \mathbf{X}_{k_\text{end}} - \mathrm{P}_\Gamma \mathbf{A} \|}{\| \mathrm{P}_\Gamma \mathbf{X}_{k_\text{end}-10} - \mathrm{P}_\Gamma \mathbf{A} \|} \right)^{\frac{1}{10}} \in [0, 1], \tag{5.11}$$

where k_end is the final iteration.

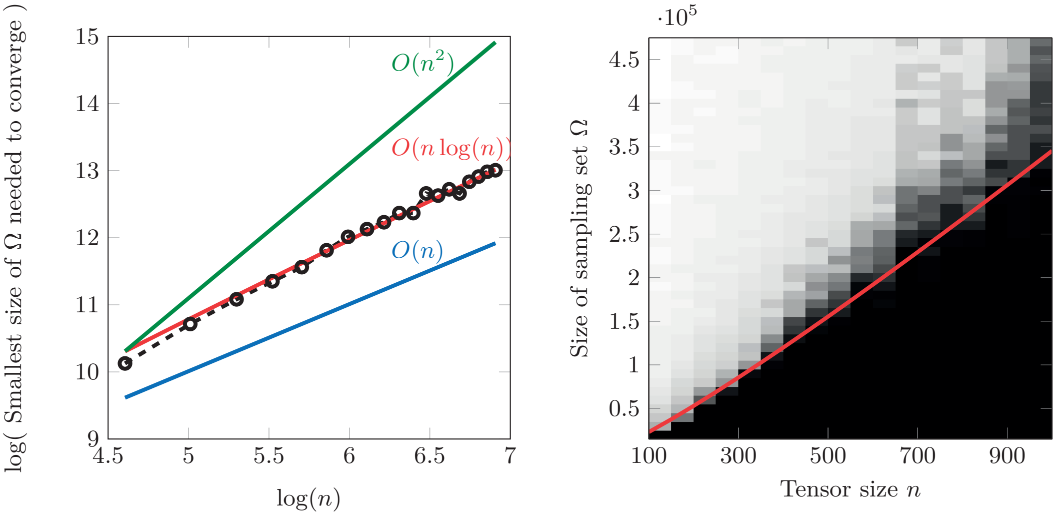


Figure 5.4: Scaling of the sampling set size needed to reconstruct the original tensor of fixed multilinear rank $(10, 10, 10)$. ***Left:*** *Minimum size of sampling set needed to attain convergence vs. tensor size* $n = n_1 = n_2 = n_3$. ***Right:*** *Phase transition of the convergence speed* (5.11). *White means fast convergence, black means no convergence. The red line corresponds to* $O(n\log(n))$.

5.3.3. Applications

In contrast to synthetic data sets, tensors from applications usually do not posses a clear, well-defined multilinear rank. Often, they exhibit a rather smooth decay of the singular values in each matricization. In such a setting, Algorithm 5.1 requires a good initial guess, as directly applying it with a (large) fixed rank $\mathbf{r}$ usually results in severe overfitting. We propose the following heuristic to address this problem: Starting from a multilinear rank-$(1, 1, 1)$-approximation, we iteratively increase the multilinear rank in each mode and rerun our algorithm with the previous result as initial guess. This procedure is repeated until the prescribed final multilinear rank $\mathbf{r}$ is reached. We increase the multilinear rank every time the current relative change in the square root of the cost function is smaller than a tolerance δ:

$$\left|\sqrt{f(\mathbf{X}_{k-1})} - \sqrt{f(\mathbf{X}_k)}\right| < \delta\sqrt{f(\mathbf{X}_k)}. \tag{5.12}$$

Initially, we use a large value for δ, say $\delta = 1$. We observed this approach to be effective at steering the algorithm into the direction of the optimal solution. Once we arrive at the final rank $\mathbf{r}$, we can also use (5.12) as a stopping criterion with a much smaller value for δ, say $\delta = 0.001$. In case of convergence problems, the initial value for δ should be chosen smaller, at the cost of additional iterations. In the following numerical experiments, we always include this initialization procedure in the reported computation times. For a more detailed look at different rank increasing strategies, we refer to Section 5.3.4, where the process employed in the following applications is denoted by *random per-mode cyclic*.

Hyperspectral image. As a first application of our algorithm to real-world data, we consider the hyperspectral image "Ribeira" [FNA04] discussed in [SPM$^+$11]. This results in a tensor of size $1017 \times 1340 \times 33$, where each slice corresponds to an image of the same scene measured at a different wavelength. To provide a faithful comparison, we proceed in the same way as in [SPM$^+$11] and resize the tensor to $203 \times 268 \times 33$ by bilinear interpolation before applying our algorithm. The results are shown in Table 5.1. The reconstruction quality is assessed in terms of the *normalized root mean squared error*:

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

where Ω^c is the complement of the sampling set, that is, the unknown entries. We compare with the results reported in [SPM$^+$11] for the tensor completion algorithm *tensor*, the frame-wise matrix completion approach *frame*, and matrix completion applied to the *mode-3* matricization only. This approach uses the fact that the variation between the frames in the spectral mode is very low. As shown in Figure 5.5, the singular values of the matricizations decay at a different rate. We take this into account in our algorithm, by choosing the final mode-1 and mode-2 ranks of the approximation significantly larger than the mode-3 rank. It can be observed that our algorithm (*geomCG*) yields very competitive results, especially in the case where the sampling set is small. There is one case of overfitting for geomCG$(55, 55, 5)$, marked by a star.

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

$$f : [-1, 1]^3 \to \mathbb{R}, \quad x \mapsto \mathrm{e}^{-\|x\|_2} \tag{5.13}$$

discretized on a uniform tensor grid with mesh width $h = 1/100$. The function values are collected in a tensor $\mathbf{A} \in \mathbb{R}^{201 \times 201 \times 201}$. In this setting, we assume that the location of the singularity is known a priori. As f has a cusp at the origin, the information in $\mathbf{A}$ is strongly localized at this point and tensor completion applied naively to $\mathbf{A}$ would not lead to reasonable compression. To avoid this effect, we therefore cut out a small hypercube $[-0.1, 0.1]^3$, corresponding to the $21 \times 21 \times 21$ central part of the discretized tensor. The idea is to not include this region in the sampling set Ω. The entries corresponding to this region are stored separately and reconstructed exactly after performing low-rank tensor completion on the remaining region. We therefore do also not include the central part in the test set Γ when verifying the accuracy of the completed tensor. The obtained results are shown in Figure 5.6. Already sampling 5% of the entries gives an accuracy of 10^{-5}. This would yield a compression ratio of 5.1% if we stored the involved entries. However, storing the rank-$(5, 5, 5)$ approximation along with the central part yields the significantly lower compression ratio of 0.15%.

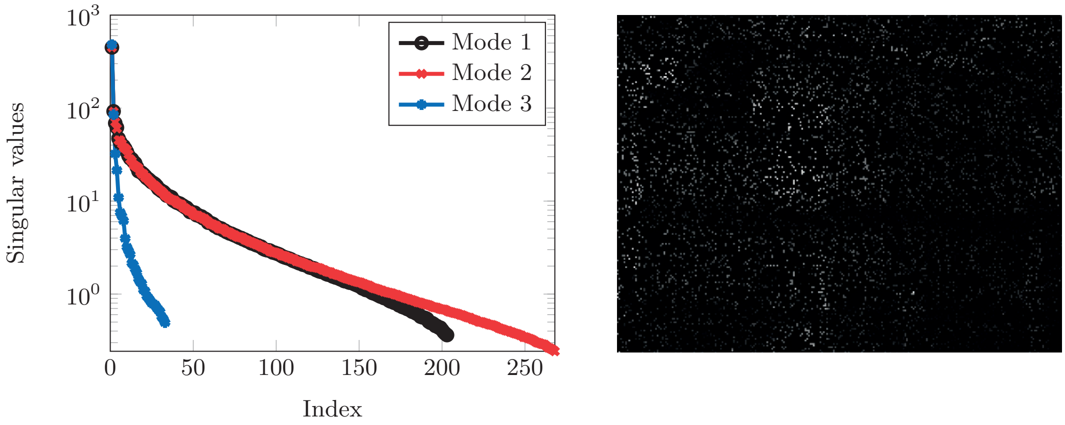
	Full Ribeira data set — sampling percentage					
	10%		30%		50%	
	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]
frame [SPM+11]	0.092	3.78	0.061	3.72	0.046	2.30
mode-3 [SPM+11]	0.068	0.27	0.018	0.31	**0.012**	0.33
tensor [SPM+11]	0.072	26.3	0.031	25.8	0.020	42.0
geomCG$(15, 15, 6)$	0.047	0.06	0.046	0.11	0.046	0.19
geomCG$(65, 65, 7)$	**0.025**	1.67	**0.017**	4.33	0.017	6.86

	First 5 frames of Ribeira data set — sampling percentage					
	10%		30%		50%	
	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]	NRMSE	time [10^3 s]
frame [SPM+11]	0.071	0.15	0.046	0.14	0.034	0.14
mode-3 [SPM+11]	0.191	0.02	0.119	0.02	0.070	0.02
tensor [SPM+11]	0.067	3.14	0.034	4.48	0.023	4.06
geomCG$(15, 15, 5)$	**0.058**	0.01	0.033	0.02	0.032	0.03
geomCG$(55, 55, 5)$	0.075*	0.15	**0.026**	0.36	**0.016**	0.42

Table 5.1: Reconstruction results for "Ribeira" hyperspectral image. The results for frame, mode-3 *and* tensor *are taken from [SPM+11]. geomCG*(r_1, r_2, r_3) *denotes the result of Algorithm 5.1 using a prescribed final multilinear rank* (r_1, r_2, r_3).

Singular value decay of matricizations

10% of hyperspectral tensor, 16th frame



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

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

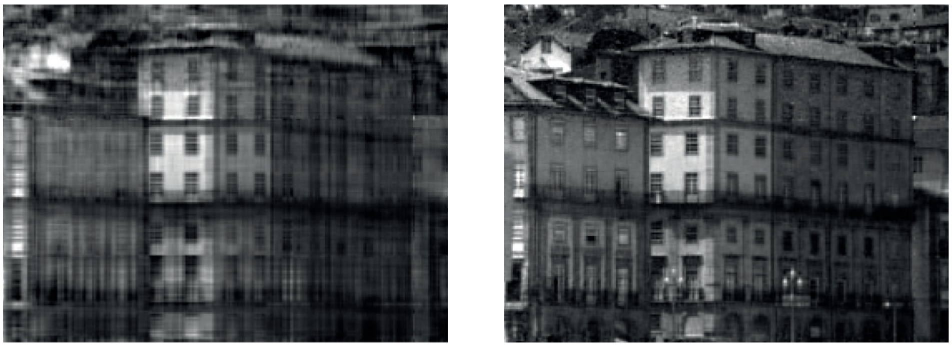


Figure 5.5: ***Top left:*** *Singular value decay of each matricization.* ***Top right:*** *The sampled tensor* $\mathrm{P}_\Omega\,\mathbf{A}$ *with 10% known entries. Unknown entries are marked in black.* ***Bottom left:*** *GeomCG with rank increase up to a final rank of* $\mathbf{r} = (15, 15, 6)$. ***Bottom right:*** *GeomCG with rank increase up to a final rank of* $\mathbf{r} = (65, 65, 7)$.

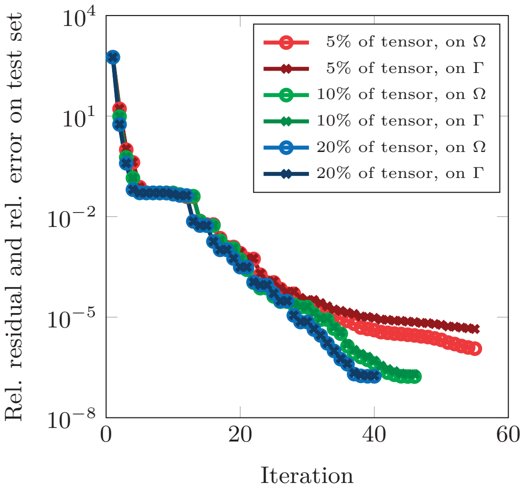


Figure 5.6: Convergence of the algorithm for the discretization of a function with cusp (5.13)*. The final rank of the approximation is* $\mathbf{r} = (10, 10, 10)$*. The part corresponding to* $[-0.1, 0.1]^3$ *is excluded from the sampling set* Ω *and the test set* Γ*.*

Stochastic elliptic PDE with L^2 basis expansion. Finally, we consider an elliptic PDE with stochastic coefficients:

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

where $y \in \Theta$ is a random variable and $D = [-1, 1]$ is the computational domain. We assume that we can expand the stochastic coefficient $a(x, \alpha)$ into

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

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

$$(A_0 + \alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3)x = f, \tag{5.14}$$

where each $A_\mu \in \mathbb{R}^{m\times m}$ is the FE stiffness matrix corresponding to the coefficient a_μ. We refer to, e.g., [SG11] for a detailed discussion of this procedure. In our examples, we choose

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

The parameters α are then sampled uniformly on a tensor grid on $[-1, 1] \times [-1, 1] \times [-1, 1]$ of size $n \times n \times n$. Assuming that we are only interested in the spatial mean of the solution for a specific set of parameters, this results in a solution tensor $\mathbf{X} \in \mathbb{R}^{n\times n\times n}$. Each entry of this tensor corresponds to the spatial mean for a certain combination of the discretized $(\alpha_1, \alpha_2, \alpha_3)$ and requires the solution of a discretized PDE. Hence, evaluating the full tensor is fairly expensive. Using tensor completion,

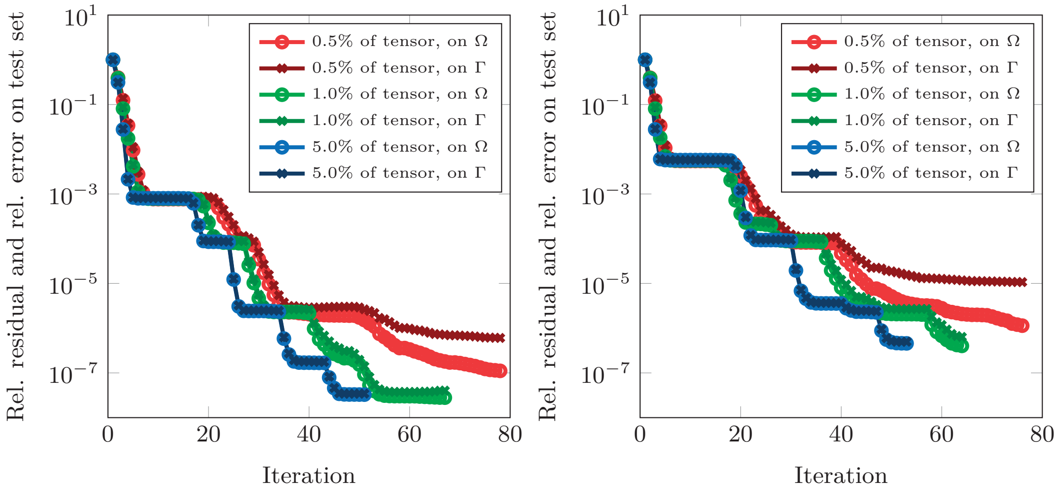


Figure 5.7: Convergence for the solution tensor of size $n = 100$ of a parametrized linear system obtained from a discretized stochastic PDE. ***Left:*** *Result for $\sqrt{\lambda_\mu} = 5\exp(-2\mu)$.* ***Right:*** *Result for $\sqrt{\lambda_\mu} = (1+\mu)^{-2}$. The final rank of the solution is $\mathbf{r} = (4,4,4)$.*

we sample $\mathbf{X}$ at (few) randomly chosen points and try to approximate the missing entries.

In Figure 5.7 we show the results of this approach for $m = 50, n = 100$, and two different choices of λ_μ. We used the basis coefficients $\sqrt{\lambda_\mu} = 5\exp(-2\mu)$ and $\sqrt{\lambda_\mu} = (1+\mu)^{-2}$, respectively. As the second choice results in slower singular value decays, our algorithm requires more iterations to attain the same accuracy. Using 5% of the tensor as a sampling set is in both cases sufficient to recover the original tensor to good precision. As the sampling set gets smaller, overfitting of the sampling data is more likely to occur, especially for the second choice of λ_μ.

5.3.4. Comparison of rank adaptation strategies

As already briefly mentioned in Section 5.3.3, there is an obvious disadvantage in the way the tensor completion problem (5.1) is defined: The rank of the underlying problem has to be known *a priori*. Instead, we can require that the solution should have rank smaller or equal than a certain prescribed maximal rank. This maximal rank $\mathbf{r}_{\max}$ is determined by the computational resources available and not necessarily by the underlying data. Thus, we perform a *rank adaptation* procedure to reach $\mathbf{r}_{\max}$.

First, we run geomCG with $\mathbf{r}_1 = (1,1,\dots,1)$ to obtain a result $\mathbf{X}_1$. Then, we add a rank-1 correction term $\mathbf{R}$ and then run geomCG again with starting guess $\mathbf{X}_1 + \mathbf{R}$ and $\mathbf{r}_2 = \operatorname{rank}(\mathbf{X}_1 + \mathbf{R})$ to obtain $\mathbf{X}_2$. An optimal rank correction would be given by the minimizer of

$$\min_{\substack{\mathbf{R}\in\mathbb{R}^{n_1\times\cdots\times n_d}\\ \operatorname{rank}(\mathbf{R})=(1,\dots,1)}} f(\mathbf{X}_1 + \mathbf{R}), \tag{5.15}$$

which is not computationally feasible to compute exactly. Instead, we can choose $\mathbf{R}$ to be a rank-1 approximation of the negative Euclidean gradient $-\operatorname{Grad} f(\mathbf{X}_1)$, so that we perform one step of steepest descent to obtain $\mathbf{X}_2$, see [UV15] for a discussion of such greedy rank-1 updates in the context of matrix completion. A much simpler choice is a random rank-1 tensor $\mathbf{R}$ of small norm as a correction. Furthermore, we can either increase all ranks simultaneously by 1 or cycle through the individual modes $\mu = 1, \dots, d$.

In the following, we compare the performance of different rank increasing strategies for the three-dimensional case $d = 3$.

- **Random increase:** To all basis matrices U_μ, we add a random correction vector $R_\mu \in \mathbb{R}^{n_\mu}$, with i.i.d. normally distributed entries, $R_\mu(i_\mu) \sim \mathcal{N}(0,1)$ for $i_\mu = 1, \dots, n_\mu$.
  $$U_\mu \leftarrow \Big[U_\mu, R_\mu\Big] \in \mathbb{R}^{n_\mu \times r_\mu + 1},$$
  where we also orthogonalize the last column R_μ of U_μ such that $U_\mu^\mathsf{T} U_\mu = I_\mu$ still holds. The core tensor is extended to size $(r+1)^3$ by adding slices with entries of magnitude 10^{-15} in each mode.

- **Random per-mode cyclic:** Analogous to *random increase*, but only one mode is enlarged at once, with a cyclic change of the modes:
  $$(r_1, r_2, r_3) \to (r_1 + 1, r_2, r_3) \to (r_1 + 1, r_2 + 1, r_3) \to (r_1 + 1, r_2 + 1, r_3 + 1) \dots$$

- **Pursuit:** Following the *Riemannian pursuit* scheme of [TTW$^+$14], we choose $\mathbf{R}$ to be the rank-1 approximation of the residual, i.e. the Euclidean gradient $\operatorname{Grad} f(\mathbf{X}_i)$. As $\operatorname{Grad} f(\mathbf{X}_i)$ is a sparse tensor, we obtain its rank-1 approximation $\mathbf{R}$ using a sparse SVD. The enrichment $\mathbf{R}$ is then added to the current iterate with a certain step length α,
  $$\mathbf{X}_{i+1} = \mathbf{X}_i + \alpha \mathbf{R},$$
  such that the cost function is minimized. Here, α is chosen using the formula for the linearized line search (5.4), which, for this case, is an exact line search.

- **Subspace:** Here, we increase the basis matrices as in the *random increase* strategy, but the updated core tensor $\mathbf{S} \in \mathbb{R}^{r_1+1 \times \dots \times r_d+1}$ is chosen *optimal* in the sense
  $$\mathbf{S} = \operatorname*{argmin}_{\mathbf{C} \in \mathbb{R}^{r_1+1 \times \dots \times r_3+1}} \| \mathrm{P}_\Omega(\mathbf{C} \times_1 U_1 \times_2 U_2 \times_3 U_3) - \mathrm{P}_\Omega \mathcal{A} \|^2.$$
  This amounts to solving the optimal rank correction problem (5.15) on a much smaller subspace. It is a linear least squares problem with a system matrix of size $|\Omega| \times (r+1)^d$, which can be solved exactly using $O(|\Omega| r^{2d})$ operations. Hence, this approach is limited to very low-dimensional problems.

To compare the approaches, we run a fixed number of iterations of geomCG for each rank; 30 iterations when the rank is increased in all three modes simultaneously and 10 iterations in *random per-mode cyclic*. Figure 5.8 shows the resulting convergence for the Karhunen-Loève experiment of Section 5.3.3 with $\sqrt{\lambda_\mu} = 5\exp(-2\mu)$. For all strategies we plot the relative error on the sampling set Ω (dashed line) and on

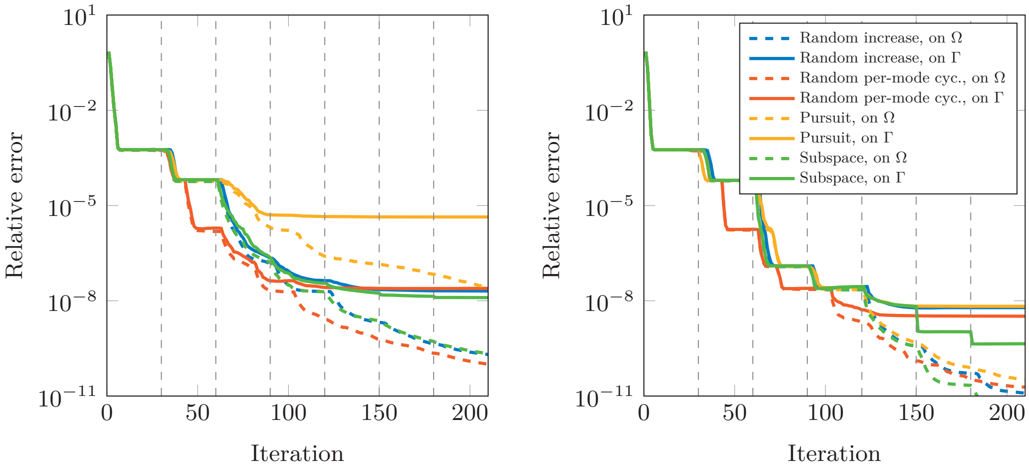


Figure 5.8: Convergence for the solution tensor of size $n = 100$ of a discretized stochastic PDE when using different rank adaptation strategies. Dashed lines show relative residual on Ω whereas solid lines refer to the relative error on Γ. ***Left:*** *1% sampling.* ***Right:*** *5% sampling.*

the testing set Γ (solid line). The vertical dashed lines correspond to the number of iterations after which a full rank update $(r_1, r_2, r_3) \to (r_1+1, r_2+1, r_3+1)$ has taken place: 30 iterations for the global rank increase strategies and $3 \cdot 10 = 30$ iterations for *random per-mode cyclic*. We observe that using the residual instead of a random enlargement performs similar (5%) or even worse for low sampling (1%). In all cases, *subspace* and *random per-mode cyclic* perform best. For a very small sampling set, where only 1% of the entries are given, none of the rank adaptation strategies is able to achieve a relative error on the test set Γ smaller than 10^{-8}.

As a second example, we assess the performance of the different strategies on a tensor obtained by discretizing the function

$$f(x, y, z) = \frac{1}{1 + x^2 + y^2 + z^2}$$

on the cube $[-1, 1]^3$ using a uniform tensor grid with 101 grid points in all directions. The results for the reconstruction of the resulting tensor $\mathbf{A} \in \mathbb{R}^{101\times 101\times 101}$ are shown in Figure 5.9. For 5% sampling, the problem seems to be very easy and all strategies perform equally well. In contrast to this, for 1% sampling, all strategies fail except *subspace*.

Due to the good performance of both the per-mode cyclic and the subspace strategy, we are led to consider yet another approach,

- **Subspace per-mode cyclic:** Analogous to *subspace*, but only one mode is enlarged at once in a cyclic way as in *random per-mode cyclic*.

The result is shown in Figure 5.10 for 1% sampling, the same setup as before in Figure 5.9. We see that the cyclic approach improves quite a bit over *subspace*. It is remarkable that even in this apparently very difficult case (remember that all other approaches completely failed), we are able to complete the tensor up to a relative error on Γ of 10^{-6}.

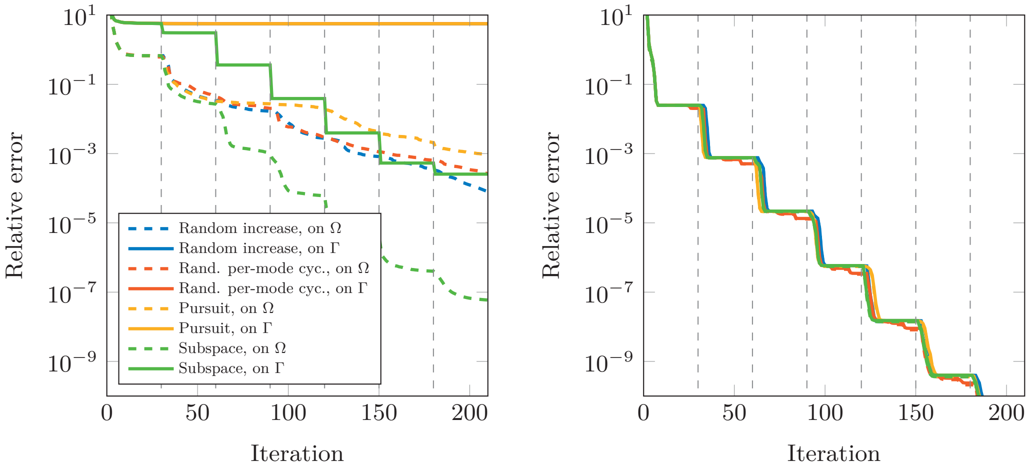


Figure 5.9: Convergence for the reconstruction of a function discretized on a uniform grid of size $101 \times 101 \times 101$ *when using different rank adaptation strategies. Dashed lines show the relative residual on* Ω *whereas solid lines refer to the relative error on* Γ. ***Left:*** *1% sampling. The solid red and blue lines are covered by the solid yellow line.* ***Right:*** *5% sampling.*

We can conclude that the subspace optimization approaches allow for much better reconstruction results at very low sampling rates, but require a very expensive solution of a least-squares problem at each rank increasing step. Hence, they do not scale well with the number of dimensions d and are limited to smaller problems. For higher sampling rates, a random rank increasing strategy works just as well, for almost no extra cost. In contrast to the matrix case reported in [TTW^{+}14], the introduction of gradient information does not improve the reconstruction quality and can even yield worse results than a random choice.

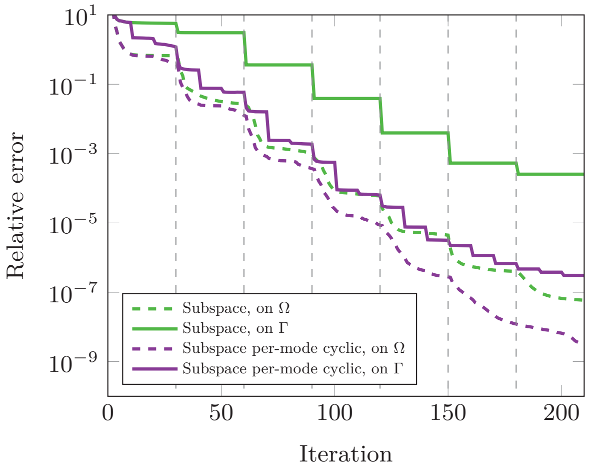


Figure 5.10: Comparison of the two rank increasing strategies subspace *and* subspace per-mode cyclic *for the discretized function tensor with 1% sampling.*

5.4. Tensor train case

If higher dimensional problems are considered, the tensor train format as introduced in Chapter 4 is preferred.

5.4.1. Evaluation of the cost function and the gradient

Cost function. In the tensor completion problem (5.1), the cost function is given by $f(\mathbf{X}) = \|\mathrm{P}_\Omega \mathbf{X} - \mathrm{P}_\Omega \mathbf{A}\|^2/2$. As the data samples $\mathrm{P}_\Omega\,\mathbf{A}$ are supplied as input, evaluating the cost function reduces to evaluating the entries of the sparse tensor $\mathrm{P}_\Omega\,\mathbf{X}$. We compute the application of the sampling operator P_Ω to a TT tensor $\mathbf{X}$ by direct evaluation of the matrix product for each sampling point:

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

Thus, for each sampling point, we need to compute $d-1$ matrix-vector multiplications. Hence, evaluating the tensor at $|\Omega|$ sampling points involves $O(|\Omega|(d-1)r^2)$ operations.

Gradient. According to Section 5.3, the Riemannian gradient is obtained by projecting the Euclidean gradient onto the tangent space.

As the Euclidean gradient $\mathbf{G} := \operatorname{Grad} f(\mathbf{X}) = \mathrm{P}_\Omega\,\mathbf{X} - \mathrm{P}_\Omega\,\mathbf{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is a very large but sparse tensor, its projection onto the tangent space has to be handled with care. When successively calculating the projection (4.17), the intermediate results are, in general, dense already after the first step. For example, the intermediate result $\mathbf{G}^{<\mu>}\mathbf{X}_{\geq\mu+1}$ is a huge dense matrix of size $\mathbb{R}^{n_1 n_2\cdots n_\mu\times r_\mu}$ whereas the resulting $\delta\widetilde{\mathbf{U}}_\mu^{\mathsf{L}}$ is only of size $\mathbb{R}^{r_\mu n_\mu\times r_{\mu+1}}$. Let us restate the equations of the first order variations $\delta\widetilde{\mathbf{U}}_\mu$:

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

The projection $(\mathbf{I}_{r_\mu n_\mu} - \mathbf{U}_\mu^{\mathsf{L}}(\mathbf{U}_\mu^{\mathsf{L}})^{\mathsf{T}})$ is a straight-forward matrix multiplication and can be done separately at the end. Instead, we focus on $\mathbf{C}_\mu^{\mathsf{L}} := (I_\mu\otimes\mathbf{X}_{\leq\mu-1})^{\mathsf{T}}\mathbf{G}^{<\mu>}\mathbf{X}_{\geq\mu+1}$. Let $\Theta_{i_\mu}\subset\Omega$ be the set of sampling points $(j_1,\dots,j_d)$ where the μth index j_μ coincides with i_μ:

$$\Theta_{i_\mu} = \{(j_1,\dots j_d)\in\Omega \mid j_\mu = i_\mu\}\subset\Omega. \tag{5.16}$$

The i_μth slice of $\mathbf{C}_\mu$ is given by a sum over Θ_{i_μ}, as the other entries do not contribute:

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

which can be reformulated as

$$C(i_\mu) = \sum_{(j_1,\dots j_d)\in\Theta_{i_\mu}} \mathbf{G}(j_1,\dots,j_d)\big[U_1(j_1)\cdots U_{\mu-1}(j_{\mu-1})\big]^{\mathsf{T}}\big[V_{\mu+1}(j_{\mu+1})\cdots V_d(j_d)\big].$$

Algorithm 5.2 Computing the Riemannian gradient

```
Input: Left-orth. cores $\mathbf{U}_\mu$ and right-orth. cores $\mathbf{V}_\mu$ of $\mathbf{X} \in \mathcal{M}_\mathbf{r}$, Euclid. gradient $\mathbf{Z}$
Output: First order variations $\delta\widetilde{\mathbf{U}}_\mu$ for $\mu = 1, \ldots, d$.
  {% Initialize cores}
  for $\mu = 1, \ldots, d$ do
    $\mathbf{C}_\mu \leftarrow \mathrm{zeros}(r_\mu, n_\mu, r_{\mu+1})$
  end for

  for $(j_1, j_2, \ldots, j_d) \in \Omega$ do

    {% Precompute left matrix product}
    $\mathbf{U}_L\{1\} \leftarrow \mathbf{U}_1(j_1)$
    for $\mu = 2, \ldots, d-1$ do
      $\mathbf{U}_L\{\mu\} \leftarrow \mathbf{U}_L\{\mu-1\}\mathbf{U}_2(j_2)$
    end for

    $\mathbf{V}_R \leftarrow \mathbf{V}_d(j_d)$

    {% Calculate the cores beginning from the right}
    $\mathbf{C}_d(j_d) \leftarrow \mathbf{C}_d(j_d) + \mathbf{G}(j_1, j_2, \ldots, j_d) \cdot \mathbf{U}_L\{d-1\}^\mathsf{T}$
    for $\mu = d-1, \ldots, 2$ do
      $\mathbf{C}_\mu(j_\mu) \leftarrow \mathbf{C}_\mu(j_\mu) + \mathbf{G}(j_1, j_2, \ldots, j_d) \cdot \mathbf{U}_L\{\mu-1\}^\mathsf{T}\mathbf{V}_R^\mathsf{T}$
      $\mathbf{V}_R \leftarrow \mathbf{V}_\mu(j_\mu)\mathbf{V}_R^\mathsf{T}$
    end for
    $\mathbf{C}_1(j_1) \leftarrow \mathbf{C}_1(j_1) + \mathbf{G}(j_1, j_2, \ldots, j_d) \cdot \mathbf{V}_R^\mathsf{T}$
  end for
  $\delta\widetilde{\mathbf{U}}_d \leftarrow \mathbf{C}_d$
  {% Project cores $1, \ldots, d-1$ onto orth. complement of the range of $\mathbf{U}_\mu^\mathsf{L}$}
  for $\mu = 1, \ldots, d-1$ do
    $\delta\widetilde{\mathbf{U}}_\mu^\mathsf{L} \leftarrow \mathbf{C}_\mu^\mathsf{L} - \mathbf{U}_\mu^\mathsf{L}(\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}\mathbf{C}_\mu^\mathsf{L}$
  end for
```

The algorithmic implementation for all first order variations $\delta\widetilde{\mathbf{U}}_\mu$ for $\mu = 1, \ldots, d$ is shown in Algorithm 5.2, with reuse of intermediate matrix products. For each entry, we have to perform $d-1$ small matrix-vector products for a total cost of $O(d|\Omega|r^2)$. The projection onto the orthogonal complement at the end adds another $O(dnr^3)$ operations.

5.4.2. Calculation of the new iterate

Search direction. Many options exist for the search direction update within the framework of nonlinear conjugate gradient. Here, we choose the Fletcher-Reeves update (2.8),

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

as it needs only about $2dnr^2$ operations when computing the norm as the square root of the inner product, see Section 4.4.4. In our experiments, we observed that choosing a different scheme, such as Polak-Ribière+, (2.9), has almost no influence on the convergence of Algorithm 5.1 (RTTC).

Line search. The line search is computed using the linearized line search (5.4). Due to the term $\mathrm{P}_\Omega\, \eta_k$, this requires the sampling of $|\Omega|$ entries of the search direction $\eta_k \in T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}$. Using the result of Section 4.4.3, we write η_k as a TT tensor of rank $2\mathbf{r}$ and reuse the sampling routine for TT tensors with an acceptable amount of extra cost — the rank is doubled, hence the number of operations is quadrupled. In total, the approximate line search can be calculated in $O(|\Omega|(d-1)r^2)$ operations.

Retraction. The new iterate $\mathbf{X}_{k+1} = R(\mathbf{X}_k, \alpha_k\eta_k)$ is computed using the TT-SVD procedure described in Section 4.5 for a total cost of approximately $O(dnr^3)$ operations.

5.4.3. *Computational complexity*

If we sum up the computational complexity needed for one iteration of RTTC, we arrive at a total cost of

$$O(dnr^3 + d|\Omega|r^2) \tag{5.17}$$

operations. Note that for reconstruction, the number of samples $|\Omega|$ should be $O(dnr^2)$, as it needs to be at least as large as the dimension of the manifold (4.8). Therefore, in realistic cases, the algorithm will have a complexity of $O(d^2nr^4)$.

5.4.4. *Adaptive rank adjustment*

As discussed in 5.3.4, a rank adaptation scheme is necessary, as the rank tuple $\mathbf{r} = (r_0, r_1, \ldots, r_d)$ of the underlying data might be unknown. This is even more severe for the high-dimensional TT setting, where the ranks r_μ can be very different from each other, see e.g. [Ose11c, Table 3.2] for a common distribution of the ranks. Thus, we define a maximal rank $\mathbf{r}_{\max}$ determined by the computational resources available and not necessarily by the underlying data. As the rank has a strong influence on the complexity of the algorithm, see Section 5.4.3, this motivates the following rank-adaptive procedure analogous to the Tucker case. First, we run RTTC with $\mathbf{r}_1 = (1, 1, \ldots, 1)$ to obtain a result $\mathbf{X}_1$. Then, we add a rank-1 correction term $\mathbf{R}$ and then run RTTC again with starting guess $\mathbf{X}_1 + \mathbf{R}$ and $\mathbf{r}_2 = \operatorname{rank}(\mathbf{X}_1 + \mathbf{R})$ to obtain $\mathbf{X}_2$. We were not able to extend the successful subspace optimization scheme as described in Section 5.3.4 to the TT case in a convincing way and the high-dimensionality makes a global update very costly. An ALS-like single core optimization scheme is computationally feasible, but did not improve on the reconstruction quality.

Instead, we perform a simple local update to only increase one rank-index at a time, cycling through each r_μ, $\mu = 1, \ldots, d-1$. To modify the rank r_μ, we set

Algorithm 5.3 Adaptive rank adjustment

```
Input: Sampled data $P_\Omega \mathbf{A}$, $P_{\Omega_C} \mathbf{A}$, maximal rank $\mathbf{r}_{\max}$, acceptance parameter $\rho$
Output: Completed tensor $\mathbf{X}$ with $\operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) \leq \mathbf{r}_{\max}$
  $\mathbf{X}$ random tensor, $\mathbf{r} := \operatorname{rank}_{\mathrm{TT}}(\mathbf{X}) = (1, 1, \ldots, 1)^{\mathsf{T}}$
  $\mathbf{X} \leftarrow$ Result of RTTC using $\mathbf{X}$ as starting guess.
  for $k = 2, \ldots, r_{\max}$ do
    for $\mu = 1, \ldots, d-1$ do
      if $r_\mu \geq r_{\max,\mu}$ then
        $\mu$th rank is maximal. Skip.
      else
        $\mathbf{X}_{\text{new}} \leftarrow$ Increase $\mu$th rank of $\mathbf{X}$ to $(1, r_1, \ldots, r_\mu + 1, \ldots r_{d-1}, 1)$ using (5.18)
        $\mathbf{X}_{\text{new}} \leftarrow$ Result of Algorithm 5.1 using $\mathbf{X}_{\text{new}}$ as starting guess.

        if $(\varepsilon_{\Omega_C}(\mathbf{X}_{\text{new}}) - \varepsilon_{\Omega_C}(\mathbf{X})) > \rho\, \varepsilon_{\Omega_C}(\mathbf{X})$ then
          Revert step.
        else
          $\mathbf{X} \leftarrow \mathbf{X}_{\text{new}}$        % accept step.
        end if
      end if
    end for
  end for
```

$$\mathbf{U}_\mu^{\mathsf{L}} \leftarrow \begin{bmatrix} \mathbf{U}_\mu^{\mathsf{L}} & R_\mu \end{bmatrix}, \qquad \mathbf{U}_{\mu+1}^{\mathsf{R}} \leftarrow \begin{bmatrix} \mathbf{U}_{\mu+1}^{\mathsf{R}} \\ R_{\mu+1} \end{bmatrix}. \tag{5.18}$$

When we choose $R_\mu \in \mathbb{R}^{r_{\mu-1}n_\mu \times 1}$ and $R_{\mu+1} \in \mathbb{R}^{1 \times n_{\mu+1}r_{\mu+1}}$ from a local projection of the approximated gradient, we obtain an AMEn-like procedure [DS14]. While very successful in the case of linear systems and eigenvalue problems, see Chapter 7, we have found that choosing R_μ and $R_{\mu+1}$ as random vectors with small magnitude, say, 10^{-8}, performs just as well, similar to the Tucker case in Section 5.3.4. The resulting rank-adaptive algorithm is shown in Algorithm 5.3. This procedure is repeated until either the prescribed residual tolerance is fulfilled or $\mathbf{r} = \mathbf{r}_{\max}$ is reached.

As the main goal of the first steps is to steer the iterates into the right direction, only few steps of RTTC are required at each level. We make use of the stagnation criterion (5.7) with a crude tolerance of, say, $\delta = 0.01$. With this choice, RTTC usually needs less than 10-15 iteration steps at each level. At the final rank, a finer tolerance is used to allow for more iterations.

In the worst-case scenario, this leads to $(d-1)r_{\max}$ executions of RTTC. To limit the occurrences of unnecessary rank increases, it can be helpful to revert the rank in the current mode if the increase did not improve on ε_{Ω_C}. To detect such cases, we measure the relative change from $\varepsilon_{\Omega_C}(\mathbf{X})$ to $\varepsilon_{\Omega_C}(\mathbf{X}_{\text{new}})$. If it did improve, we always accept the step; additionally, the parameter $\rho \geq 0$ allows for slight increases of the error in one rank-increasing step, with the hope that it will decrease in later steps. In the numerical experiments, we chose $\rho = 1$.

As a side remark, we mention that a very different approach to introduce rank-adaptivity could try to replace the retraction step by a rounding procedure, which does not truncate to a fixed rank, but adjusts the rank according to some prescribed accuracy. As the step $\mathbf{X}_k + \alpha\eta_k$ has up to rank $2\mathbf{r}$, see Remark 4.10, this would allow to possibly double the rank in each step. We have performed a few tests in this direction, but could not achieve good performance.

5.5. Numerical experiments for the TT case

In the following section we investigate the performance of RTTC, Algorithm 5.1, and compare it to existing methods, namely HTOpt [DSH13, DSH15], ADF [GKK13] and a simple alternating optimization scheme, see Section 5.5.2.

To quantify the reconstruction quality of the algorithms, we measure the relative error on a test set Γ different from Ω and Ω_C. All sampling sets are created by randomly sampling each index i_μ from a uniform distribution on $\{1, \dots, n_\mu\}$. We sample more entries than necessary, such that after removing duplicates we end up with a sampling set of the desired size.

As mentioned in the introduction, cross approximation techniques can be used if the sampling set Ω is not prescribed, but can be chosen freely. This is often the case when dealing with the approximation of high-dimensional functions and solutions of parameter-dependent PDEs. To show the effectiveness of tensor completion in these applications, we compare RTTC to state-of-the-art cross-approximation algorithms based on the TT [DS14] and HT format [BG14].

We have implemented RTTC in Matlab based on the TTeMPS toolbox, see `http://anchp.epfl.ch/TTeMPS`. As shown in Section 5.4, the calculation of the cost function, the gradient and the linearized line search involve operations on sparse tensors. To obtain an efficient algorithm we have implemented these crucial steps in C using the Mex-function capabilities of Matlab with direct calls to BLAS routines wherever possible. For example, when evaluating the cost function, see Section 5.4.1, permuting the storage of the cores of size $r_\mu \times n_\mu \times r_{\mu+1}$ to arrays of size $r_\mu \times r_{\mu+1} \times n_\mu$ before starting the sampling procedure allows us to continuously align the matrix blocks in memory for fast calls to the BLAS routine DGEMV.

While the computational complexity, Section 5.4.3, suggests that most parts of the algorithm are equally expensive, the majority of the wall-time is spent on the calculation of the gradient, see Section 5.4.1, as it involves $|\Omega|$ operations on single indices. Inevitably, this leads to unaligned memory accesses which are a bottleneck on today's computers. This can be partly avoided by an embarrassingly parallel distribution of the loop over all indices in the sampling set Ω, but is out of the scope of this chapter.

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

5.5.1. *Scaling of the algorithm*

As a first experiment, we check the computational complexity per iteration of RTTC as derived in Section 5.4.3. In Figure 5.11, we show three plots corresponding to the scaling with respect to (left) the tensor size n, (middle) the number of dimensions d and (right) the tensor rank r. In all cases, the original data is a TT tensor of known rank r whose cores consist of entries uniformly, randomly chosen from $[0, 1]$. In the first case (left), we have chosen $d = 10$, $\mathbf{r} = (1, 10, \ldots, 10, 1)$, and mode size $n \in \{20, 40, \ldots, 200\}$. In (middle), we have set the mode size to $n = 100$, $\mathbf{r} = (1, 10, \ldots, 10, 1)$, and dimensions ranging from $d = 3$ to $d = 20$. Finally, for (right) we set $d = 10$, $n = 100$ and $\mathbf{r} = (1, r, \ldots, r, 1)$ with $r \in \{2, \ldots, 25\}$. For each configuration, we run 10 steps of RTTC with 5 repetitions. The sampling set size $|\Omega|$ is chosen such that it scales like $O(nr^2)$. The lines are linear fits for n and d and a quartic fit for tensor rank r. We observe that RTTC indeed scales linearly with n and d and to fourth order in r (due to the scaling of Ω), in accordance with (5.17).

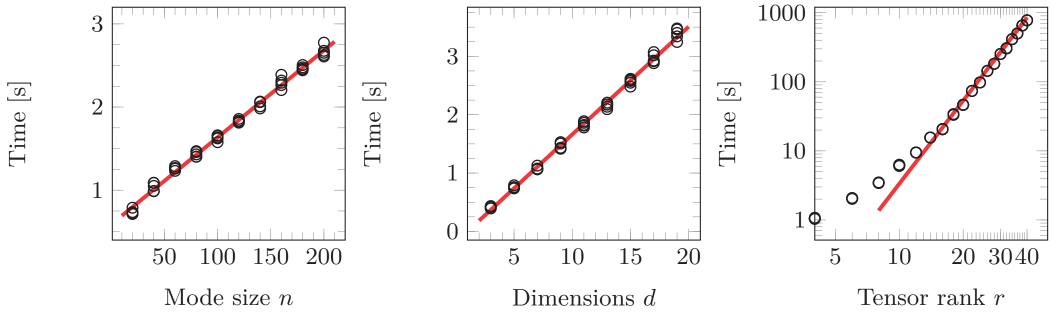


Figure 5.11: Scaling of RTTC with respect to tensor size n, number of dimensions d and tensor rank r. In the rightmost figure, in log-log scale, the line corresponds to a fourth order dependence on the tensor rank. See Section 5.5.1 for details.

5.5.2. *Comparison to an alternating linear scheme*

To evaluate the performance of our proposed algorithm, we have implemented a simple *alternating linear scheme* (ALS) algorithm to approach the tensor completion problem (5.1). This approach was suggested by Grasedyck, Kluge and Krämer [GKK15]. In the TT format, algorithms based on ALS-type optimization are a simple but surprisingly effective way to solve an optimization scheme. In each step, all cores but one are kept fixed and the optimization problem is reduced to a small optimization problem of a single core. The core is replaced by its locally optimal solution and fixed, while the optimization scheme moves to the neighboring core. This approach has been successfully used to solve linear systems and eigenvalue problems in the TT format, see e.g. [DO12, HRS12a, RU13].

For the tensor completion problem (5.1), one single core optimization step of the

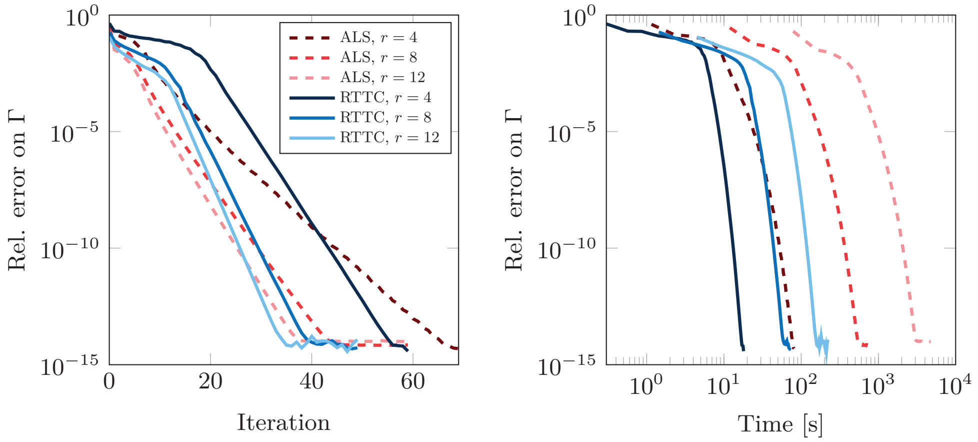


Figure 5.12: Comparison between the convergence of RTTC and ALS. The underlying tensor is a random TT tensor of size $n = 50$, $d = 10$ and known rank $r \in \{4, 8, 12\}$. ***Left:*** *Rel. error with respect to number of iterations. One iteration of ALS corresponds to one half-sweep.* ***Right:*** *Rel. error with respect to time.*

ALS scheme is given by

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

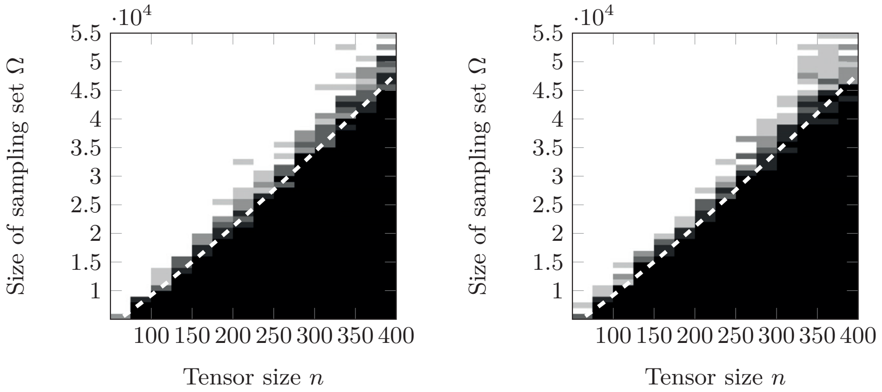
Cycling through all cores U_μ for $\mu = 1, \dots, d$ multiple times until convergence yields the full ALS completion scheme. To solve one core optimization step efficiently, we look at the i_μth slice of the new core $\widetilde{\mathbf{U}}_\mu$, making use of notation (5.16):

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

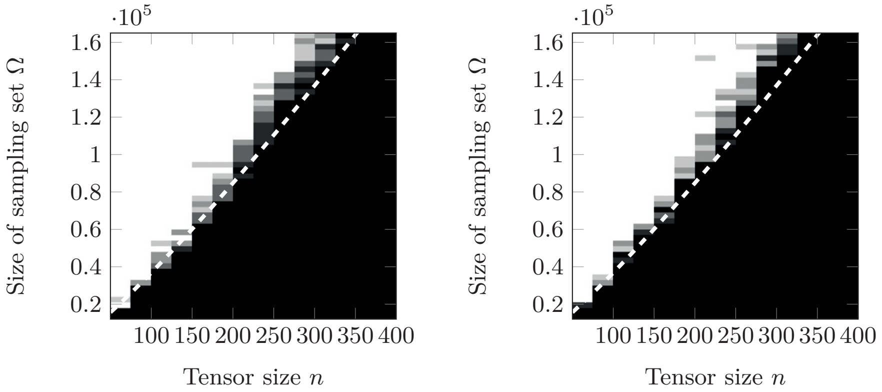
which is a linear least squares problem for $\operatorname{vec}(V(i_\mu))$ with a system matrix of size $|\Theta_{i_\mu}| \times r_{\mu-1} r_\mu$. As $\sum_{i_\mu=1}^{n_\mu} |\Theta_{i_\mu}| = |\Omega|$, solving the least squares problems to obtain the new core $\widetilde{\mathbf{U}}_\mu$ can be performed in $O(|\Omega| r^4)$ operations. Updating each core $\mu = 1, \dots, d$ once (one so-called half-sweep of the ALS procedure) then results in a total cost of $O(d|\Omega| r^4)$ operations. Compared to the complexity of RTTC, this ALS procedure involves the fourth power of the rank instead of the square. In Figure 5.12, we compare the convergence of the two approaches as a function of iterations and time when reconstructing a random TT tensor of size $n = 50$, $d = 10$ and known rank r chosen from $\{4, 8, 12\}$. While the iteration count is similar, the ALS approach takes around an order magnitude more time to converge, slowing down considerably as the rank of the underlying data increases. To make sure that the implementation is similarly optimized as RTTC, the setup of the least squares system matrices is performed in optimized MEX-functions, written in C.

5.5.3. Reconstruction of synthetic data sets

In Figure 5.13 phase plots are depicted, showing the ability of RTTC and the ALS procedure to recover a tensor as a function of the number of known values of the original tensor. The results are shown for two different numbers of dimensions, $d = 5$ and $d = 10$. We run both algorithms 5 times for each combination of tensor size n and sampling set size $|\Omega|$. The brightness represents how many times the iteration converged, where white means 5 out of 5 converged and black corresponds to no convergence. We call the iteration convergent if the relative error on the test set Γ is smaller than 10^{-4} after at most 250 iterations. Note that in almost all convergent cases, the algorithms have reached this prescribed accuracy within less than 100 steps.



(a) Phase plot for $d = 5$. **Left:** RTTC. **Right:** ALS.



(b) Phase plot for $d = 10$. **Left:** RTTC. **Right:** ALS.

Figure 5.13: Phase plots for RTTC and ALS for (a) $d = 5$ *and* (b) $d = 10$*, showing the ability to recover the original data as a function of the mode size n and the size of the sampling set* Ω*. The underlying tensor is a random TT tensor of known rank* $\mathbf{r} = (1, 3, \dots, 3, 1)$*. White means convergent in all 5 runs, black means convergent in no runs. The white dashed curve corresponds to* $20n\log(n)$ *for* $d = 5$ *and* $80n\log(n)$ *for* $d = 10$*.*

The original data $\mathbf{A}$ is constructed as a TT tensor of rank $\mathbf{r} = (1, 3, \ldots, 3, 1)$ with the entries of each core being uniform random samples from $[0, 1]$. The question of how many samples are required to recover the underlying data has been a very active topic in the last years, see Subsection 5.3.2 for a discussion. In the Tucker case, the numerical experiments suggested a scaling behaviour similar to the $O(nr \log(n))$ samples needed in the matrix case. In Figure 5.13 we have included a white dashed curve corresponding to $20n \log(n)$ for $d = 5$ and $80n \log(n)$ for $d = 10$.

5.5.4. *Interpolation of high-dimensional functions*

In the next section, we will investigate the application of tensor completion to discretizations of high dimensional functions. The random tensors considered in the previous Section 5.5.3 are constructed to be of a certain well-defined rank. For discretizations of function data, we can, in general, only hope for a sufficiently fast decay of the singular values in each matricization, resulting in good approximability by a low-rank tensor. As the rank is not known *a priori*, we will employ the rank-increasing strategy presented in Section 5.4.4, Algorithm 5.3, for all following experiments, both for RTTC and ALS.

Comparison to ADF

We compare the accuracy and timings of RTTC and the ALS procedure to an *alternating direction fitting* by Grasedyck *et al.* [GKK13] for different values of the maximum rank. We reconstruct the setup in [GKK13, Section 4.2] for RTTC and the ALS scheme to compare against the therein stated results. The original data tensor of dimension $d = 8$ and mode sizes $n = 20$ is constructed by

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

The results are shown in Table 5.2 for different values of the maximal rank $r := \max_\mu r_\mu$. Both sampling set Ω and test set Γ are chosen to be of size $|\Omega| = |\Gamma| = 10dnr^2$.

The results are shown in Table 5.2 for different values of the maximal rank $r := \max_\mu r_\mu$. We can see that the reconstruction quality is similar for all algorithms, but the ADF scheme, also implemented in MATLAB, suffers from exceedingly long computation times. A large part of these timing differences may be attributed to the fact that it does not use optimized MEX routines for the sampling. Therefore, we focus mostly on the achieved reconstruction error and only include the timings as a reference.

For the ALS procedure, the quartic instead of quadratic scaling of the complexity with respect to the rank r becomes noticeable.

	ADF		RTTC		ALS	
r	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time
2	$1.94 \cdot 10^{-2}$	14 s	$6.42 \cdot 10^{-3}$	11.1 s	$7.55 \cdot 10^{-3}$	5.90 s
3	$2.05 \cdot 10^{-3}$	1.3 min	$5.60 \cdot 10^{-3}$	25.4 s	$5.63 \cdot 10^{-3}$	19.1 s
4	$1.72 \cdot 10^{-3}$	24 min	$1.73 \cdot 10^{-3}$	45.7 s	$1.50 \cdot 10^{-3}$	46.5 s
5	$1.49 \cdot 10^{-3}$	39 min	$1.53 \cdot 10^{-3}$	1.3 min	$9.49 \cdot 10^{-4}$	1.7 min
6	$2.25 \cdot 10^{-3}$	1.7 h	$7.95 \cdot 10^{-4}$	2.1 min	$1.09 \cdot 10^{-3}$	3.3 min
7	$8.53 \cdot 10^{-4}$	2.6 h	$3.49 \cdot 10^{-4}$	3.3 min	$3.92 \cdot 10^{-4}$	6.4 min

Table 5.2: Comparison of the reconstruction accuracy of RTTC to an ADF completion algorithm [GKK13] for different values of the maximal rank. The underlying data is a discretized function resulting in a tensor of dimension $d = 8$ and mode size $n = 20$, see Section 5.5.4. The results and timings for ADF are taken from [GKK13].

Comparison to HTOpt

We now compare the reconstruction performance to a Riemannian optimization scheme on the manifold of tensors in the Hierarchical Tucker format, *HTOpt* [DSH13, DSH15].

	HTOpt		ADF		RTTC		ALS	
$^{\lvert\Omega\rvert}/_{n^d}$	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time	$\varepsilon_\Gamma(\mathbf{X}_{\mathrm{end}})$	time
0.001	$1.15 \cdot 10^{0}$	28.4 s	$6.74 \cdot 10^{-2}$	53.0 s	$9.17 \cdot 10^{-2}$	4.99 s	—	—
0.005	$1.67 \cdot 10^{0}$	28.8 s	$6.07 \cdot 10^{-3}$	3.2 m	$6.79 \cdot 10^{-3}$	5.11 s	$2.19 \cdot 10^{-2}$	3.87 s
0.01	$1.99 \cdot 10^{0}$	29.7 s	$3.08 \cdot 10^{-2}$	6.4 m	$1.19 \cdot 10^{-3}$	5.72 s	$1.59 \cdot 10^{-2}$	4.69 s
0.05	$3.68 \cdot 10^{-3}$	29.6 s	$1.95 \cdot 10^{-5}$	18 m	$2.13 \cdot 10^{-4}$	6.42 s	$5.64 \cdot 10^{-4}$	6.94 s
0.1	$1.99 \cdot 10^{-3}$	29.3 s	$6.50 \cdot 10^{-5}$	27 m	$2.29 \cdot 10^{-5}$	7.58 s	$2.09 \cdot 10^{-5}$	9.95 s

Table 5.3: Comparison of the reconstruction accuracy of RTTC to ALS, ADF and a Riemannian optimization scheme using the Hierarchical Tucker format (HTOpt) for different sizes of the sampling set. The underlying data is a discretized function resulting in a tensor of dimension $d = 4$ and mode size $n = 20$, see Section 5.5.4.

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

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

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

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

In Table 5.3 we present the results for different sizes of the sampling set $|\Omega|$ and the corresponding relative error on the test set Γ, $|\Gamma| = 100$. Sampling and test sets are chosen identically for all algorithms. Since this example is not covered in [GKK13], we now use the reference implementation of ADF provided by the authors. We observe that the proposed RTTC is the fastest and yields better reconstruction than HTOpt. For sampling ratios 0.001, 0.005, 0.01, HTOpt fails to recover the original data within the specified number of maximum iterations, 250. For the smallest sampling ratio 0.001, the ALS procedure does not have enough samples available to solve the least squares problems. Altogether, RTTC, ALS and ADF perform very similar when comparing the reconstruction quality. We assume that the reason for the worse performance of HTOpt is due to the missing rank-adaptivity. For high sampling ratios, the use of a rank adaptation scheme such as the one described in Section 5.4.4 is less important, as overfitting is very unlikely to occur.

5.5.5. Parameter-dependent PDE: Comparison to cross approximation

As an example of a parameter-dependent PDE, we investigate the cookie problem [Tob12, BG14]: Consider the heat equation on the unit square $D = [0,1]^2$ with $d = m^2$ disks $D_{s,t}$ of radius $\rho = \frac{1}{4m+2}$ and midpoints $(\rho(4s-1), \rho(4t-1))$ for $s,t = 1,\dots,m$. A graphical depiction of this setup is shown in Figure 5.14 for the cases $m = 3$ and $m = 4$. The heat conductivities of the different disks $D_{s,t}$ are described by the coefficient vector $p = (p_1,\dots,p_d)$, yielding the piecewise constant diffusion coefficient

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

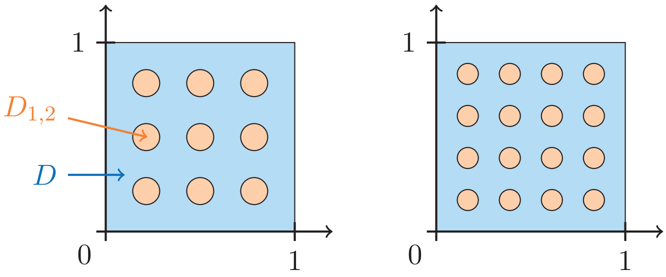
with each $p_\mu \in [\frac{1}{2}, 2]$. The temperature $u(x,p)$ at position $x \in D$ for a certain choice of parameters $p \in [\frac{1}{2}, 2]^d$ is then described by the diffusion equation

$$\begin{aligned} -\operatorname{div}(a(x,p)\nabla u(x,p)) &= 1, \qquad x \in D, \\ u(x,p) &= 0, \qquad x \in \partial D. \end{aligned} \tag{5.19}$$

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

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

Following the setup of [BG14], we discretize the parameter space $[\frac{1}{2}, 2]^d$ by spectral collocation using $n = 10$ Chebyshev points for each mode $\mu = 1,\dots d$. Hence, calculating the spatial average of the solution of (5.19) for each collocation point



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

Figure 5.14: Graphical depiction of the cookie problem for values $m = 3$ *and* $m = 4$ *resulting in 9 and 16 cookies, respectively.*

in the parameter space would result in a solution tensor $\mathbf{A}$ of size 10^d. As this represents an infeasible amount of work, we instead try to approximate the solution tensor by a low-rank approximation $\mathbf{X}$.

A common tool to tackle such problems are cross-approximation techniques. In Table 5.4 we compare the results obtained by a recent algorithm in the hierarchical Tucker format [BG14], denoted by *HT-Cross*, to the most recent cross-approximation technique in the TT format [SO11, DS14], denoted by *TT-Cross*, implemented in the routine `amen_cross` of the TT-Toolbox [ODSK14]. The results for *HT-Cross* are taken from the corresponding paper [BG14], in which the authors also compare their algorithm to a sparse grid approach. The specified tolerance is shown along with the relative reconstruction error on a test set Γ of size 100 and the number of sampling points. Both algorithms are able to construct a low-rank approximation $\mathbf{X}$ up to the prescribed accuracy, while *TT-Cross* needs about 10 times more sampling points. This could be due to an additional search along the modes of the solution tensor $\mathbf{A}$, as we have a mode size of $n = 10$ in our example. At each sampling point, the PDE (5.19) is solved for the corresponding parameter set $p = (p_1, \dots, p_d)$ using the FEniCS [LMW12] finite element package, version 1.4.0.

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

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

	TT-Cross		HT-Cross *(from [BG14])*		RTTC	
Tol.	$\varepsilon_\Gamma(\mathbf{X})$	Eval.	$\varepsilon_\Gamma(\mathbf{X})$	Eval.	$\varepsilon_\Gamma(\mathbf{X})$	$\lvert\Omega\rvert$
10^{-3}	$8.35 \cdot 10^{-4}$	11681	$3.27 \cdot 10^{-4}$	1548	$9.93 \cdot 10^{-5}$	1548
10^{-4}	$2.21 \cdot 10^{-5}$	14631	$1.03 \cdot 10^{-4}$	2784	$8.30 \cdot 10^{-6}$	2784
10^{-5}	$1.05 \cdot 10^{-5}$	36291	$1.48 \cdot 10^{-5}$	3224	$6.26 \cdot 10^{-6}$	3224
10^{-6}	$1.00 \cdot 10^{-6}$	42561	$2.74 \cdot 10^{-6}$	5338	$6.50 \cdot 10^{-7}$	5338
10^{-7}	$1.31 \cdot 10^{-7}$	77731	$1.88 \cdot 10^{-7}$	9475	$1.64 \cdot 10^{-7}$	9475

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

	TT-Cross		HT-Cross *(from [BG14])*		RTTC	
Tol.	$\varepsilon_\Gamma(\mathbf{X})$	Eval.	$\varepsilon_\Gamma(\mathbf{X})$	Eval.	$\varepsilon_\Gamma(\mathbf{X})$	$\lvert\Omega\rvert$
10^{-3}	$8.17 \cdot 10^{-4}$	22951	$3.98 \cdot 10^{-4}$	2959	$2.84 \cdot 10^{-4}$	2959
10^{-4}	$3.93 \cdot 10^{-5}$	68121	$2.81 \cdot 10^{-4}$	5261	$2.10 \cdot 10^{-5}$	5261
10^{-5}	$9.97 \cdot 10^{-6}$	79961	$1.27 \cdot 10^{-5}$	8320	$1.07 \cdot 10^{-5}$	8320
10^{-6}	$1.89 \cdot 10^{-6}$	216391	$3.75 \cdot 10^{-6}$	12736	$1.89 \cdot 10^{-6}$	12736
10^{-7}	—	—	$3.12 \cdot 10^{-7}$	26010	$7.12 \cdot 10^{-7}$	26010

(b) 16 cookies ($m = 4$)

Table 5.4: Reconstruction results for the cookie problem using cross-approximation and tensor completion. The last entry in Table (b) for TT-Cross has been omitted due to an exceedingly high amount of function evaluations. The results for HT-Cross have been taken from the corresponding paper [BG14].

5.6. Conclusion

We have shown that the framework of Riemannian optimization yields a very effective nonlinear CG method for performing tensor completion. More specifically, we have derived algorithms based on the manifold of fixed multilinear rank (GeomCG) and of fixed tensor train rank (RTTC). One of the main contributions consists of a careful discussion of the algorithmic and implementation details, showing that the method scales well for large data sets and is competitive to existing methods for tensor completion.

For applications, a rank increasing scheme was shown to be essential to be able to recover the original data from very few samples. We have investigated several different strategies for the Tucker case and present a computationally feasible scheme for the TT case. In the numerical experiments, we have shown applicability of tensor completion to the reconstruction of data sets such as hyperspectral images and function-related tensors. Furthermore, tensor completion with randomly chosen samples was shown to yield similar reconstruction errors as an adaptive cross-approximation approach with the same number of samples.

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

Chapter 6

▶ Application 2: Linear Systems

This chapter is concerned with the approximate solution of large-scale linear systems $Ax = f$ with $A \in \mathbb{R}^{N \times N}$. In certain applications, such as the structured discretization of d-dimensional partial differential equations (PDEs), the size of the linear system naturally decomposes as $N = n_1 n_2 \cdots n_d$ with $n_\mu \in \mathbb{N}$ for $\mu = 1, \ldots, d$. This allows us to view $Ax = f$ as a tensor equation

$$\mathcal{A}\mathbf{X} = \mathbf{F}, \tag{6.1}$$

where $\mathbf{F}, \mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ are tensors of order d and $\mathcal{A}$ is a linear operator on $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$.

The tensor equations considered in this chapter admit a decomposition of the form

$$\mathcal{A} = \mathcal{L} + \mathcal{V}, \tag{6.2}$$

where $\mathcal{L}$ is a Laplace-like operator with the matrix representation

$$L = I_{n_d} \otimes \cdots \otimes I_{n_2} \otimes L_1 + I_{n_d} \otimes \cdots \otimes I_{n_3} \otimes L_2 \otimes I_{n_1} + \cdots + L_d \otimes I_{n_{d-1}} \otimes \cdots \otimes I_1, \tag{6.3}$$

with matrices $L_\mu \in \mathbb{R}^{n_\mu \times n_\mu}$ and identity matrices I_{n_μ}. The term $\mathcal{V}$ is dominated by $\mathcal{L}$ in the sense that $\mathcal{L}$ is assumed to be a good preconditioner for $\mathcal{A}$. Equations of this form arise, for example, from the discretization of the Schrödinger Hamiltonian [Lub08], for which $\mathcal{L}$ and $\mathcal{V}$ correspond to the discretization of the kinetic and the potential energy terms, respectively. In this application, $\mathcal{A}$ (and thus also L_μ) is symmetric positive definite. In the following, we restrict ourselves to this case, although some of the developments can, in principle, be generalized to indefinite and nonsymmetric matrices.

Assuming $\mathcal{A}$ to be symmetric positive definite allows us to reformulate (6.1) as an optimization problem

$$\min_{\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}} \frac{1}{2}\langle \mathbf{X}, \mathcal{A}\mathbf{X} \rangle - \langle \mathbf{X}, \mathbf{F} \rangle. \tag{6.4}$$

It is well-known that the above problem is equivalent to minimizing the $\mathcal{A}$-induced norm of the error $\|\mathbf{X} - \mathcal{A}^{-1}\mathbf{F}\|_{\mathcal{A}}$. Neither (6.1) nor (6.4) are computationally tractable for larger values of d. During the last decade, low-rank tensor techniques have been developed that aim at dealing with this curse of dimensionality by approximating $\mathbf{F}$ and $\mathbf{X}$ in a compressed format; see [GKT13, Hac12] for overviews. One approach consists of restricting (6.4) to a subset $\mathcal{M} \subset \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ of low-rank tensors:

$$\min_{\mathbf{X} \in \mathcal{M}} f(\mathbf{X}) := \frac{1}{2}\langle \mathbf{X}, \mathcal{A}\mathbf{X} \rangle - \langle \mathbf{X}, \mathbf{F} \rangle. \tag{6.5}$$

In this chapter, we focus on the cases where $\mathcal{M}$ is either the set of tensors of fixed multilinear or tensor train rank as introduced in Chapters 3 and 4, respectively. We will then use Riemannian optimization techniques, see Chapter 2, to address (6.5).

Similar to Euclidean optimization, the condition number of the Riemannian Hessian of the objective function is instrumental in predicting the performance of first-order optimization algorithms on manifolds; see Theorem 2.27. As will be evident from (6.14) in Section 6.2.1, an ill-conditioned operator $\mathcal{A}$ can be expected to yield an ill-conditioned Riemannian Hessian. As this is the case for the applications we consider, any naive first-order method will be prohibitively slow and noncompetitive with existing methods. This is a fundamental difference to the tensor completion problem which we considered in the previous Chapter 5, which is intrinsically well-conditioned.

For Euclidean optimization, it is well known that preconditioning or, equivalently, adapting the underlying metric can be used to address the slow convergence of such first-order methods. Combining steepest descent with the Hessian as a (variable) preconditioner yields the Newton method with (local) second order convergence [Nes04, Sec. 1.3.1]. To overcome the high computational cost associated with Newton's method, several approximate Newton methods exist that use cheaper second-order models. For example, Gauss–Newton is a particularly popular approximation when solving non-linear least-squares problems. For Riemannian optimization, the connection between preconditioning and adapting the metric is less immediate and we explore both directions to speed up first-order methods. On the one hand, we will consider a rather ad hoc way to precondition the Riemannian gradient direction. On the other hand, we will consider an approximate Newton method that can be interpreted as a constrained Gauss–Newton method. This requires setting up and solving linear systems with the Riemannian Hessian or an approximation thereof. In [VV10], it was shown that neglecting curvature terms in the Riemannian Hessian leads to an efficient low-rank solver for Lyapunov matrix equations. We will extend these developments to more general equations with tensors approximated in the Tucker and the TT formats.

Riemannian optimization is by no means the only sensible approach to finding low-rank tensor approximations to the solution of the linear system (6.1). For linear operators only involving the Laplace-like operator (6.3), exponential sum approximations [Gra04, HK06] and tensorized Krylov subspace methods [KT10] are effective and allow for a thorough convergence analysis. For more general equations, a straightforward approach is to apply standard iterative methods, such as the Richardson iteration or the CG method, to (6.1) and represent all iterates in the low-rank tensor format; see [BG13, Dol13, KO10b, KS11, KT11a] for examples. One critical issue in this approach is to strike a balance between maintaining convergence and avoiding excessive intermediate rank growth of the iterates. Only recently, this has been analyzed in more detail [BD15]. A very different approach consists of applying alternating optimization techniques to the constrained optimization problem (6.5). Such methods have originated in quantum physics, most notably the so called DMRG method to address eigenvalue problems in the context of strongly correlated quantum lattice systems, see [Sch11] for an overview. The ideas of DMRG and related methods have been extended to linear systems in the numerical analysis community in [DO12, DS14, HRS12a, Ose11a] and are generally referred to as alternating linear schemes (ALS). While such methods often exhibit fast convergence, especially for operators of the form (6.2), their global convergence properties are poorly understood. Even the existing local convergence results for ALS [RU13, UV13] offer little intuition on the convergence *rate*. The efficient implementation of ALS for low-rank tensor formats can be a challenge. In the presence of larger ranks, the (dense) subproblems that need to be solved in every step of ALS are large and tend to be ill-conditioned. In [KT11b] and Chapter 7, this issue is addressed by combining an iterative solver with a preconditioner tailored to the subproblem. The design of such a preconditioner is by no means simple, even the knowledge of an effective preconditioner for the full-space problem (6.1) is generally not sufficient. So far, the only known effective preconditioners are based on exponential sum approximations for operators with Laplace-like structure (6.3), which is inherited by the subproblems.

Compared to existing approaches, the preconditioned low-rank Riemannian optimization methods proposed in this paper have a number of advantages. Due to imposing the manifold constraint, the issue of rank growth is completely avoided. Our methods have a global nature, all components of the low-rank tensor format are improved at once and hence the stagnation typically observed during ALS sweeps is avoided. Moreover, we completely avoid the need for solving subproblems very accurately. One of our methods can make use of preconditioners for the full-space problem (6.1), while for the other methods preconditioners are implicitly obtained from approximating the Riemannian Hessian. A disadvantage shared with existing methods, our method strongly relies on the decomposition (6.2) of the operator to construct effective preconditioners.

In passing, we mention that there is another notion of preconditioning for Riemannian optimization on a low-rank matrix manifold, see, e.g., [MS14a, MS14b, NS12]. These techniques address the ill-conditioning of the manifold parametrization, an aspect that is not related and relevant to our developments, as we do not directly work with the parametrization.

This chapter is based on the technical report [KSV15].

6.1. First-order Riemannian optimization and preconditioning

In this section, we discuss ways to incorporate preconditioners into simple first-order Riemannian optimization methods. Throughout this section, $\mathcal{M}_{\mathbf{r}}$ denotes either the manifold of tensors of fixed multilinear or TT rank, unless otherwise specified.

6.1.1. Riemannian gradient descent

For the cost function (6.5) associated with linear systems, the Euclidean gradient is given by

$$\operatorname{Grad} f(\mathbf{X}) = \mathcal{A}\mathbf{X} - \mathbf{F}$$

and the Riemannian gradient is obtained by projection,

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

Together with the retraction R of Section 3.6 and Section 4.5, this yields the basic Riemannian gradient descent algorithm:

$$\mathbf{X}_{k+1} = R(\mathbf{X}_k + \alpha_k \xi_k), \quad \text{with} \quad \xi_k = -\,\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}} \operatorname{Grad} f(\mathbf{X}_k). \tag{6.6}$$

To highlight the similarities of the iterate-and-truncate and Riemannian optimization schemes, we slightly abuse the notation of the retraction R by defining its domain of definition to be the full space $\mathbb{R}^{n_1 \times \cdots \times n_d}$ instead of the tangent bundle $T\mathcal{M}$. This only makes sense for this particular choice of SVD-based retractions, $\mathrm{P}_{\mathbf{r}}^{\mathrm{HO}}(\mathbf{X}_k + \alpha_k \xi_k)$ and $\mathrm{P}_{\mathbf{r}}^{\mathrm{TT}}(\mathbf{X}_k + \alpha_k \xi_k)$ for the Tucker and TT format, respectively.

The linearized line search procedure according to Section 2.8.1 is given for the cost function (6.5) by

$$\operatorname*{argmin}_{\alpha} f(\mathbf{X}_k + \alpha \xi_k) = -\frac{\langle \xi_k, \operatorname{Grad} f(\mathbf{X}_k) \rangle}{\langle \xi, \mathcal{A}\xi \rangle}. \tag{6.7}$$

6.1.2. Truncated preconditioned Richardson iteration

Truncated Richardson iteration. The Riemannian gradient descent defined by (6.6) closely resembles a truncated Richardson iteration for solving linear systems:

$$\mathbf{X}_{k+1} = R(\mathbf{X}_k + \alpha_k \xi_k), \quad \text{with} \quad \xi_k = -\operatorname{Grad} f(\mathbf{X}_k) = \mathbf{F} - \mathcal{A}\mathbf{X}_k, \tag{6.8}$$

which was proposed for the CP tensor format in [KS11]. For the hierarchical Tucker format, a variant of the TT format, the iteration (6.8) has been analyzed in [BD15]. In contrast to manifold optimization, the rank does not need to be fixed but can be adjusted to strike a balance between low rank and convergence speed.

Without the rank truncation, the Richardson iteration is a classic Euclidean steepest descent algorithm for the cost function (6.4). In this case, the linearized line search 2.8.1 is exact and convergence results are well-known. For example, one can show using Kantorovich's inequality that the energy norm of the distance to the exact solution $\mathbf{X}_*$ fulfills [NW06, Thm. 3.3]

$$\|\mathbf{X}_{k+1} - \mathbf{X}_*\|_{\mathcal{A}} \leq \frac{\kappa(\mathcal{A}) - 1}{\kappa(\mathcal{A}) + 1} \|\mathbf{X}_k - \mathbf{X}_*\|_{\mathcal{A}},$$

where $\kappa(\mathcal{A})$ is the condition number of the matrix representation A of $\mathcal{A}$. Thus, the convergence of the Richardson iteration depends crucially on the conditioning of the linear system. For the rank-truncated Richardson it also has been observed, for example in [KPT14], that it greatly benefits from preconditioners, not only to attain an acceptable convergence speed but also to avoid excessive rank growth of the intermediate iterates.

Preconditioned Richardson iteration. For the standard Richardson iteration $\mathbf{X}_{k+1} = \mathbf{X}_k - \alpha_k \xi_k$ a symmetric positive definite preconditioner $\mathcal{P}$ for $\mathcal{A}$ can be incorporated as follows:

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \alpha_k \mathcal{P}^{-1} \xi_k \quad \text{with} \quad \xi_k = \mathbf{F} - \mathcal{A}\mathbf{X}_k. \tag{6.9}$$

Using the Cholesky factorization $\mathcal{P} = \mathcal{C}\mathcal{C}^{\mathsf{T}}$, this iteration turns out to be equivalent to applying the Richardson iteration to the transformed symmetric positive definite linear system

$$\mathcal{C}^{-1}\mathcal{A}\mathcal{C}^{-\mathsf{T}}\mathbf{Y} = \mathcal{C}^{-1}\mathbf{F}$$

after changing coordinates by $\mathcal{C}^{\mathsf{T}}\mathbf{X}_k$. At the same time, (6.9) can be viewed as applying gradient descent in the inner product induced by $\mathcal{P}$.

Truncated preconditioned Richardson iteration. The most natural way of combining truncation with preconditioning leads to the *truncated preconditioned Richardson iteration*

$$\mathbf{X}_{k+1} = R\big(\mathbf{X}_k + \alpha_k \mathcal{P}^{-1} \xi_k\big), \quad \text{with} \quad \xi_k = \mathbf{F} - \mathcal{A}\mathbf{X}_k, \tag{6.10}$$

see also [KS11]. In view of Riemannian gradient descent (6.6), it appears natural to project the search direction to the tangent space, leading to the "geometric" variant

$$\mathbf{X}_{k+1} = R\big(\mathbf{X}_k + \alpha_k \, \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}} \, \mathcal{P}^{-1} \xi_k\big), \quad \text{with} \quad \xi_k = \mathbf{F} - \mathcal{A}\mathbf{X}_k. \tag{6.11}$$

In terms of convergence, we have observed that the scheme (6.11) behaves similar to (6.10); see §6.3.3. However, it can be considerably cheaper per iteration: Since only tangent vectors need to be retracted in (6.11), the computation of the HOSVD/TT-SVD in R involves only tensors of bounded rank, regardless of the rank of $\mathcal{P}^{-1}\xi_k$. In particular, with $\mathbf{r}$ the multilinear or TT rank of $\mathbf{X}_k$, the corresponding rank of $\mathbf{X}_k + \alpha_k \, \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}} \, \mathcal{P}^{-1}\xi_k$ is at most $2\mathbf{r}$; see Section 3.6 and Section 4.5, respectively. On the other hand, in (6.10) the rank of $\mathbf{X}_k + \alpha_k \mathcal{P}^{-1}\xi_k$ is determined primarily by the quality of the preconditioner $\mathcal{P}$ and can possibly be very large.

Another advantage occurs for the special but important case when $\mathcal{P}^{-1} = \sum_{\alpha=1}^{s} \mathcal{P}_\alpha$, where each term $\mathcal{P}_\alpha$ is relatively cheap to apply. For example, when $\mathcal{P}^{-1}$ is an exponential sum preconditioner [BH05] then $\mathcal{P}_\alpha$ is a Kronecker product of small matrices. By the linearity of $\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}$, we have

$$\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}} \, \mathcal{P}^{-1}\xi_k = \sum_{\alpha=1}^{s} \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}} \, \mathcal{P}_\alpha \xi_k, \tag{6.12}$$

which makes it often cheaper to evaluate this expression in the iteration (6.11). To see this, for example, for the TT format, suppose that $\mathcal{P}_\alpha \xi$ has TT ranks r_p. Then the preconditioned direction $\mathcal{P}^{-1}\xi_k$ can be expected to have TT ranks sr_p. Hence, the straightforward application of $\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}$ to $\mathcal{P}^{-1}\xi_k$ requires $O(dn(sr_p)^2 r)$ operations. Using the expression on the right-hand side of (6.12) instead reduces the cost to $O(dnsr_p^2 r)$ operations, since the summation of tangent vectors amounts to simply adding their parametrizations. In contrast, since the retraction is a non-linear operation, trying to achieve similar cost savings in (6.10) by simply truncating the cumulated sum subsequently may lead to severe cancellation [KT14, Sec. 6.3].

6.2. Riemannian optimization using a quadratic model

As we will see in the numerical experiments in Section 6.3, the convergence of the first-order methods presented above crucially depends on the availability of a good preconditioner for the full problem. In this section, we present Riemannian optimization methods based on a quadratic model. In these methods, the preconditioners are derived from an approximation of the Riemannian Hessian.

6.2.1. Approximate Newton method

The Riemannian Newton method [AMS08] applied to (6.5) determines the search direction ξ_k from the equation

$$\mathcal{H}_{\mathbf{X}_k}\xi_k = -\,\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\mathrm{Grad}\,f(\mathbf{X}_k), \tag{6.13}$$

where the symmetric linear operator $\mathcal{H}_{\mathbf{X}_k} : T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}} \to T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}$ is the Riemannian Hessian of (6.5). Using [AMT13], we have

$$\begin{aligned}\mathcal{H}_{\mathbf{X}_k} &= \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\left[\mathrm{Hess}\,f(\mathbf{X}_k) + J_{\mathbf{X}_k}\,\mathrm{Grad}\,f(\mathbf{X}_k)\right]\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\\ &= \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\left[\mathcal{A} + J_{\mathbf{X}_k}(\mathcal{A}\mathbf{X}_k - \mathbf{F})\right]\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\end{aligned} \tag{6.14}$$

with the Euclidean Hessian $\mathrm{Hess}\,f(\mathbf{X}_k)$ and the Fréchet derivative[1] $J_{\mathbf{X}_k}$ of $\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}$ at $\mathbf{X}_k$.

As usual, the Newton equation is only well-defined near a strict local minimizer and solving it exactly is prohibitively expensive in a large-scale setting. We therefore approximate the linear system (6.13) in two steps: First, we drop the term containing $J_{\mathbf{X}_k}$ and second, we replace $\mathcal{A} = \mathcal{L} + \mathcal{V}$ by $\mathcal{L}$. The first approximation can be interpreted as neglecting the curvature of $\mathcal{M}_{\mathbf{r}}$, or equivalently, as linearizing the manifold at $\mathbf{X}_k$. Indeed, this term is void if $\mathcal{M}_{\mathbf{r}}$ would be a (flat) linear subspace. This approximation is also known as constrained Gauss–Newton (see, e.g, [Boc87]) since it replaces the constraint $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$ with its linearization $\mathbf{X} \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ and neglects the constraints in the Lagrangian. The second approximation is natural given the assumption of $\mathcal{L}$ being a good preconditioner for $\mathcal{A} = \mathcal{L} + \mathcal{V}$. In addition, our derivations and numerical implementation will rely extensively on the fact that the Laplacian $\mathcal{L}$ acts on each tensor dimension separately.

The result is an *approximate Newton method* where the search direction ξ_k is determined from

$$\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\,\mathcal{L}\,\mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}}\,\xi_k = \mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}(\mathbf{F} - \mathcal{A}\mathbf{X}_k). \tag{6.15}$$

Since $\mathcal{L}$ is positive definite, this equation is always well-defined for any $\mathbf{X}_k$. In addition, ξ_k is also gradient-related and hence the iteration

$$\mathbf{X}_{k+1} = R(\mathbf{X}_k + \alpha_k\xi_k)$$

---

[1] $J_{\mathbf{X}_k}$ is an operator from $\mathbb{R}^{n\times n\times\cdots\times n}$ to the space of self-adjoint linear operators $T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}} \to T_{\mathbf{X}_k}\mathcal{M}_{\mathbf{r}}$.

is guaranteed to converge globally to a stationary point of the cost function if α_k is determined from Armijo backtracking, see Theorem 2.24.

Despite all the simplifications, the numerical solution of (6.15) turns out to be a nontrivial task. In the following section, we explain an efficient algorithm for solving (6.15) exactly when $\mathcal{M}_{\mathbf{r}}$ is the Tucker manifold. For the TT manifold, this approach is no longer feasible and we therefore present an effective preconditioner that can be used for solving (6.15) with the preconditioned CG method.

6.2.2. The approximate Riemannian Hessian in the Tucker case

The solution of the linear system (6.15) was addressed for the matrix case ($d = 2$) in [VV10, Sec. 7.2]. In the following, we extend this approach to tensors in the Tucker format. To keep the presentation concise, we restrict ourselves to $d = 3$; the extension to $d > 3$ is straightforward.

For tensors of order three in the Tucker format, we write (6.15) as follows:

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\mathcal{L}\,\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\xi = \eta, \tag{6.16}$$

where

- $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$ is parametrized by factor matrices U_1, U_2, U_3 having *orthonormal columns* and the core tensor $\mathbf{S}$;
- the right-hand side $\eta \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is given in terms of its *gauged* parametrization $\delta U_1^\eta, \delta U_2^\eta, \delta U_3^\eta$ and $\delta \mathbf{S}^\eta$, as in (3.12) and (3.14);
- the unknown $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is to be determined in terms of its *gauged* parametrization $\delta U_1, \delta U_2, \delta U_3$ and $\delta\mathbf{S}$, again as in (3.12) and (3.14).

To derive equations for δU_μ with $\mu = 1, 2, 3$ and $\delta\mathbf{S}$ we exploit that $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ decomposes orthogonally into $\mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_4$; see (3.13). This allows us to split (6.16) into a system of four coupled equations by projecting onto $\mathcal{V}_\mu$ for $\mu = 1, \ldots, 4$.

In particular, since $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ by assumption, we can insert $\mathbf{Z} := \mathcal{L}\,\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\xi = \mathcal{L}\xi$ into the equations of Prop. 3.8. By exploiting the structure of $\mathcal{L}$ (see (6.3)) and the orthogonality of the gauged representation of tangent vectors (see (3.14)), we can simplify the expressions considerably and arrive at the equations

$$\begin{aligned}
\delta U_1^\eta &= \mathrm{P}_{U_1}^{\perp}\Big(L_1 U_1 \delta S_{(1)} + L_1 \delta U_1 S_{(1)} + \delta U_1 S_{(1)}\big[I_{r_3} \otimes U_2^\mathsf{T} L_2 U_2 + U_3^\mathsf{T} L_3 U_3 \otimes I_{r_2}\big]\Big) S_{(1)}^\dagger \\
\delta U_2^\eta &= \mathrm{P}_{U_2}^{\perp}\Big(L_2 U_2 \delta S_{(2)} + L_2 \delta U_2 S_{(2)} + \delta U_2 S_{(2)}\big[I_{r_3} \otimes U_1^\mathsf{T} L_1 U_1 + U_3^\mathsf{T} L_3 U_3 \otimes I_{r_1}\big]\Big) S_{(2)}^\dagger \\
\delta U_3^\eta &= \mathrm{P}_{U_3}^{\perp}\Big(L_3 U_3 \delta S_{(3)} + L_3 \delta U_3 S_{(3)} + \delta U_3 S_{(3)}\big[I_{r_2} \otimes U_1^\mathsf{T} L_1 U_1 + U_2^\mathsf{T} L_2 U_2 \otimes I_{r_1}\big]\Big) S_{(3)}^\dagger \\
\delta \mathbf{S}^\eta &= \big[U_1^\mathsf{T} L_1 U_1 \delta S_{(1)} + U_1^\mathsf{T} L_1 \delta U_1 S_{(1)}\big]^{(1)} + \big[U_2^\mathsf{T} L_2 U_2 \delta S_{(2)} + U_2^\mathsf{T} L_2 \delta U_2 S_{(2)}\big]^{(2)} \\
&\qquad + \big[U_3^\mathsf{T} L_3 U_3 \delta S_{(3)} + U_3^\mathsf{T} L_3 \delta U_3 S_{(3)}\big]^{(3)}.
\end{aligned} \tag{6.17}$$

Additionally, the gauge conditions need to be satisfied:

$$U_1^\mathsf{T} \delta U_1 = U_2^\mathsf{T} \delta U_2 = U_3^\mathsf{T} \delta U_3 = 0. \tag{6.18}$$

In order to solve these equations, we will use the first three equations of (6.17), together with (6.18), to substitute δU_μ in the last equation of (6.17) and determine a decoupled equation for $\delta \mathbf{S}$. Rearranging the first equation of (6.17), we obtain

$$\begin{aligned} \mathrm{P}_{U_1}^{\perp} \Big(L_1 \delta U_1 + \delta U_1 S_{(1)} [I_{r_3} \otimes U_2^{\mathsf{T}} L_2 U_2 + U_3^{\mathsf{T}} L_3 U_3 \otimes I_{r_2}] S_{(1)}^{\dagger} \Big) & \\ = \delta U_1^{\eta} - \mathrm{P}_{U_1}^{\perp} L_1 U_1 \delta S_{(1)} S_{(1)}^{\dagger}. & \end{aligned}$$

Vectorization and adhering to (6.18) yields the saddle point system

$$\begin{bmatrix} G & I_{r_1} \otimes U_1 \\ I_{r_1} \otimes U_1^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} \operatorname{vec}(\delta U_1) \\ y_1 \end{bmatrix} = \begin{bmatrix} b_1 \\ 0 \end{bmatrix}, \tag{6.19}$$

where

$$\begin{aligned} G &= I_{r_1} \otimes L_1 + (S_{(1)}^{\dagger})^{\mathsf{T}} (I_{r_3} \otimes U_2^{\mathsf{T}} L_2 U_2 + U_3^{\mathsf{T}} L_3 U_3 \otimes I_{r_2}) S_{(1)}^{\mathsf{T}} \otimes I_{n_1}, \\ b_1 &= \operatorname{vec}(\delta U_1^{\eta}) - ((S_{(1)}^{\dagger})^{\mathsf{T}} \otimes \mathrm{P}_{U_1}^{\perp} L_1 U_1) \operatorname{vec}(\delta S_{(1)}), \end{aligned}$$

and $y_1 \in \mathbb{R}^{r_1^2}$ is the dual variable. The positive definiteness of L_1 and the full rank conditions on U_1 and $\mathbf{S}$ imply that the above system is nonsingular; see, e.g., [BGL05]. Using the Schur complement $G_S = -(I_{r_1} \otimes U_1)^{\mathsf{T}} G^{-1} (I_{r_1} \otimes U_1)$, we obtain the explicit expression

$$\operatorname{vec}(\delta U_1) = \Big(I_{n_1 r_1} + G^{-1} (I_{r_1} \otimes U_1) G_S^{-1} (I_{r_1} \otimes U_1^{\mathsf{T}}) \Big) G^{-1} b_1 = w_1 - F_1 \operatorname{vec}(\delta S_{(1)}), \tag{6.20}$$

with

$$\begin{aligned} w_1 &:= \Big(I_{n_1 r_1} + G^{-1} (I_{r_1} \otimes U_1) G_S^{-1} (I_{r_1} \otimes U_1^{\mathsf{T}}) \Big) G^{-1} \operatorname{vec}(\delta U_1^{\eta}), \\ F_1 &:= \Big(I_{n_1 r_1} + G^{-1} (I_{r_1} \otimes U_1) G_S^{-1} (I_{r_1} \otimes U_1^{\mathsf{T}}) \Big) G^{-1} \Big((S_{(1)}^{\dagger})^{\mathsf{T}} \otimes \mathrm{P}_{U_1}^{\perp} L_1 U_1 \Big). \end{aligned}$$

Expressions analogous to (6.20) can be derived for the other two factor matrices:

$$\begin{aligned} \operatorname{vec}(\delta U_2) &= w_2 - F_2 \operatorname{vec}(\delta S_{(2)}), \\ \operatorname{vec}(\delta U_3) &= w_3 - F_3 \operatorname{vec}(\delta S_{(3)}), \end{aligned}$$

with suitable analogs for w_2, w_3, F_2, and F_3. These expressions are now inserted into the last equation of (6.17) for $\delta \mathbf{S}^{\eta}$. To this end, define permutation matrices $\Pi_{i \to j}$ that map the vectorization of the ith matricization to the vectorization of the jth matricization:

$$\Pi_{i \to j} \operatorname{vec}(\delta \mathbf{S}_{(i)}) = \operatorname{vec}(\delta \mathbf{S}_{(j)}),$$

By definition, $\operatorname{vec}(\delta S_{(1)}) = \operatorname{vec}(\delta \mathbf{S})$, and we finally obtain the following linear system for $\operatorname{vec}(\delta \mathbf{S})$:

$$\begin{aligned} F \operatorname{vec}(\delta \mathbf{S}) = \operatorname{vec}(\delta \mathbf{S}^{\eta}) - (S_{(1)}^{\mathsf{T}} \otimes U_1^{\mathsf{T}} L_1) w_1 - \Pi_{2 \to 1} (S_{(2)}^{\mathsf{T}} \otimes U_2^{\mathsf{T}} L_2) w_2 & \\ - \Pi_{3 \to 1} (S_{(3)}^{\mathsf{T}} \otimes U_3^{\mathsf{T}} L_3) w_3, & \end{aligned} \tag{6.21}$$

with the $r_1r_2r_3 \times r_1r_2r_3$ matrix

$$\begin{aligned} F := I_{r_2r_3} \otimes U_1^\mathsf{T} L_1 U_1 - (S_{(1)}^\mathsf{T} \otimes U_1^\mathsf{T} L_1) F_1 \\ + \Pi_{2\to 1} \Big[I_{r_1r_3} \otimes U_2^\mathsf{T} L_2 U_2 - (S_{(2)}^\mathsf{T} \otimes U_2^\mathsf{T} L_2) F_2 \Big] \Pi_{1\to 2} \\ + \Pi_{3\to 1} \Big[I_{r_1r_2} \otimes U_3^\mathsf{T} L_3 U_3 - (S_{(3)}^\mathsf{T} \otimes U_3^\mathsf{T} L_3) F_3 \Big] \Pi_{1\to 3}. \end{aligned}$$

For small ranks, the linear system (6.21) is solved by forming the matrix F explicitly and using a direct solver. Since this requires $O(r_1^3 r_2^3 r_3^3)$ operations, it is advisable to use an iterative solver for larger ranks, in which the Kronecker product structure can be exploited when applying F; see also [VV10]. Once we have obtained $\delta\mathbf{S}$, we can easily obtain $\delta U_1, \delta U_2, \delta U_3$ using (6.20).

Remark 6.1. *The application of* G^{-1} *needed in* (6.20) *as well as in the construction of* G_S *can be implemented efficiently by noting that* G *is the matrix representation of the Sylvester operator* $V \mapsto L_1 V + V \Gamma_1^\mathsf{T}$, *with the matrix*

$$\Gamma_1 := (S_{(1)}^\dagger)^\mathsf{T} (I_{r_3} \otimes U_2^\mathsf{T} L_2 U_2 + U_3^\mathsf{T} L_3 U_3 \otimes I_{r_2}) S_{(1)}^\mathsf{T}.$$

The $r_1 \times r_1$ *matrix* Γ_1 *is non-symmetric but it can be diagonalized by first computing a QR decomposition* $S_{(1)}^\mathsf{T} = Q_S R_S$ *such that* $Q_S^T Q_S = I_{r_1}$ *and then computing the spectral decomposition of the symmetric matrix*

$$Q_S^\mathsf{T} (I_{r_3} \otimes U_2^\mathsf{T} L_2 U_2 + U_3^\mathsf{T} L_3 U_3 \otimes I_{r_2}) Q_S.$$

After diagonalization of Γ_1, *the application of* G^{-1} *requires the solution of* r_1 *linear systems with the matrices* $L_1 + \lambda I$, *where* λ *is an eigenvalue of* Γ_1; *see also [Sim13]. The Schur complement* $G_S \in \mathbb{R}^{r_1^2 \times r_1^2}$ *is constructed explicitly by applying* G^{-1} *to the* r_1^2 *columns of* $I_{r_1} \otimes U_1$.

Analogous techniques apply to the computation of w_2, F_2, *and* w_3, F_3.

Assuming, for example, that each L_μ is a tri-diagonal matrix, the solution of a linear system with the shifted matrix $L_\mu + \lambda I$ can be performed in $O(n)$ operations. Therefore, using Remark 6.1, the construction of the Schur complement G_S requires $O(nr^3)$ operations. Hence, the approximate Newton equation (6.16) can be solved in $O(dnr^3 + r^9)$ operations. This cost dominates the complexity of the Riemannian gradient calculation and the retraction step.

6.2.3. The approximate Riemannian Hessian in the TT case

When using the TT format, it seems to be much harder to solve the approximate Newton equation (6.15) directly and we therefore resort to the preconditioned conjugate gradient (PCG) method for solving the linear system iteratively. We use the following commonly used stopping criterion [NW06, Ch. 7.1] for accepting the approximation $\tilde{\xi}$ produced by PCG:

$$\| \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}} [\mathcal{L}\tilde{\xi} - \nabla f(\mathbf{X}_k)] \| \le \min\Big(0.5, \sqrt{\| \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}} \nabla f(\mathbf{X}_k) \|}\Big) \cdot \| \mathrm{P}_{T_{\mathbf{X}_k}\mathcal{M}_\mathbf{r}} \nabla f(\mathbf{X}_k) \|.$$

To derive an effective preconditioner for PCG, we first examine the approximate Newton equation (6.15) more closely. For d-dimensional tensors in the TT format, it takes the form

$$\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\mathcal{L}\,\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}\,\xi = \eta, \tag{6.22}$$

where

- $\mathbf{X} \in \mathcal{M}_{\mathbf{r}}$ is parametrized by its cores $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_d$ and is *d-orthogonal*;
- the right-hand side $\eta \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is represented in terms of its *gauged* parametrization $\delta\mathbf{U}_1^\eta$, $\delta\mathbf{U}_2^\eta$, …, $\delta\mathbf{U}_d^\eta$, as in (4.12);
- the unknown $\xi \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ needs to be determined in terms of its *gauged* parametrization $\delta\mathbf{U}_1, \delta\mathbf{U}_2, \ldots, \delta\mathbf{U}_d$, again as in (4.12).

When PCG is applied to (6.22) with a preconditioner $\mathcal{B}\colon T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}} \to T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$, we need to evaluate an expression of the form $\xi = \mathcal{B}\eta$ for a given, arbitrary vector $\eta \in T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$. Again, ξ and η are represented using the gauged parametrization above.

We will present two block Jacobi preconditioners for (6.22); both are variants of parallel subspace correction (PSC) methods [Xu92]. They mainly differ in the way the tangent space $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is split into subspaces.

A block diagonal Jacobi preconditioner

The most immediate choice for splitting $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}$ is to simply take the direct sum (4.10). The PSC method is then defined in terms of the local operators

$$\mathcal{L}_\mu\colon \mathcal{V}_\mu \to \mathcal{V}_\mu, \qquad \mathcal{L}_\mu = \mathrm{P}_{\mathcal{V}_\mu}\,\mathcal{L}\,\mathrm{P}_{\mathcal{V}_\mu}\big|_{\mathcal{V}_\mu}, \qquad \mu = 1, \ldots, d,$$

where $\mathrm{P}_{\mathcal{V}_\mu}$ is the orthogonal projector onto $\mathcal{V}_\mu$; see Prop. 4.9. The operators $\mathcal{L}_\mu$ are symmetric and positive definite, and hence invertible, on $\mathcal{V}_\mu$. This allows us to express the resulting preconditioner as [Xu01, Sec. 3.2]

$$\mathcal{B} = \sum_{\mu=1}^{d} \mathcal{L}_\mu^{-1}\,\mathrm{P}_{\mathcal{V}_\mu} = \sum_{\mu=1}^{d} \Big(\mathrm{P}_{\mathcal{V}_\mu}\,\mathcal{L}\,\mathrm{P}_{\mathcal{V}_\mu}\big|_{\mathcal{V}_\mu}\Big)^{-1}\,\mathrm{P}_{\mathcal{V}_\mu}.$$

The action of the preconditioner $\xi = \mathcal{B}\eta$ can thus be computed as $\xi = \sum_{\mu=1}^{d} \xi_\mu$ with

$$\xi_\mu = \Big(\mathrm{P}_{\mathcal{V}_\mu}\,\mathcal{L}\,\mathrm{P}_{\mathcal{V}_\mu}\big|_{\mathcal{V}_\mu}\Big)^{-1}\,\mathrm{P}_{\mathcal{V}_\mu}\,\eta, \qquad \mu = 1, \ldots, d.$$

Local problems. The local equations determining ξ_μ,

$$\mathrm{P}_{\mathcal{V}_\mu}\,\mathcal{L}\,\mathrm{P}_{\mathcal{V}_\mu}\,\xi_\mu = \mathrm{P}_{\mathcal{V}_\mu}\,\eta, \qquad \xi_\mu \in \mathcal{V}_\mu, \qquad \mu = 1, \ldots, d, \tag{6.23}$$

can be solved for all $\xi_\mu \in \mathcal{V}_\mu$ in parallel. To simplify the calculation, we consider the vectorized form of the equation, such that $\mathcal{L}$ is replaced by its matrix representation $L \in \mathbb{R}^{n_1\cdots n_d \times n_1\cdots n_d}$ and the projections $\mathrm{P}_{\mathcal{V}_\mu}$ are also now acting on the vectorized space $\mathbb{R}^{n_1\cdots n_d}$:

$$\mathrm{P}_{\mathcal{V}_\mu}\,L\,\mathrm{P}_{\mathcal{V}_\mu}\,\mathrm{vec}(\xi_\mu) = \mathrm{P}_{\mathcal{V}_\mu}\,\mathrm{vec}(\eta) \qquad \xi_\mu \in \mathcal{V}_\mu, \qquad \mu = 1, \ldots, d,$$

By (4.11), we have $\operatorname{vec}(\xi_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)$ for some gauged $\delta\mathbf{U}_\mu$. Since $\eta = \sum_{\mu=1}^{d} \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu^\eta)$, the right hand side $\mathrm{P}_{\mathcal{V}_\mu}\,\eta$ reduces to $\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu^\eta)$ and we obtain

$$\mathrm{P}_{\mathcal{V}_\mu}\, L\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu^\eta), \qquad \mu = 1, \dots, d,$$

under the additional constraint $(\delta\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}\mathbf{U}_\mu^\mathsf{L} = 0$ when $\mu \neq d$. Now we use the explicit expression (4.13) for the first order variation $\delta\mathbf{U}_\mu^\eta$, $\mu \neq d$, into which we insert $L\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)$:

$$\mathrm{P}_{\mu,\mathsf{L}}^{\perp} \left(I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1}^\mathsf{T}\right) \Big[L\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)\Big]^{<\mu>} \mathbf{X}_{\geq\mu+1}\left(\mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1}\right)^{-1} = (\delta\mathbf{U}_\mu^\eta)^\mathsf{L}, \tag{6.24}$$

where we used the shorthand notation $\mathrm{P}_{\mu,\mathsf{L}}^{\perp} := I_{n_\mu r_{\mu-1}} - \mathbf{U}_\mu^\mathsf{L}(\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}$. For $\mu = d$, the explicit expression (4.14) the first order variation $\delta\mathbf{U}_d^\eta$ leads to

$$\left(I_{n_d} \otimes \mathbf{X}_{\leq d-1}^\mathsf{T}\right) \Big[L\mathbf{X}_{\neq d} \operatorname{vec}(\delta\mathbf{U}_d)\Big]^{<d>} = (\delta\mathbf{U}_d^\eta)^\mathsf{L}. \tag{6.25}$$

The application of the Laplace-like operator L to $\mathbf{X}_{\neq\mu}$ can be decomposed into three parts,

$$L\mathbf{X}_{\neq\mu} = \widetilde{\mathcal{L}}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1} + \mathbf{X}_{\geq\mu+1} \otimes L_\mu \otimes \mathbf{X}_{\leq\mu-1} + \mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \widetilde{\mathcal{L}}_{\leq\mu-1} \tag{6.26}$$

with the reduced leading and trailing terms

$$\begin{aligned}
\widetilde{\mathcal{L}}_{\leq\mu-1} &= \left(\sum_{\nu=1}^{\mu-1} I_{n_{\mu-1}} \otimes \cdots \otimes L_\nu \otimes \cdots \otimes I_{n_1}\right) \mathbf{X}_{\leq\mu-1}, \\
\widetilde{\mathcal{L}}_{\geq\mu+1} &= \left(\sum_{\nu=\mu+1}^{d} I_{n_d} \otimes \cdots \otimes L_\nu \otimes \dots I_{n_{\mu+1}}\right) \mathbf{X}_{\geq\mu+1}.
\end{aligned}$$

Using the recursive relation (4.2), we see that the μth unfolding of any TT tensor $\mathbf{X}$ can be expressed as $\mathbf{X}^{<\mu>} = (I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1})\mathbf{U}_\mu^\mathsf{L}\mathbf{X}_{\geq\mu+1}^\mathsf{T}$. Applying this identity to $[L\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)]^{<\mu>}$ we obtain

$$\begin{aligned}
\left[L\mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)\right]^{<\mu>} = {} & \left(I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1}\right) \delta\mathbf{U}_\mu^\mathsf{L} \widetilde{\mathcal{L}}_{\geq\mu+1}^\mathsf{T} \\
& + \left(L_\mu \otimes \mathbf{X}_{\leq\mu-1} + I_{n_\mu} \otimes \widetilde{\mathcal{L}}_{\leq\mu-1}\right) \delta\mathbf{U}_\mu^\mathsf{L} \mathbf{X}_{\geq\mu+1}^\mathsf{T}.
\end{aligned}$$

Inserting this expression into (6.24) yields for $\mu \neq d$

$$\begin{aligned}
\mathrm{P}_{\mu,\mathsf{L}}^{\perp} \Big[& \delta\mathbf{U}_\mu^\mathsf{L} \widetilde{\mathcal{L}}_{\geq\mu+1}^\mathsf{T} \mathbf{X}_{\geq\mu+1}\left(\mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1}\right)^{-1} \\
& + \left(L_\mu \otimes I_{r_{\mu-1}} + I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1}^\mathsf{T} \widetilde{\mathcal{L}}_{\leq\mu-1}\right) \delta\mathbf{U}_\mu^\mathsf{L}\Big] = (\delta\mathbf{U}_\mu^\eta)^\mathsf{L}.
\end{aligned}$$

After defining the (symmetric positive definite) matrices $\mathcal{L}_{\leq\mu-1} = \mathbf{X}_{\leq\mu-1}^\mathsf{T} \widetilde{\mathcal{L}}_{\leq\mu-1}$ and $\mathcal{L}_{\geq\mu+1} = \mathbf{X}_{\geq\mu+1}^\mathsf{T} \widetilde{\mathcal{L}}_{\geq\mu+1}$, we finally obtain

$$\mathrm{P}_{\mu,\mathsf{L}}^{\perp} \Big[\delta\mathbf{U}_\mu^\mathsf{L} \mathcal{L}_{\geq\mu+1}\left(\mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1}\right)^{-1} + \left(L_\mu \otimes I_{r_{\mu-1}} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1}\right)\delta\mathbf{U}_\mu^\mathsf{L}\Big] = (\delta\mathbf{U}_\mu^\eta)^\mathsf{L}, \tag{6.27}$$

with the gauge condition $(\delta\mathbf{U}_\mu^\mathsf{L})^\mathsf{T}\mathbf{U}_\mu^\mathsf{L} = 0$. For $\mu = d$, there is no gauge condition and (6.25) becomes

$$\delta\mathbf{U}_d^\mathsf{L} + \left(L_d \otimes I_{r_d} + I_{n_d} \otimes \mathcal{L}_{\leq d-1}\right) \delta\mathbf{U}_d^\mathsf{L} = (\delta\mathbf{U}_d^\eta)^\mathsf{L}. \tag{6.28}$$

Efficient solution of local problems. The derivations above have led us to the linear systems (6.27) and (6.28) for determining the local component ξ_μ. While (6.28) is a Sylvester equation and can be solved with standard techniques, more work is needed to address (6.27) efficiently. Since $\mathcal{L}_{\geq\mu+1}$ and $\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1}$ are symmetric positive definite, they admit a generalized eigenvalue decomposition: There is an invertible matrix Q such that $\mathcal{L}_{\geq\mu+1}Q = (\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})Q\Lambda$ with Λ diagonal and $Q^{\mathsf{T}}(\mathbf{X}^{\mathsf{T}}_{\geq\mu+1}\mathbf{X}_{\geq\mu+1})Q = I_{r_\mu}$. This transforms (6.27) into

$$\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\left[\delta\mathbf{U}^{\mathsf{L}}_{\mu}Q^{\mathsf{T}}\Lambda + (L_\mu \otimes I_{r_\mu} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1})\,\delta\mathbf{U}^{\mathsf{L}}_{\mu}Q^{\mathsf{T}}\right] = (\delta\mathbf{U}^{\eta}_{\mu})^{\mathsf{L}}Q^{\mathsf{T}}.$$

Setting $\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu} = \delta\mathbf{U}^{\mathsf{L}}_{\mu}Q^{\mathsf{T}}$ and $(\widetilde{\delta\mathbf{U}}^{\eta}_{\mu})^{\mathsf{L}} = (\delta\mathbf{U}^{\eta}_{\mu})^{\mathsf{L}}Q^{\mathsf{T}}$, we can formulate these equations column-wise:

$$\mathrm{P}^{\perp}_{\mu,\mathsf{L}}\left[\lambda_i I_{r_\mu n_\mu} + L_\mu \otimes I_{r_\mu} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1}\right]\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}(:,i) = (\widetilde{\delta\mathbf{U}}^{\eta}_{\mu})^{\mathsf{L}}(:,i), \tag{6.29}$$

where $\lambda_i = \Lambda(i,i) > 0$. Because Q is invertible, the gauge-conditions on $\delta\mathbf{U}^{\mathsf{L}}_{\mu}$ are equivalent to $(\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu})^{\mathsf{T}}\mathbf{U}^{\mathsf{L}}_{\mu} = 0$. Combined with (6.29), we obtain – similar to (6.19) – the saddle point systems

$$\begin{bmatrix} G_{\mu,i} & \mathbf{U}^{\mathsf{L}}_{\mu} \\ (\mathbf{U}^{\mathsf{L}}_{\mu})^{\mathsf{T}} & 0 \end{bmatrix}\begin{bmatrix} \widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}(:,i) \\ y \end{bmatrix} = \begin{bmatrix} (\widetilde{\delta\mathbf{U}}^{\eta}_{\mu})^{\mathsf{L}}(:,i) \\ 0 \end{bmatrix} \tag{6.30}$$

with the symmetric positive definite matrix

$$G_{\mu,i} = \lambda_i I_{n_\mu} \otimes I_{r_\mu} + L_\mu \otimes I_{r_\mu} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1} \tag{6.31}$$

and the dual variable $y \in \mathbb{R}^{r_\mu}$. The system (6.30) is solved for each column of $\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}$:

$$\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}(:,i) = \Big(I_{n_\mu r_\mu} + G^{-1}_{\mu,i}\,\mathbf{U}^{\mathsf{L}}_{\mu}\,G^{-1}_{S}\,(\mathbf{U}^{\mathsf{L}}_{\mu})^{\mathsf{T}}\Big)G^{-1}_{\mu,i}\,(\widetilde{\delta\mathbf{U}}^{\eta}_{\mu})^{\mathsf{L}}(:,i),$$

using the Schur complement $G_S := -(\mathbf{U}^{\mathsf{L}}_{\mu})^{\mathsf{T}}G^{-1}_{\mu,i}\mathbf{U}^{\mathsf{L}}_{\mu}$. Transforming back then yields $\delta\mathbf{U}^{\mathsf{L}}_{\mu} = \widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}Q^{-\mathsf{T}}$.

Remark 6.2. *Analogous to Remark 6.1, the application of* $G^{-1}_{\mu,i}$ *benefits from the fact that the matrix* $G_{\mu,i}$ *defined in* (6.31) *represents the Sylvester operator*

$$V \mapsto (L_\mu + \lambda_i I_{n_\mu})V + V\mathcal{L}_{\leq\mu-1}.$$

After diagonalization of $\mathcal{L}_{\leq\mu-1}$*, the application of* $G^{-1}_{\mu,i}$ *requires the solution of* r_μ *linear systems with the matrices* $L_\mu + (\lambda_i + \beta)I_{n_\mu}$*, where* β *is an eigenvalue of* $\mathcal{L}_{\leq\mu-1}$*. The Schur complements* $G_S \in \mathbb{R}^{r_\mu \times r_\mu}$ *are constructed explicitly by applying* $G^{-1}_{\mu,i}$ *to the* r_μ *columns of* $\mathbf{U}^{\mathsf{L}}_{\mu}$*.*

Assuming again that solving with the shifted matrices $L_\mu + (\lambda_i + \beta)I_{n_\mu}$ can be performed in $O(n_\mu)$ operations, the construction of the Schur complement G_S needs $O(n_\mu r_\mu^2)$ operations. Repeating this for all r_μ columns of $\widetilde{\delta\mathbf{U}}^{\mathsf{L}}_{\mu}$ and all cores $\mu = 1, \ldots, d-1$ yields a total computational complexity of $O(dnr^3)$ for applying the block-Jacobi preconditioner.

An overlapping block-Jacobi preconditioner

The block diagonal preconditioner discussed above is computationally expensive due to the need for solving the saddle point systems (6.30). To avoid them, we will construct a PSC preconditioner for the subspaces

$$\widehat{\mathcal{V}}_\mu := \left\{ \xi_\mu \;\middle|\; \operatorname{vec}(\xi_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu),\ \delta\mathbf{U}_\mu \in \mathbb{R}^{r_{\mu-1}\times n_\mu \times r_\mu} \right\} = \operatorname{span} \mathbf{X}_{\neq\mu}$$

for $\mu = 1, \dots, d$. Observe that $\mathcal{V}_\mu \subsetneq \widehat{\mathcal{V}}_\mu$ for $\mu \neq d$. Hence, the decomposition $T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}} = \cup_{\mu=1}^d \widehat{\mathcal{V}}_\mu$ is no longer a direct sum as in (4.10). The advantage of $\widehat{\mathcal{V}}_\mu$ over $\mathcal{V}_\mu$, however, is that the orthogonal projector $\mathrm{P}_{\widehat{\mathcal{V}}_\mu}$ onto $\widehat{\mathcal{V}}_\mu$ is considerably easier. In particular, since $\mathbf{X}$ is d-orthogonal, we obtain

$$\mathrm{P}_{\widehat{\mathcal{V}}_\mu} = \mathbf{X}_{\neq\mu}(\mathbf{X}_{\neq\mu}^\mathsf{T}\mathbf{X}_{\neq\mu})^{-1}\mathbf{X}_{\neq\mu}^\mathsf{T} = \mathbf{X}_{\neq\mu}\left[(\mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1})^{-1} \otimes I_{n_\mu} \otimes I_{r_{\mu-1}} \right] \mathbf{X}_{\neq\mu}^\mathsf{T}. \tag{6.32}$$

The PSC preconditioner corresponding to the subspaces $\widehat{\mathcal{V}}_\mu$ is given by

$$\widehat{\mathcal{B}} = \sum_{\mu=1}^d \left(\mathrm{P}_{\widehat{\mathcal{V}}_\mu} \mathcal{L}\, \mathrm{P}_{\widehat{\mathcal{V}}_\mu} \Big|_{\widehat{\mathcal{V}}_\mu} \right)^{-1} \mathrm{P}_{\widehat{\mathcal{V}}_\mu}.$$

The action of the preconditioner $\xi = \widehat{\mathcal{B}}\eta$ can thus be computed as $\xi = \sum_{\mu=1}^d \xi_\mu$ with

$$\mathrm{P}_{\widehat{\mathcal{V}}_\mu} \mathcal{L}\, \mathrm{P}_{\widehat{\mathcal{V}}_\mu} \xi_\mu = \mathrm{P}_{\widehat{\mathcal{V}}_\mu} \eta, \qquad \xi_\mu \in \widehat{\mathcal{V}}_\mu, \qquad \mu = 1, \dots, d. \tag{6.33}$$

Local problems. To solve the local equations (6.33), we proceed as in the previous section, but the resulting equations will be considerably simpler. Again, we consider the vectorized form of the equation, such that $\mathcal{L}$ is replaced by its matrix representation $L \in \mathbb{R}^{n_1\cdots n_d \times n_1 \cdots n_d}$ and the projections $\mathrm{P}_{\widehat{\mathcal{V}}_\mu}$ are also now acting on the vectorized space $\mathbb{R}^{n_1\cdots n_d}$. Let

$$\mathrm{P}_{\widehat{\mathcal{V}}_\mu} \operatorname{vec}(\eta) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\widehat{\delta\mathbf{U}}{}_\mu^\eta)$$

for some $\widehat{\delta\mathbf{U}}{}_\mu^\eta$, which will generally differ from the gauged $\delta\mathbf{U}_\mu^\eta$ parametrization of η. Writing $\operatorname{vec}(\xi_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu)$, we obtain the linear systems

$$\mathrm{P}_{\widehat{\mathcal{V}}_\mu} L \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\widehat{\delta\mathbf{U}}{}_\mu^\eta)$$

for $\mu = 1, \dots, d$. Plugging in (6.32) yields

$$\left[(\mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1})^{-1} \otimes I_{n_\mu} \otimes I_{r_{\mu-1}} \right] \mathbf{X}_{\neq\mu}^\mathsf{T} L \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu) = \operatorname{vec}(\widehat{\delta\mathbf{U}}{}_\mu^\eta). \tag{6.34}$$

Analogous to (6.26), we can write

$$\mathbf{X}_{\neq\mu}^\mathsf{T} L \mathbf{X}_{\neq\mu} = \mathcal{L}_{\geq\mu+1} \otimes I_{n_\mu} \otimes I_{r_{\mu-1}} + \mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1} \otimes L_\mu \otimes I_{r_{\mu-1}} \tag{6.35}$$

$$+ \mathbf{X}_{\geq\mu+1}^\mathsf{T}\mathbf{X}_{\geq\mu+1} \otimes I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1} \tag{6.36}$$

with the left and right parts

$$\mathcal{L}_{\leq\mu-1} = \mathbf{X}_{\leq\mu-1}^{\mathsf{T}} \left(\sum_{\nu=1}^{\mu-1} I_{n_{\mu-1}} \otimes \cdots \otimes L_\nu \otimes \cdots \otimes I_{n_1} \right) \mathbf{X}_{\leq\mu-1},$$

$$\mathcal{L}_{\geq\mu+1} = \mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \left(\sum_{\nu=\mu+1}^{d} I_{n_d} \otimes \cdots \otimes L_\nu \otimes \dots I_{n_{\mu+1}} \right) \mathbf{X}_{\geq\mu+1}.$$

In the left unfolding, we obtain

$$\begin{aligned}(\mathbf{X}_{\neq\mu}^{\mathsf{T}} L \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu))^{\mathsf{L}} = \delta\mathbf{U}_\mu^{\mathsf{L}} \mathcal{L}_{\geq\mu+1} \\ + (L_\mu \otimes I_{r_{\mu-1}} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1}) \delta\mathbf{U}_\mu^{\mathsf{L}} \mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \mathbf{X}_{\geq\mu+1}.\end{aligned}$$

Finally, (6.34) can be written in the left unfolding as

$$\delta\mathbf{U}_\mu^{\mathsf{L}} \mathcal{L}_{\geq\mu+1} (\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \mathbf{X}_{\geq\mu+1})^{-1} + (L_\mu \otimes I_{r_{\mu-1}} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1}) \delta\mathbf{U}_\mu^{\mathsf{L}} = (\widehat{\delta\mathbf{U}}_\mu^{\eta})^{\mathsf{L}}. \quad (6.37)$$

Efficient solution of local problems. The above equations can be directly solved as follows: Using the generalized eigendecomposition of $\mathcal{L}_{\geq\mu+1} Q = (\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \mathbf{X}_{\geq\mu+1}) Q \Lambda$, we can write (6.37) column-wise as

$$G_{\mu,i} \, \widetilde{\delta\mathbf{U}}_\mu^{\mathsf{L}}(:, i) = (\widetilde{\widehat{\delta\mathbf{U}}_\mu^{\eta}})^{\mathsf{L}}(:, i)$$

with the system matrix

$$G_{\mu,i} = \lambda_i I_{n_\mu} \otimes I_{r_\mu} + L_\mu \otimes I_{r_\mu} + I_{n_\mu} \otimes \mathcal{L}_{\leq\mu-1}, \qquad \lambda_i = \Lambda(i, i),$$

and the transformed variables $\widetilde{\delta\mathbf{U}}_\mu^{\mathsf{L}} := \delta\mathbf{U}_\mu^{\mathsf{L}} Q^{\mathsf{T}}$ and $(\widetilde{\widehat{\delta\mathbf{U}}_\mu^{\eta}})^{\mathsf{L}} := (\widehat{\delta\mathbf{U}}_\mu^{\eta})^{\mathsf{L}} Q^{\mathsf{T}}$. Solving with $G_{\mu,i}$ can again be achieved by efficient solvers for Sylvester equations, see Remark 6.2. After forming $\delta\mathbf{U}_\mu^{\mathsf{L}} = \widetilde{\delta\mathbf{U}}_\mu^{\mathsf{L}} Q^{-\mathsf{T}}$ for all μ, we have obtained the solution as an ungauged parametrization:

$$\operatorname{vec}(\xi) = \operatorname{vec}(\widehat{\mathcal{B}}\eta) = \sum_{\mu=1}^{d} \mathbf{X}_{\neq\mu} \operatorname{vec}(\delta\mathbf{U}_\mu).$$

To obtain the gauged parametrization of ξ satisfying (4.12), we can simply apply (4.13) to compute $\mathrm{P}_{T_{\mathbf{X}}\mathcal{M}_{\mathbf{r}}}(\xi)$ and exploit that ξ is a TT tensor (with doubled TT ranks compared to $\mathbf{X}$).

Assuming again that solving with L_μ can be performed in $O(n_\mu)$ operations, we end up with a total computational complexity of $O(dnr^3)$ for applying the overlapping block-Jacobi preconditioner. Although this is the same asymptotic complexity as the non-overlapping scheme, the constant and computational time can be expected to be significantly lower thanks to not having to solve saddle point systems in each step.

Remark 6.3. *When the different parametrization of the tangent space described in Section 4.4.2 is used, the Gram matrix* $\mathbf{X}_{\geq\mu+1}^{\mathsf{T}} \mathbf{X}_{\geq\mu+1}$ *in* (6.27) *and* (6.37) *becomes the identity matrix. This leads to a more stable calculation of the corresponding unknown* $\delta\mathbf{U}_\mu$. *We make use of this transformation in our implementation but omit it here for readability.*

6.2.4. Connection to ALS

The overlapping block-Jacobi preconditioner $\widehat{\mathcal{B}}$ is closely related to the alternating linear scheme (ALS) applied to (6.1). There are, however, crucial differences explaining why $\widehat{\mathcal{B}}$ is significantly cheaper per iteration than ALS.

Using $\operatorname{vec}(\mathbf{X}) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu)$, one so-called micro-step of ALS fixes $\mathbf{X}_{\neq\mu}$ and replaces $\mathbf{U}_\mu$ by the minimizer of (see, e.g., [HRS12a, Alg. 1])

$$\min_{\mathbf{U}_\mu} \frac{1}{2}\langle \mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu), A\mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu)\rangle - \langle \mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu), \operatorname{vec}(\mathbf{F})\rangle.$$

After $\mathbf{U}_\mu$ has been updated, ALS proceeds to the next core until all cores have eventually been updated in a particular order, for example, $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_d$. The solution to the above minimization problem is obtained from solving the ALS subproblem

$$\mathbf{X}_{\neq\mu}^\mathsf{T} A\mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu) = \mathbf{X}_{\neq\mu}^\mathsf{T} \operatorname{vec}(\mathbf{F}).$$

It is well-known that ALS can be seen as a block version of non-linear Gauss–Seidel. The subproblem typically needs to be computed iteratively since the system matrix $\mathbf{X}_{\neq\mu}^\mathsf{T} A\mathbf{X}_{\neq\mu}$ is often unmanageably large.

We refer to a more in-depth discussion of alternating schemes in the context of eigenvalue problems in Chapter 7.

When $\mathbf{X}$ is μ-orthogonal, $\mathbf{X}_{\geq\mu+1}^\mathsf{T} \mathbf{X}_{\geq\mu+1} = I_{r_\mu}$ and the ALS subproblem has the same form as the subproblem (6.34) in the overlapping block-Jacobi preconditioner $\widehat{\mathcal{B}}$. However, there are crucial differences:

- ALS directly optimizes for the cores and as such uses $\mathcal{A}$ in the optimization problem. The approximate Newton method, on the other hand, updates (all) the cores using a search direction obtained from minimizing the quadratic model (6.15). It can therefore use any positive definite approximation of $\mathcal{A}$ to construct this model, which we choose as $\mathcal{L}$. Since (6.34) is the preconditioner for this quadratic model, it uses $\mathcal{L}$ as well.
- ALS updates each core immediately and it is a block version of non-linear Gauss–Seidel for (6.1), whereas $\widehat{\mathcal{B}}$ updates all the cores simultaneously resembling a block version of linear Jacobi.
- Even in the large-scale setting of $n_\mu \gg 10^3$, the subproblems (6.34) can be solved efficiently in closed form as long as $L_\mu + \lambda I_{n_\mu}$ allows for efficient system solves, e.g., for tridiagonal L_μ. This is not possible in ALS where the subproblems have to be formulated with $\mathcal{A}$ and typically need to be solved iteratively using PCG.

Remark 6.4. *Instead of PSC, we experimented with a symmetrized version of a successive subspace correction (SSC) preconditioner, also known as a back and forth ALS sweep. However, the higher computational cost per iteration of SSC was not offset by a possibly improved convergence behavior.*

6.3. Numerical experiments

In this section, we compare the performance of the different preconditioned optimization techniques discussed in this paper for two representative test cases.

We have implemented all algorithms in Matlab. For the TT format, we have made use of the TTeMPS toolbox, see `http://anchp.epfl.ch/TTeMPS`. All numerical experiments and timings are performed on a 12 core Intel Xeon X5675, 3.07 GHz, 192 GiB RAM using Matlab 2014a, running on Linux kernel 3.2.0-0.

To simplify the discussion, we assume throughout this section that the tensor size and ranks are equal along all modes and therefore state them as scalar values: $n = \max_\mu n_\mu$ and $r = \max_\mu r_\mu$.

6.3.1. Test case 1: Newton potential

As a standard example leading to a linear system of the form (6.2), we consider the partial differential equation

$$\begin{aligned} -\Delta u(x) + V(x) &= f(x), \quad & x \in \Omega =]-10, 10[^{\,d}, \\ u(x) &= 0, & x \in \partial\Omega. \end{aligned}$$

with the Laplace operator Δ, the *Newton potential* $V(x) = \|x\|^{-1}$, and the source function $f \colon \mathbb{R}^d \to \mathbb{R}$. Equations of this type are used to describe the energy of a charged particle in an electrostatic potential.

We discretize the domain Ω by a uniform tensor grid with n^d grid points and corresponding mesh width h. Then, by finite difference approximation on this tensor grid, we obtain a tensor equation of the type (6.1), where the linear operator $\mathcal{A}$ is the sum of the d-dimensional Laplace operator as in (6.3) with $L_\mu = \frac{1}{h^2}\operatorname{tridiag}(-1, 2, -1) \in \mathbb{R}^{n\times n}$, and the discretized Newton potential $\mathbf{V}$. To create a low-rank representation of the Newton potential, $V(x)$ is approximated by a rank 10 tensor $\mathbf{V}$ using exponential sums [Hac10]. The application of $\mathcal{A}$ to a tensor $\mathbf{X}$ is given by

$$\mathcal{A}\mathbf{X} = \mathcal{L}\mathbf{X} + \mathbf{V} \star \mathbf{X},$$

where $\star$ denotes the Hadamard product defined in Section 3.3.4 and Section 4.2.4. The application of this operator increases the ranks significantly: If $\mathbf{X}$ has rank r then $\mathcal{A}\mathbf{X}$ has rank $(2 + 10)r = 12r$.

6.3.2. Test case 2: Anisotropic diffusion equation

As a second example, we consider the anisotropic diffusion equation

$$\begin{aligned} -\operatorname{div}(D\nabla u(x)) &= f(x), \quad & x \in \Omega =]-10, 10[^{\,d}, \\ u(x) &= 0, & x \in \partial\Omega, \end{aligned}$$

with a tridiagonal diffusion matrix $D = \operatorname{tridiag}(\alpha, 1, \alpha) \in \mathbb{R}^{d\times d}$. The discretization on a uniform tensor grid with n^d grid points and mesh width h yields a linear equation with system matrix $A = L + V$ consisting of the potential term

$$\begin{aligned} V = {} & I_n \otimes \cdots \otimes I_n \otimes B_2 \otimes 2\alpha B_1 \,+\, I_n \otimes \cdots \otimes I_n \otimes B_3 \otimes 2\alpha B_2 \otimes I_n \\ & + \,\cdots\, + \, B_d \otimes 2\alpha B_{d-1} \otimes I_n \otimes \cdots \otimes I_n, \end{aligned}$$

and the Laplace part L defined as in the previous example. The matrix $B_\mu = \frac{1}{2h}\operatorname{tridiag}(-1, 0, 1) \in \mathbb{R}^{n\times n}$ represents the one-dimensional central finite difference matrix for the first derivative.

As described in Section 4.7, the corresponding linear operator $\mathcal{A}$ acting on $\mathbf{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ can be represented as a TT operator of rank 3, with the cores given by

$$A_1(i_1, j_1) = \begin{bmatrix} L_1(i_1, j_1) & 2\alpha B_1(i_1, j_1) & I_{n_1}(i_1, j_1) \end{bmatrix}, \quad A_d(i_d, j_d) = \begin{bmatrix} I_{n_d}(i_d, j_d) \\ B_d(i_d, j_d) \\ L_d(i_d, j_d) \end{bmatrix},$$

and

$$A_\mu(i_\mu, j_\mu) = \begin{bmatrix} I_{n_\mu}(i_\mu, j_\mu) & 0 & 0 \\ B_\mu(i_\mu, j_\mu) & 0 & 0 \\ L_\mu(i_\mu, j_\mu) & 2\alpha B_\mu(i_\mu, j_\mu) & I_{n_\mu}(i_\mu, j_\mu) \end{bmatrix}, \quad \mu = 2, \ldots, d-1.$$

In the Tucker format, this operator is also of rank 3. Given a tensor $\mathbf{X}$ in the representation (3.4), the result $\mathbf{Y} = \mathcal{A}\mathbf{X}$ is explicitly given by $\mathbf{Y} = \mathbf{G} \times_1 V_1 \times_2 \cdots \times_d V_d$ with

$$V_\mu = \begin{bmatrix} L_\mu U_\mu & U_\mu & B_\mu U_\mu \end{bmatrix} \in \mathbb{R}^{n\times 3r_\mu}$$

and core tensor $\mathbf{G} \in \mathbb{R}^{3r_1\times\cdots\times 3r_d}$ which has a block structure shown in Figure 6.1 for the case $d = 3$.

$$\mathbf{G} =$$

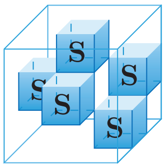


Figure 6.1: Structure of the core tensor $\mathbf{G}$ *for the case* $d = 3$ *resulting from an application of the anisotropic diffusion operator.*

The rank of $\mathcal{A}$ increases linearly with the band width of the diffusion matrix D. For example, a pentadiagonal structure would yield an operator of rank 4. See also [KRS13] for more general bounds in terms of certain properties of D.

6.3.3. Results for the Tucker format

For tensors represented in the Tucker format we want to investigate the convergence of the truncated preconditioned Richardson (6.10) and its Riemannian variant (6.11),

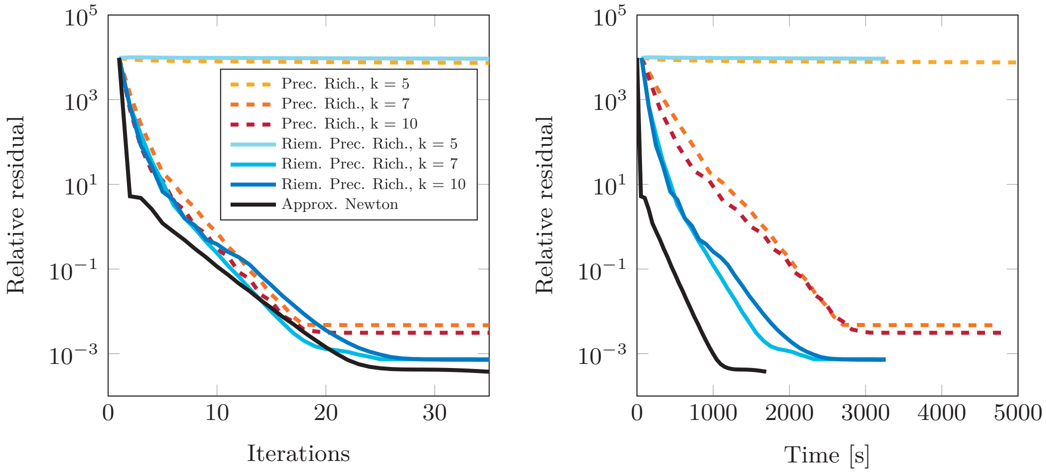


Figure 6.2: Newton potential with $d = 3$. *Comparison of truncated preconditioned Richardson, truncated Riemannian preconditioned Richardson, and the approximate Newton scheme when applied to the Newton potential in the Tucker format. For the Richardson iterations, exponential sum approximations with $k \in \{5, 7, 10\}$ terms are compared.* ***Left:*** *Relative residual as a function of iterations.* ***Right:*** *Relative residual as a function of time*

and compare them to the approximate Newton scheme discussed in Section 6.2.2. Figure 6.2 displays the obtained results for the first test case, the Newton potential, where we set $d = 3$, $n = 100$, and used multilinear ranks $r = 15$. Figure 6.3 displays the results for the second test case, the anisotropic diffusion operator with $\alpha = \frac{1}{4}$, using the same settings. In both cases, the right hand side is given by a random rank-1 Tucker tensor. To create a full space preconditioner for both Richardson approaches, we approximate the inverse Laplacian by an exponential sum of $k \in \{5, 7, 10\}$ terms. It can be clearly seen that the quality of the preconditioner has a strong influence on the convergence. For $k = 5$, convergence is extremely slow. Increasing k yields a drastic improvement on the convergence.

With an accurate preconditioner, the truncated Richardson scheme converges fast with regard to the number of iterations, but suffers from very long computation times due to the exceedingly high intermediate ranks. In comparison, the Riemannian Richardson scheme yields similar convergence speed, but with significantly reduced computation time due to the additional projection into the tangent space. The biggest saving in computational effort comes from relation (6.12) which allows us to avoid having to form the preconditioned residual $\mathcal{P}^{-1}(\mathbf{F} - \mathcal{A}\mathbf{X}_k)$ explicitly, a quantity with very high rank. Note that for both Richardson approaches, it is necessary to round the Euclidean gradient to lower rank using a tolerance of, say, 10^{-5} before applying the preconditioner to avoid excessive intermediate ranks.

The approximate Newton scheme converges equally well as the best Richardson approaches with regard to the number of iterations and does not require setting up a preconditioner. For the first test case, it only needs about half of the time as the best Richardson approach.

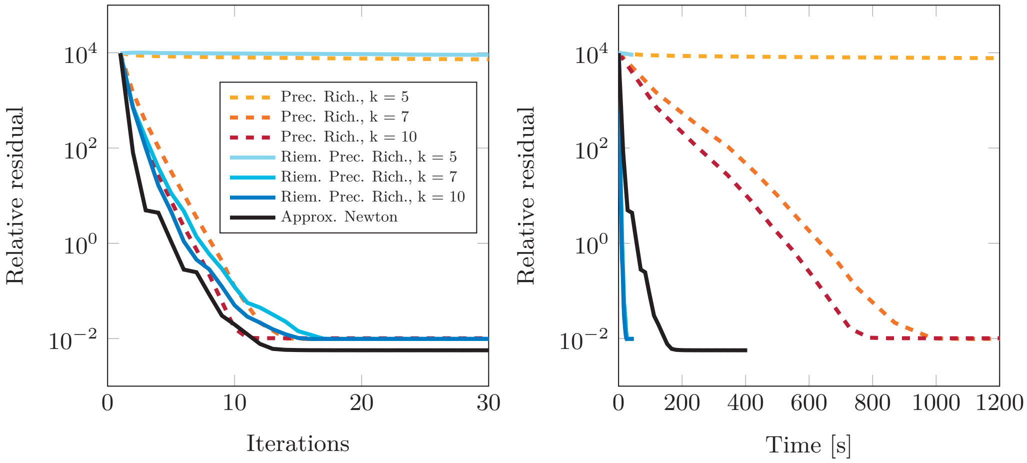


Figure 6.3: Anisotropic diffusion with $d = 3$. *Comparison of truncated Preconditioned Richardson, truncated Riemannian preconditioned Richardson, and the approximate Newton scheme when applied to the Newton potential in the Tucker format. For the Richardson iterations, exponential sum approximations with* $k \in \{5, 7, 10\}$ *terms are compared.* ***Left:*** *Relative residual as a function of iterations.* ***Right:*** *Relative residual as a function of time*

For the second test case, it is significantly slower than Riemannian preconditioned Richardson. Since this operator is of lower rank than the Newton potential, the additional complexity of constructing the approximate Hessian does not pay off in this case.

Quadratic convergence. In Figure 6.4 we investigate the convergence of the approximate Newton scheme when applied to a pure Laplace operator, $A = L$, and to the anisotropic diffusion operator $A = L + V$. In order to have an exact solution of known rank $r = 4$, we construct the right hand side by applying A to a random rank 4 tensor. For the dimension and tensor size we have chosen $d = 3$ and $n = 200$, respectively. By construction, the exact solution lies on the manifold. Hence, if the

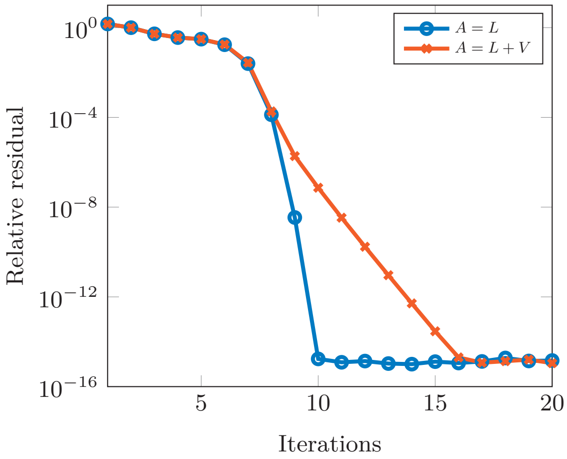


Figure 6.4: Convergence of the approximate Newton method for the zero-residual case when applied to a pure Laplace operator L and to the anisotropic diffusion operator $L + V$.

approximate Newton method converges to this solution, we have zero residual and our Gauss–Newton approximation of (6.14) is an exact second-order model despite only containing the projection of the Euclidean Hessian A. In other words, we expect quadratic convergence when $A = L$ but only linear when $A = L + V$ since our approximate Newton method (6.15) only solves with L. This is indeed confirmed in Figure 6.4.

6.3.4. Results for the TT format

In the TT format, we compare the convergence of our approximate Newton scheme (with the overlapping block-Jacobi preconditioner $\widehat{\mathcal{B}}$) to a standard approach, the alternating linear scheme (ALS). We have chosen $d = 60$, $n = 100$, and a random rank-one right hand sides of norm one. In the first test case, the Newton potential, we have chosen TT ranks $r = 10$ for the approximate solution. The corresponding convergence curves are shown in Figure 6.5. We observe that the approximate Newton scheme needs significantly less time to converge than the ALS scheme. As a reference, we have also included a steepest descent method using the overlapping block-Jacobi scheme directly as a preconditioner for every gradient step instead of using it to solve the approximate Newton equation (6.22). The additional effort of solving the Newton equation approximately clearly pays off.

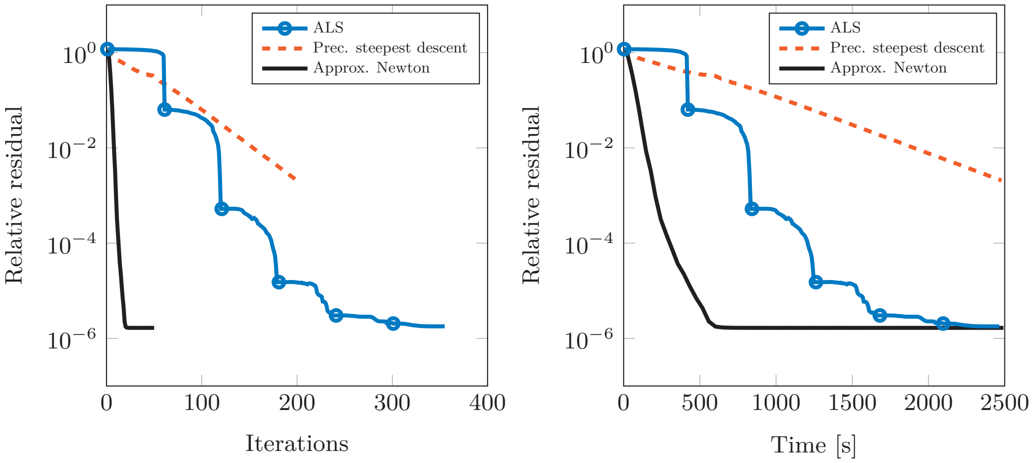


Figure 6.5: Newton potential with $d = 60$. *Convergence of ALS compared to preconditioned steepest descent with overlapping block-Jacobi as preconditioner and the approximate Newton scheme.* ***Left:*** *Relative residual as function of iterations.* ***Right:*** *Relative residual as function of time.*

In Figure 6.6, we show results for the anisotropic diffusion case. To obtain a good accuracy of the solution, we have to choose a relatively high rank of $r = 25$ in this case. Here, the approximate Newton scheme is still faster, especially at the beginning of the iteration, but the final time needed to reach a residual of 10^{-4} is similar to ALS.

Note that in Figures 6.5 and 6.6 the plots with regard to the number of iterations are to be read with care due to the different natures of the algorithms. One ALS

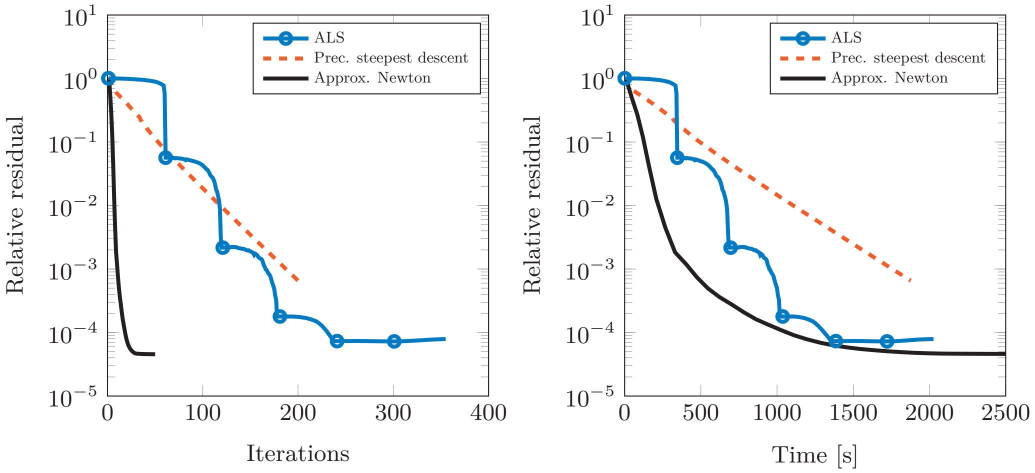


Figure 6.6: Anisotropic diffusion with $d = 60$. *Convergence of ALS compared to preconditioned steepest descent with overlapping block-Jacobi as preconditioner and the approximate Newton scheme.* ***Left:*** *Relative residual as function of iterations.* ***Right:*** *Relative residual as function of time.*

iteration corresponds to the optimization of one core. In the plots, the beginning of each half-sweep of ALS is denoted by a circle. To assessment the performance of both schemes as fairly as possible, we have taken considerable care to provide the same level of optimization to the implementations of both the ALS and the approximate Newton scheme.

Mesh-dependence of the preconditioner. To investigate how the performance of the preconditioner depends on the mesh width of the discretization, we look again at the anisotropic diffusion operator and measure the convergence as the mesh width h and therefore the tensor size $n \in \{60, 120, 180, 240, 360, 420, 480, 540, 600\}$ changes by one order of magnitude. As in the test for quadratic convergence, we construct the right hand side by applying A to a random rank 3 tensor. For the dimension and tensor size we have chosen $d = 3$ and $n = 200$, respectively.

To measure the convergence, we take the number of iterations needed to converge to a relative residual of 10^{-6}. For each tensor size, we perform 30 runs with random starting guesses of rank $r = 3$. The result is shown in Figure 6.7, where circles are drawn for each combination of size n and number of iterations needed. The radius of each circle denotes how many runs have achieved a residual of 10^{-6} for this number of iterations.

On the left plot of Figure 6.7 we see the results of dimension $d = 10$, whereas on the right plot we have $d = 30$. We see that the number of iterations needed to converge changes only mildly as the mesh width varies over one order of magnitude. In addition, the dependence on d is also not very large.

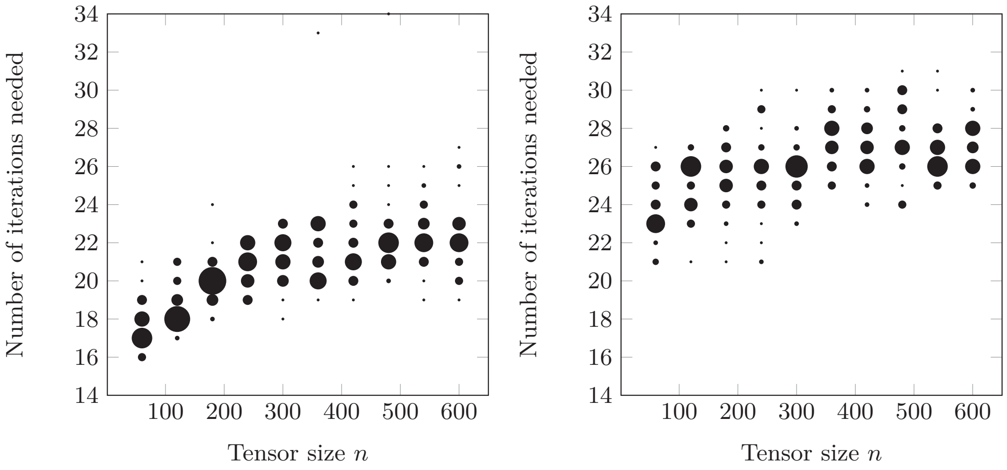


Figure 6.7: Number of iterations that the proposed approximate Newton scheme needs to reach a relative residual of 10^{-6} *for different mesh widths* $h = 1/n$*. The solution has dimension* $d = 10$ *and rank* $r = 3$*. We perform 30 runs for each size. The radii of the circles corresponds to the number of runs achieving this number of iterations.* ***Left:*** *Dimension* $d = 10$*.* ***Right:*** *Dimension* $d = 30$*.*

6.3.5. Rank-adaptivity

Note that in many applications, rank-adaptivity of the algorithm is a desired property. For the Richardson approach, this would result in replacing the fixed-rank truncation with a tolerance-based rounding procedure. In the alternating optimization, this would lead to the DMRG or AMEn algorithms. In the framework of Riemannian optimization, rank-adaptivity can be introduced by successive runs of increasing rank, using the previous solution as a warm start for the next rank. For a recent discussion of this approach, see [UV15]. A basic example of introducing rank-adaptivity to the approximate Newton scheme is shown in Figure 6.8. Starting from ranks $r^{(0)} = 1$, we run the approximate Newton scheme for 10 iterations and use this result to warm start the algorithm with ranks $r^{(i)} = r^{(i-1)} + 5$. At each rank, we perform 10 iterations of the approximate Newton scheme. The result is compared to the convergence of approximate Newton when starting directly with the target rank $r^{(i)}$. We see that the obtained relative residuals match for each of the ranks $r^{(i)}$. Although the adaptive rank scheme is slower for a desired target rank due to the additional intermediate steps, it offers more flexibility when we want to instead prescribe a desired accuracy. For a relative residual of 10^{-3}, the adaptive scheme needs about half the time than using the (too large) rank $r = 36$.

In the case of tensor completion, we have seen in Chapter 5 that rank adaptivity is a crucial ingredient to avoid overfitting and to steer the algorithm into the right direction, see also [Van13, TTW^{+}14, UV15]. For difficult completion problems, careful core-by-core rank increases become necessary. Here, for linear systems, such a core-by-core strategy does not seem to be necessary, as the algorithms will converge even if we directly optimize using rank $r = 36$.

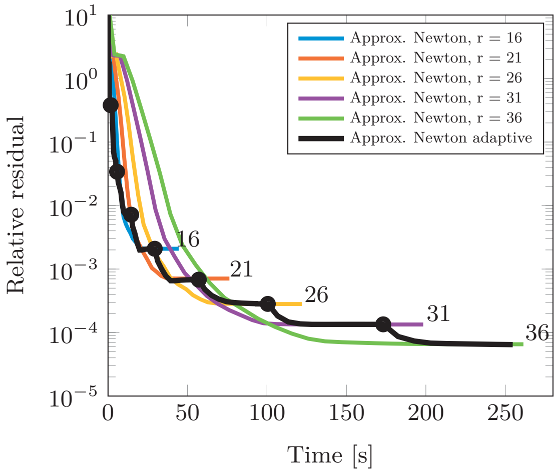


Figure 6.8: Rank-adaptivity for approximate Newton applied to the anisotropic diffusion equation with $n = 100$, $d = 10$. *Starting from rank 1, the rank is increased by 5 after 10 iterations per rank. Each rank increase is denoted by a black circle. The other curves show the convergence when running approximate Newton directly with the target rank.*

6.4. Conclusion

We have investigated different ways of introducing preconditioning into Riemannian gradient descent. As a simple but effective approach, we have seen the Riemannian truncated preconditioned Richardson scheme. Another approach used second-order information by means of approximating the Riemannian Hessian. In the Tucker case, the resulting approximate Newton equation could be solved efficiently in closed form, whereas in the TT case, we have shown that this equation can be solved iteratively in a very efficient way using PCG with an overlapping block-Jacobi preconditioner.

The numerical experiments demonstrate favorable performance of the proposed algorithms when compared to standard non-Riemannian approaches, such as truncated preconditioned Richardson and ALS. The advantages of the approximate Newton scheme become especially pronounced in cases when the linear operator is expensive to apply, e.g., the Newton potential.

Chapter 7

▶ Application 3: Eigenvalue Problems

The computation of the p smallest eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$ for a symmetric matrix $A \in \mathbb{R}^{N\times N}$ can be posed as the minimization problem

$$\min \{ \operatorname{trace}(X^\mathsf{T} A X) \mid X \in \mathbb{R}^{N\times p},\ X^\mathsf{T} X = I_p \}. \tag{7.1}$$

The minimum is given by $\lambda_1 + \lambda_2 + \cdots + \lambda_p$, with a minimizer X^* forming an orthonormal basis for the span of the first p eigenvectors $x_1^*, \dots, x_p^* \in \mathbb{R}^N$. This formulation has been the basis of several established eigenvalue solvers, see [AMSVD02, EAS99, ST00, SW82] for examples.

The required computational cost of traditional methods for solving an $N \times N$ eigenvalue problem scales at least linearly with N, which makes them unfeasible for treating very large values of N. The common approach is to exploit an underlying tensor product structure

$$\mathbb{R}^N \cong \mathbb{R}^{n_1\times n_2\times\cdots\times n_d},$$

just as in Chapter 6. Here, we assume that the matrix X^* containing the eigenvectors can be well approximated in the TT format, to which we restrict the admissible set of the optimization problem (7.1). The resulting (highly nonlinear) optimization can be addressed by alternating block optimization techniques. This approach has been successfully used in the context of DMRG for simulating strongly correlated spin systems, see, e.g., the survey [Sch11] and the references therein. In the numerical analysis community, related ideas have been considered in [DKOS13, KO10a, KT11b]. We mention in passing that another tensor-based approach to eigenvalue computation consists of combining a standard iterative solver (such as the Lanczos or LOBPCG methods) with repeated low-rank compression [HKST12, HW12, KO10b, KT11b, Leb11]; see [GKT13, Sec. 3.1] for further references. Low-rank matrix and tensor techniques have also been used successfully for performing large-scale electronic structure calculations in computational quantum chemistry, see [CEM12, HFY$^+$04, KKF11, Kho14] for examples.

In this work, we propose a novel low-rank tensor method that allows for the computation of several smallest eigenvalues, offers rank adaptivity, and the possibility to incorporate preconditioners. For this purpose, we combine and extend several existing techniques. First, the block TT format proposed by Dolgov et al. [DKOS13], which has previously been considered in the context of Wilson's numerical renormalization group [PV2012, WVS$^+$09], is used for representing an orthonormal basis of p vectors. The approach for solving (7.1) considered in [DKOS13, PV2012] consists of cyclically optimizing each core of the block TT format, after shifting and merging the index enumerating the eigenvalues to this core. This makes the method different from alternating block optimization approaches, such as ALS [HRS12a, KT11b], and allows for rank adaptivity if $p > 1$. Second, we combine this idea with locally enriching each core based on projected gradient information. Such an enrichment was initially proposed by White [Whi05] for eigenvalue problems, and extended to symmetric as well as nonsymmetric linear systems by Dolgov and Savostyanov [DS13a, DS13b, DS14]. Enrichment allows for rank adaptivity even in the case $p = 1$. Preconditioners, which arise naturally in the context of PDE eigenvalue problems, can be easily incorporated by using the preconditioned gradient instead of the gradient.

The content presented in this chapter is based on the published article [KSU14].

7.1. Extraction of two neighboring cores

As we have seen in (4.4), the vectorization vec($\mathbf{X}$) of a TT tensor $\mathbf{X}$ allows us to isolate its μth core,

$$\operatorname{vec}(\mathbf{X}) = \mathbf{X}_{\neq\mu} \operatorname{vec}(\mathbf{U}_\mu).$$

In this chapter we extend this to extract two neighboring cores. Using the recursive relation (4.2) for $\mathbf{X}_{\geq\mu+1}$ we get for every $\mu = 1, \dots, d-1$

$$\operatorname{vec}(\mathbf{X}) = \mathbf{X}_{\neq\mu,\mu+1} \cdot \operatorname{vec}(\mathbf{U}_\mu^{\mathsf{L}} \mathbf{U}_{\mu+1}^{\mathsf{R}}), \tag{7.2}$$

with

$$\mathbf{X}_{\neq\mu,\mu+1} = \mathbf{X}_{\geq\mu+2} \otimes I_{n_{\mu+1}} \otimes I_{n_\mu} \otimes \mathbf{X}_{\leq\mu-1}. \tag{7.3}$$

In terms of tensor spaces, this means

$$x \in \operatorname{range}(\mathbf{X}_{\neq\mu,\mu+1}) = \operatorname{range}(\mathbf{X}_{\geq\mu+2}) \otimes \mathbb{R}^{n_{\mu+1}} \otimes \mathbb{R}^{n_\mu} \otimes \operatorname{range}(\mathbf{X}_{\leq\mu-1}). \tag{7.4}$$

For later reference we note that

$$\mathbf{X}_{\neq\mu,\mu+1}(I_{r_{\mu+1}n_{\mu+1}} \otimes \mathbf{U}_\mu^{\mathsf{L}}) = \mathbf{X}_{\neq\mu+1}. \tag{7.5}$$

In this chapter, we usually assume that $\mathbf{X}$ is μ-orthogonal when focusing on the μth core.

7.2. The block TT format for a collection of vectors

When computing $p > 1$ eigenvectors, we need to work with p vectors $x_1, x_2, \dots, x_p \in \mathbb{R}^{n_1 n_2 \cdots n_d}$ simultaneously. Instead of representing each vector individually in the TT format, we will represent them jointly in a block TT format, following [DKOS13]. For this purpose, we fix an arbitrary position μ and use the representation

$$\begin{gathered} x_\alpha(i_1, i_2, \dots, i_d) = U_1(i_1) \cdots U_{\mu-1}(i_{\mu-1}) U_{\mu,\alpha}(i_\mu) U_{\mu+1}(i_{\mu+1}) \cdots U_d(i_d), \\ 1 \leq \alpha \leq p. \end{gathered} \tag{7.6}$$

To highlight the position of the index α, we call this format the *block-μ TT format*, see Fig. 7.1 for an illustration. It coincides with the usual TT format except for

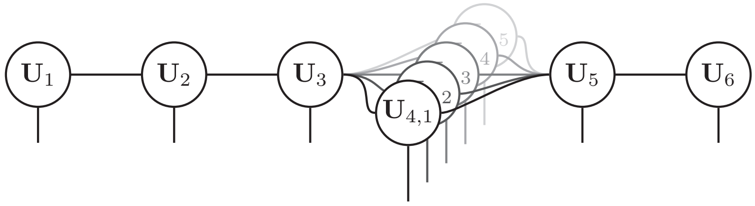


Figure 7.1: Graphical representation of a block-4 TT tensor of order 6.

the core $\mathbf{U}_{\mu,\alpha}$, which is different for every α. None of the other cores depends on α, which is the crucial difference between the block-μ TT format and a collection of p TT formats.

Writing $U_\mu(\alpha, i_\mu)$ instead of $U_{\mu,\alpha}(i_\mu)$ in (7.6), it becomes clear that the block-μ TT format can be obtained by applying the TT format to the vectorization of the matrix

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix} \in \mathbb{R}^{n_1 n_2 \cdots n_d \times p}$$

into a vector of length $n_1 \cdots n_{\mu-1}(pn_\mu)n_{\mu+1} \cdots n_d$, where we treat the double index (α, i_μ) at position μ as a single index j_μ running from 1 to $n_\mu p$ in appropriate order. In terms of tensor spaces, (7.6) states that

$$\begin{aligned} \operatorname{range}(X) &= \operatorname{span}\{x_1, x_2, \dots, x_p\} \\ &\subseteq \operatorname{range}(\mathbf{X}_{\neq\mu}) = \operatorname{range}(\mathbf{X}_{\leq\mu-1}) \otimes \mathbb{R}^{n_\mu} \otimes \operatorname{range}(\mathbf{X}_{\geq\mu+1}). \end{aligned} \tag{7.7}$$

The position of the index α can be moved by separating it from the core $\mathbf{U}_\mu$ and attaching it to the $(\mu+1)$th core, see Algorithm 7.1 for details. The minimal-rank decomposition (7.8) needed in Algorithm 7.1 can be performed by, e.g., a QR decomposition with column pivoting or an SVD. In either case, the updated left unfolding $\mathbf{U}_\mu^{\mathsf{L}} \leftarrow Q$ has orthonormal columns and thus the resulting block-$(\mu+1)$ tensor $\mathbf{X}$ is now $(\mu+1)$-orthogonalized.

Algorithm 7.1 Conversion from block-μ TT format to block-$(\mu+1)$ TT format

```
Input: $\mathbf{X}$ in block-$\mu$ TT format, $\mu < d$.
Output: New cores $\mathbf{U}_\mu$ and $\mathbf{U}_{\mu+1,\alpha}$ representing $\mathbf{X}$ in block-$(\mu+1)$ TT format.
1: Perform a minimal-rank decomposition
       $\begin{bmatrix} \mathbf{U}_{\mu,1}^{\mathsf{L}} & \mathbf{U}_{\mu,2}^{\mathsf{L}} & \dots & \mathbf{U}_{\mu,p}^{\mathsf{L}} \end{bmatrix} = Q \begin{bmatrix} P_1 & P_2 & \dots & P_p \end{bmatrix},$
       $Q \in \mathbb{R}^{r_{\mu-1}n_\mu \times s},\ P_\alpha \in \mathbb{R}^{s \times r_\mu}.$        (7.8)
2: Update cores:
       $\mathbf{U}_\mu^{\mathsf{L}} \leftarrow Q, \quad \mathbf{U}_{\mu+1,\alpha}^{\mathsf{R}} \leftarrow P_\alpha \mathbf{U}_{\mu+1}^{\mathsf{R}}, \quad \alpha = 1, 2, \dots, p.$
3: Update rank:    $r_\mu \leftarrow s.$
```

Algorithm 7.2 Conversion from block-μ format to block-$(\mu-1)$ format

```
Input: $\mathbf{X}$ in block-$\mu$ TT format, $\mu > 1$.
Output: New cores $\mathbf{U}_\mu$ and $\mathbf{U}_{\mu-1,\alpha}$ representing $\mathbf{X}$ in block-$(\mu-1)$ TT format.
1: Perform a minimal-rank decomposition
     $\begin{bmatrix} \mathbf{U}^{\mathsf{R}}_{\mu,1} \\ \mathbf{U}^{\mathsf{R}}_{\mu,2} \\ \vdots \\ \mathbf{U}^{\mathsf{R}}_{\mu,p} \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_p \end{bmatrix} P, \quad Q_\alpha \in \mathbb{R}^{r_{\mu-1} \times s}, \quad P \in \mathbb{R}^{s \times n_\mu r_\mu}.$
2: Update cores:
     $\mathbf{U}^{\mathsf{R}}_\mu = P, \quad \mathbf{U}^{\mathsf{L}}_{\mu-1,\alpha} = \mathbf{U}^{\mathsf{L}}_{\mu-1} Q_\alpha, \quad \alpha = 1, 2, \dots, p.$
3: Update rank:   $r_{\mu-1} \leftarrow s$.
```

7.3. Trace minimization in the block TT format

We return to the trace minimization problem (7.1), with the additional constraint that $X = [x_1, x_2, \dots, x_p]$ is represented in the block-1 TT format.

7.3.1. Alternating optimization

Our starting point is the recently proposed alternating algorithm of Pižorn and Verstraete [PV2012], called *variational numerical renormalization group*, for excited states calculations in quantum many-body systems. In the numerical analysis community, a variant of this algorithm has been proposed by Dolgov et al. [DKOS13].

The algorithm proceeds by alternatingly optimizing each core of the block TT format, similar to block coordinate descent methods in optimization. Given a μ-orthogonalized tensor $\mathbf{X}$ in the block-μ TT format, let us consider the optimization of its μth core. We have

$$X = \mathbf{X}_{\neq\mu} V$$

with the matrix

$$V = \begin{bmatrix} \operatorname{vec}(\mathbf{U}_{\mu,1}) & \operatorname{vec}(\mathbf{U}_{\mu,2}) & \dots & \operatorname{vec}(\mathbf{U}_{\mu,p}) \end{bmatrix} \in \mathbb{R}^{r_{\mu-1} n_\mu r_\mu \times p}. \tag{7.9}$$

Since $\mathbf{X}_{\neq\mu}$ has orthonormal columns, the matrix X has orthonormal columns if and only if $V^{\mathsf{T}} V = I_p$. Therefore, optimizing (7.1) with respect to the μth core becomes the trace minimization problem

$$\min\{\operatorname{trace}(V^{\mathsf{T}} \mathcal{A}_\mu V) \mid V \in \mathbb{R}^{r_{\mu-1} n_\mu r_\mu \times p},\ V^{\mathsf{T}} V = I_p\}, \tag{7.10}$$

with the *reduced matrix*

$$\mathcal{A}_\mu = \mathbf{X}_{\neq\mu}^{\mathsf{T}} A \mathbf{X}_{\neq\mu}. \tag{7.11}$$

Algorithm 7.3 One sweep of ALS for solving the trace minimization problem (7.1)

```
Input: Starting guess: right-orthogonal $\mathbf{X}$ in block-1 TT format
for $\mu = 1, 2, \dots, d-1$ do
    Replace core $\mathbf{U}_{\mu,\alpha}$ by the reshaped solution $V$ of (7.10).
    Apply Algorithm 7.1 to move position of index $\alpha$ to $\mu + 1$, such that the updated core $\mathbf{U}^{\mathsf{L}}_{\mu}$ is left-orthonormal.
end for
for $\mu = d, d-1, \dots, 2$ do
    Replace core $\mathbf{U}_{\mu,\alpha}$ by the reshaped solution $V$ of (7.10).
    Apply Algorithm 7.2 to move position of index $\alpha$ to $\mu - 1$, such that the updated core $\mathbf{U}^{\mathsf{R}}_{\mu}$ is right-orthonormal.
end for
```

In terms of tensor spaces, it follows from (4.3) and (7.7) that this problem is equivalent to

$$\min \{ \operatorname{trace}(X^{\mathsf{T}} A X) \mid \operatorname{range}(X) \subseteq \operatorname{range}(\mathbf{X}_{\neq \mu}),\ X^{\mathsf{T}} X = I_p \}. \tag{7.12}$$

After the core $\mathbf{U}_{\mu,\alpha}$ has been replaced by the reshaped solution V of (7.10), the index α is moved to the right using Algorithm 7.1. In this manner one proceeds from left to right until $\mu = d$. The described procedure constitutes one *half sweep* of *ALS*. When the dth core is reached, one continues by a half sweep from right to left in an analogous manner. Two subsequent half sweeps are called a *full sweep*.

Algorithm 7.3 describes one sweep of ALS. The solution of the reduced problem (7.10) can be done using LOBPCG [Kny01], which benefits from the use of a preconditioner for the reduced matrix A_μ. This will be discussed in Section 7.3.4.

7.3.2. Core enrichment based on gradient information

The convergence of ALS can be improved by enriching the cores with information that aims to improve the quality of the current approximation. Such a core enrichment has been proposed by White [Whi05] in the context of one-site DMRG algorithms, to overcome the unfavorable scaling of *exact* two-site DMRG algorithms. Dolgov and Savostyanov [DS13a, DS13b] have significantly extended this idea to ALS-type algorithms for solving symmetric and nonsymmetric linear systems (see also [DS15] for a discussion of the similarities and differences of both approaches). In this section, we show how core enrichment can be combined with Algorithm 7.3. For this purpose, it is convenient to first discuss a variant that enriches all cores simultaneously. Although such a *global correction* is conceptually simple, it bears the disadvantage of enlarging all TT ranks at the same time, which makes all subsequent operations of ALS significantly more expensive. We therefore develop a second variant, which only enriches the cores that are next to the core currently optimized in a step of ALS. Such a *local correction* not only decreases the computational effort, but it also allows to adjust the correction during an ALS sweep.

Global correction

This approach was termed *ALS(t+z)* in [DS13a, DS13b] and later *ALS(SD)* in [DS14]. Assume that $\mathbf{X}$ is our current approximation in block-1 TT format and suppose we add an error correction $\mathbf{R}$, also in block-1 TT format. Then $\widetilde{\mathbf{X}} = \mathbf{X} + \mathbf{R}$ can again be represented in block-1 TT format using the TT addition procedure, see Section 4.2.2, where the resulting TT rank is the sum of the TT ranks of $\mathbf{X}$ and $\mathbf{R}$.

The ALS(SD) algorithm now proceeds by applying a sweep of ALS (Algorithm 7.3) to $\widetilde{\mathbf{X}}$. According to (7.12), the first step of ALS determines the best p-dimensional subspace for the trace minimization problem constrained to the range of $\widetilde{\mathbf{X}}_{\neq 1}$. The following proposition implies that this constraint includes all linear combinations from the ranges of $\mathbf{X}$ and $\mathbf{R}$.

Proposition 7.1. *Consider a matrix $\widetilde{\mathbf{X}} = \mathbf{X} + \mathbf{R}$ in block-1 TT format with matrix representations X and R, respectively. Then the matrix $\widetilde{\mathbf{X}}_{\neq 1} = \widetilde{\mathbf{X}}_{\geq 2} \otimes I_{n_1}$ satisfies* $\mathrm{range}(\widetilde{\mathbf{X}}_{\neq 1}) \supseteq \mathrm{range}(X) + \mathrm{range}(R)$*, i.e.,*

$$\mathrm{range}(\widetilde{\mathbf{X}}_{\neq 1}) \supseteq \{XS + RT \mid S, T \in \mathbb{R}^{p \times p}\}.$$

Proof. Because of the structure of the TT cores of $\widetilde{\mathbf{X}}$, we have $\widetilde{\mathbf{X}}_{\geq 2} = \begin{bmatrix} \mathbf{X}_{\geq 2} & \mathbf{R}_{\geq 2} \end{bmatrix}$. Hence,

$$\begin{aligned} \mathrm{range}(X) + \mathrm{range}(R) &\subseteq \mathrm{range}(\mathbf{X}_{\neq 1}) + \mathrm{range}(\mathbf{R}_{\neq 1}) \\ &= (\mathrm{range}(\mathbf{X}_{\geq 2}) \otimes \mathbb{R}^{n_1}) + (\mathrm{range}(\mathbf{R}_{\geq 2}) \otimes \mathbb{R}^{n_1}) \\ &= (\mathrm{range}(\mathbf{X}_{\geq 2}) + \mathrm{range}(\mathbf{R}_{\geq 2})) \otimes \mathbb{R}^{n_1} \\ &= \mathrm{range}(\widetilde{\mathbf{X}}_{\geq 2}) \otimes \mathbb{R}^{n_1} = \mathrm{range}(\widetilde{\mathbf{X}}_{\neq 1}), \end{aligned}$$

where the first inclusion is due to (7.7). ☐

Proposition 7.1 implies that the result from the first step of ALS applied to $\widetilde{\mathbf{X}}$ is at least as good as selecting the best p linear combinations from $\mathrm{range}(X)$ and $\mathrm{range}(R)$. This point of view allows us to interpret $\mathbf{R}$ as a *subspace correction.* In particular, it is important to emphasize that the particular linear combination $\mathbf{X} + \mathbf{R}$ is only used to setup the cores; it does *not* correspond to the correction of $\mathbf{X}$ that is actually used.

Similar statements can be made when $\mathbf{X}$ and $\mathbf{R}$ are represented in the block-μ TT format for $\mu \neq 1$. In particular, this gives the possibility to inject an additional subspace acceleration before every step of Algorithm 7.3. However, the incurred growth of the TT ranks makes such an approach little attractive. In the ALS(SD) algorithm a subspace correction is therefore only added before every left-to-right and before every right-to-left half-sweep.

It remains to discuss the choice of $\mathbf{R}$. A natural candidate is the negative block residual, $R = -(AX - X\Lambda)$ with $\Lambda = X^{\mathsf{T}}AX$. This choice can be motivated as follows. As $\mathbf{A}$ is symmetric, the gradient of $\frac{1}{2}\,\mathrm{trace}(\mathbf{X}^{\mathsf{T}}\mathbf{A}\mathbf{X})$ as a function of $\mathbf{X}$ is given by $\mathbf{A}\mathbf{X}$. The optimization task (7.1) is posed on the Stiefel manifold $\{\mathbf{X} \in \mathbb{R}^{N \times p} \mid \mathbf{X}^{\mathsf{T}}\mathbf{X} = I_p\}$. Therefore, as we have seen in (2.2), the direction of steepest descent is given by the Riemannian gradient. For the case of the Stiefel manifold, we obtain [AMS08, Ex. 3.6.2]

$$(I - XX^{\mathsf{T}})(-AX) + X\,\mathrm{skew}(X^{\mathsf{T}}(-AX)) = -(AX - X\Lambda), \tag{7.13}$$

where have used that skew$(-\Lambda)$, the skew-symmetric part of $-\Lambda$, is zero.

The convergence of gradient methods for solving eigenvalue problems critically depends on certain eigenvalue gaps *relative* to the width of the spectrum of A and it deteriorates when these gaps approach zero. The method of steepest descent applied to (7.1) is no exception [Ney02]. For discretized PDE eigenvalue problems, the unboundedness of the underlying operator implies that the width of the spectrum becomes wider (therefore, the relative eigenvalue gaps become smaller) as the discretization gets refined. This effect hampers convergence, but it can be avoided by using preconditioned gradients instead. In the case of (7.1), the negative preconditioned gradient is given by the negative[1] *preconditioned block residual*

$$R = -B^{-1}(AX - X\Lambda) \tag{7.14}$$

for a preconditioner B of A; see also [BPK96]. If B is spectrally equivalent to A then the convergence rate of the resulting preconditioned gradient method does not deteriorate as the discretization gets refined [Ney02]. To make use of (7.14) in the context of low-rank tensor methods, B^{-1} needs to be represented in the operator TT format. As usual, both the computation of R (and thus of $\mathbf{R}$) and its addition to $\mathbf{X}$ will only be performed approximately, that is, low-rank truncation is applied to limit the rank growth.

Local corrections

As discussed above, the simultaneous increase of all TT ranks renders a global correction too expensive to be applied before *every* step of ALS. To address this issue, an approach based on local corrections has been proposed for linear systems in [DS14]. In the following, we will extend this approach to eigenvalue problems.

Let us consider the μth step of a left-to-right ALS sweep and suppose that the μth core has been optimized, but the position of the index α has not yet been moved to $\mu + 1$. We then augment the neighboring cores with the corresponding cores of a correction $\mathbf{R}$:

$$\widetilde{\mathbf{U}}^{\mathsf{L}}_{\mu,\alpha} = \begin{bmatrix} \mathbf{U}^{\mathsf{L}}_{\mu,\alpha} & \mathbf{R}^{\mathsf{L}}_{\mu,\alpha} \end{bmatrix}, \quad \widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1} = \begin{bmatrix} \mathbf{U}^{\mathsf{R}}_{\mu+1} \\ \mathbf{R}^{\mathsf{R}}_{\mu+1} \end{bmatrix}. \tag{7.15}$$

In particular, only the rank r_μ is changed. Similarly we proceed in a right-to-left sweep, see Algorithm 7.4 for a complete description and Figure 7.2 for an illustration. Since the analogous algorithm for linear systems was coined AMEn (Alternating Minimal Energy method) [DS14], we will call our algorithm *EVAMEn* (EigenValue AMEn).

---

[1] The sign and norm of the $\mathbf{R}$ are irrelevant for the subsequent optimization steps. We use negative residuals in the presentation, as it allows for the interpretation of the enrichment as a correction step in the direction of steepest descent.

Algorithm 7.4 One full sweep of EVAMEn for solving the trace minimization problem (7.1)

```
Input: Starting guess: right-orthogonal $\mathbf{X}$ in block-1 TT format
for $\mu = 1, 2, \dots, d-1$ do
    Replace core $\mathbf{U}_{\mu,\alpha}$ by the reshaped solution $V$ of (7.10).
    Augment cores $\mathbf{U}_{\mu,\alpha}$ and $\mathbf{U}_{\mu+1}$ with cores $\mathbf{R}_{\mu,\alpha}$ and $\mathbf{R}_{\mu+1}$ of compatible size, according to (7.15).
    Apply Algorithm 7.1 to move position of index $\alpha$ to $\mu+1$, such that the updated core $\mathbf{U}_{\mu}^{\mathsf{L}}$ is left-orthonormal.
end for
for $\mu = d, d-1, \dots, 2$ do
    Replace core $\mathbf{U}_{\mu,\alpha}$ by the reshaped solution $V$ of (7.10).
    Augment cores $\mathbf{U}_{\mu-1}$ and $\mathbf{U}_{\mu,\alpha}$ with cores $\mathbf{R}_{\mu-1}$ and $\mathbf{R}_{\mu,\alpha}$ of compatible size, according to (7.15).
    Apply Algorithm 7.1 to move position of index $\alpha$ to $\mu-1$, such that the updated core $\mathbf{U}_{\mu}^{\mathsf{R}}$ is right-orthonormal.
end for
```

We now turn to an important interpretation of Algorithm 7.4 as an alternating local subspace correction method for the famous *two-site DMRG* algorithm [Whi92], which we therefore shortly recall.

DMRG for trace minimization Assume that $\mathbf{X}$ is either in block-μ or in block-$(\mu+1)$ TT format. One step of the DMRG algorithm applied to (7.1) begins with merging two neighboring cores, at positions μ and $\mu+1$, and then solves the resulting optimization problem for the merged *supercore* $\mathbf{W}$ of size $r_{\mu-1} \times n_\mu \times n_{\mu+1} \times r_{\mu+1}$ and its matrix representation $W \in \mathbb{R}^{r_{\mu-1}n_\mu n_{\mu+1}r_{\mu+1} \times p}$:

$$\min\{\operatorname{trace}(W^{\mathsf{T}} A_{\mu,\mu+1} W) \mid W \in \mathbb{R}^{r_{\mu-1}n_\mu n_{\mu+1}r_{\mu+1} \times p},\ W^{\mathsf{T}} W = I_p\} \tag{7.16}$$

where

$$\mathcal{A}_{\mu,\mu+1} = \mathbf{X}_{\neq\mu,\mu+1}^{\mathsf{T}} A \mathbf{X}_{\neq\mu,\mu+1},$$

and the matrix $\mathbf{X}_{\neq\mu,\mu+1}$ defined in (7.3) is assumed to have orthonormal columns. By (7.4), the DMRG subproblem (7.16) is equivalent to minimizing the trace of $X^{\mathsf{T}} A X$ under the constraints $X^{\mathsf{T}} X = I_p$ and

$$\operatorname{range}(X) \subseteq \operatorname{range}(\mathbf{X}_{\geq\mu+2}) \otimes \mathbb{R}^{n_{\mu+1}} \otimes \mathbb{R}^{n_\mu} \otimes \operatorname{range}(\mathbf{X}_{\leq\mu-1}) = \operatorname{range}(\mathbf{X}_{\neq\mu,\mu+1}).$$

Once (7.16) is solved, a *minimal* subspace $\operatorname{range}(\mathbf{X}_{\leq\mu}) \subseteq \mathbb{R}^{n_\mu} \otimes \operatorname{range}(\mathbf{X}_{\leq\mu-1})$ is determined such that $\operatorname{range}(X) \subseteq \operatorname{range}(\mathbf{X}_{\geq\mu+2}) \otimes \mathbb{R}^{n_{\mu+1}} \otimes \operatorname{range}(\mathbf{X}_{\leq\mu})$ holds. Computationally, this means that the solution $\mathbf{W}$ of (7.16) is reshaped into an $(r_{\mu-1}n_\mu) \times (n_{\mu+1}r_{\mu+1}p)$ matrix W and a minimal-rank decomposition is applied to yield new cores $\mathbf{U}_\mu$, $\mathbf{U}_{\mu+1,\alpha}$ such that

$$W = \left[\operatorname{vec}(\mathbf{U}_\mu^{\mathsf{L}} \mathbf{U}_{\mu+1,1}^{\mathsf{R}}) \quad \operatorname{vec}(\mathbf{U}_\mu^{\mathsf{L}} \mathbf{U}_{\mu+1,2}^{\mathsf{R}}) \quad \dots \quad \operatorname{vec}(\mathbf{U}_\mu^{\mathsf{L}} \mathbf{U}_{\mu+1,p}^{\mathsf{R}})\right].$$

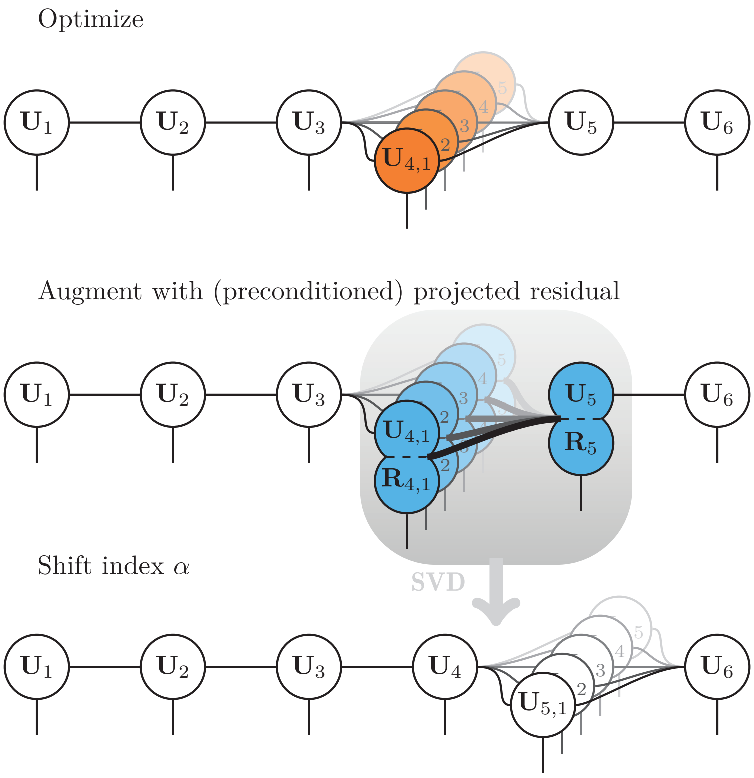


Figure 7.2: One microiteration in a left to right sweep of the EVAMEn algorithm (lines 2–4 of Algorithm 7.4).

Since this decomposition automatically chooses a new, currently optimal rank r_μ, the DMRG is rank adaptive, even for $p = 1$. Note that the above decomposition of $\mathbf{W}$ also moves the index α to the next position. In practice, the decomposition is only performed approximately, using a truncated SVD, to avoid that the updated rank r_μ becomes too large.

One full sweep of DMRG consists of processing all possible supercores in the manner described above, first from left-to-right and second from right-to-left. The global and local convergence properties of the DMRG algorithm are not well understood. However, there are cases for which DMRG needs very few sweeps to converge to high accuracy, see, e.g., [KT11b]. It is, however, quite costly, since the local two-core problem matrix $\mathcal{A}_{\mu,\mu+1}$ is of size $r_{\mu-1}n_\mu n_{\mu+1}r_\mu \times r_{\mu-1}n_\mu n_{\mu+1}r_\mu$.

Relation of Algorithm 7.4 to DMRG Following [DS14], we now explain how Algorithm 7.4, which only operates on individual cores, can be viewed as producing approximate solutions to the DMRG subproblem (7.16). For this purpose, we consider the μth step of Algorithm 7.4 and interpret the supercore

$$W^0 = \left[\operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu,1}\mathbf{U}^{\mathsf{R}}_{\mu+1}) \quad \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu,2}\mathbf{U}^{\mathsf{R}}_{\mu+1}) \quad \dots \quad \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu,p}\mathbf{U}^{\mathsf{R}}_{\mu+1})\right], \tag{7.17}$$

where $\mathbf{U}_{\mu,\alpha}$ and $\mathbf{U}_{\mu+1}$ are the cores just before the augmentation in line 3, as an initial guess for (7.16). Given the cores $\mathbf{R}_{\mu,\alpha}$ and $\mathbf{R}_{\mu+1}$ of the correction, we define[2]

$$\mathbf{R}_{\mu,\mu+1} = \begin{bmatrix}\operatorname{vec}(\mathbf{R}^{\mathsf{L}}_{\mu,1}\mathbf{R}^{\mathsf{R}}_{\mu+1}) & \operatorname{vec}(\mathbf{R}^{\mathsf{L}}_{\mu,2}\mathbf{R}^{\mathsf{R}}_{\mu+1}) & \dots & \operatorname{vec}(\mathbf{R}^{\mathsf{L}}_{\mu,p}\mathbf{R}^{\mathsf{R}}_{\mu+1})\end{bmatrix}. \tag{7.18}$$

Then the augmented cores $\widetilde{\mathbf{U}}_{\alpha,\mu}$ and $\widetilde{\mathbf{U}}_{\mu+1}$, which have been formed according to (7.15), satisfy

$$\widetilde{W}^1 := \begin{bmatrix}\operatorname{vec}(\widetilde{\mathbf{U}}^{\mathsf{L}}_{\mu,1}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1}) & \operatorname{vec}(\widetilde{\mathbf{U}}^{\mathsf{L}}_{\mu,2}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1}) & \dots & \operatorname{vec}(\widetilde{\mathbf{U}}^{\mathsf{L}}_{\mu,p}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1})\end{bmatrix} = W^0 + \mathbf{R}_{\mu,\mu+1}.$$

Moving the index α in line 4 of Algorithm 7.4 does not affect this equality, so that

$$\begin{aligned}\widetilde{W}^1 &= \begin{bmatrix}\operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1,1}) & \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1,2}) & \dots & \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\widetilde{\mathbf{U}}^{\mathsf{R}}_{\mu+1,p})\end{bmatrix} \\ &= W^0 + \mathbf{R}_{\mu,\mu+1}.\end{aligned} \tag{7.19}$$

also holds for the updated cores $\mathbf{U}_\mu$ and $\widetilde{\mathbf{U}}_{\mu+1,\alpha}$. The next step of Algorithm 7.4 begins with replacing $\widetilde{\mathbf{U}}_{\mu+1,\alpha}$ by the optimized cores $\mathbf{U}_{\mu+1,\alpha}$. After this optimization, we can regard

$$W^1 = \begin{bmatrix}\operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\mathbf{U}^{\mathsf{R}}_{\mu+1,1}) & \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\mathbf{U}^{\mathsf{R}}_{\mu+1,2}) & \dots & \operatorname{vec}(\mathbf{U}^{\mathsf{L}}_{\mu}\mathbf{U}^{\mathsf{R}}_{\mu+1,p})\end{bmatrix} \tag{7.20}$$

as an improved approximation to the true DMRG solution.

To get a better understanding of the improvement attained by $\mathbf{W}^1$, we now relate this matrix to the trace minimization (7.16) solved in DMRG. By (7.5), the problem

$$\min\left\{\operatorname{trace}(V^{\mathsf{T}}\mathcal{A}_{\neq\mu+1}V) \mid V^{\mathsf{T}}V = I_p\right\},$$

which is solved for determining W^1, is equivalent to the problem

$$\min\left\{\operatorname{trace}(W^{\mathsf{T}}\mathcal{A}_{\neq\mu,\mu+1}W) \mid W \in \operatorname{range}(I_{r_{\mu+1}n_{\mu+1}} \otimes \mathbf{U}^{\mathsf{L}}_{\mu}),\ W^{\mathsf{T}}W = I_p\right\}. \tag{7.21}$$

The following proposition shows that the constraint includes all linear combinations from $\operatorname{range}(W^0)$ and $\operatorname{range}(\mathbf{R}_{\mu,\mu+1})$.

Proposition 7.2. *Consider the matrices W^0 and $\mathbf{R}_{\mu,\mu+1}$ defined above. Then the core tensor $\mathbf{U}_\mu$, obtained after moving the index α, satisfies*

$$\begin{aligned}\operatorname{range}(I_{r_{\mu+1}n_{\mu+1}} \otimes \mathbf{U}^{\mathsf{L}}_{\mu}) &\supseteq \operatorname{range}(W^0) + \operatorname{range}(\mathbf{R}_{\mu,\mu+1}) \\ &= \{W^0S + \mathbf{R}_{\mu,\mu+1}T \mid S, T \in \mathbb{R}^{p\times p}\}.\end{aligned} \tag{7.22}$$

Proof. By (7.8), moving the index α leads to

$$\begin{bmatrix}\mathbf{U}^{\mathsf{L}}_{\mu,\alpha} & \mathbf{R}^{\mathsf{L}}_{\mu,\alpha}\end{bmatrix} = \widetilde{\mathbf{U}}^{\mathsf{L}}_{\mu,\alpha} = \mathbf{U}^{\mathsf{L}}_{\mu}P_\alpha,$$

for some full rank matrix P_α, where $\mathbf{U}_{\mu,\alpha}$ and $\mathbf{U}_\mu$ refer to the cores before and after moving the index α. This shows that all columns of W^0 and $\mathbf{R}_{\mu,\mu+1}$ are in $\mathbb{R}^{r_{\mu+1}n_{\mu+1}} \otimes \operatorname{range}(\mathbf{U}^{\mathsf{L}}_{\mu})$, implying the assertion. □

---

[2] In practice, we first choose $R_{\mu,\mu+1} \in \mathbb{R}^{r_{\mu-1}n_\mu n_{\mu+1}r_{\mu+1}\times p}$ and then decompose it into the form (7.18) by an (approximate) minimal-rank decomposition.

Combined with (7.21), Proposition 7.2 allows us to conclude that the improved approximation W^1 is at least as good as the one obtained by $W^0 S + \mathbf{R}_{\mu,\mu+1} T$ with optimal choices for $S, T \in \mathbb{R}^{p \times p}$. While this implies choosing an optimal p-dimensional subspace in the (generically) $2p$-dimensional subspace $\mathrm{range}(W^0) + \mathrm{range}(\mathbf{R}_{\mu,\mu+1})$, we should emphasize however, that the actual optimization (7.21) (or, equivalently, the optimization of the $\mathbf{U}_{\mu+1,\alpha}$) is over a much larger space of (generic) dimension $r_\mu n_{\mu+1} r_{\mu+1}$. Similar considerations can be made for the right-to-left sweep. Thus, Algorithm 7.4 can be viewed as an *enhanced* local subspace correction method for solving the DMRG subproblems (7.16) approximately.

Choice of $\mathbf{R}_{\mu,\mu+1}$ The insights gained from the relation of Algorithm 7.4 to DMRG can be used to guide the choice of the local corrections $\mathbf{R}_{\mu,\mu+1}$. A natural choice, that can also be efficiently implemented, is given by

$$\mathbf{R}_{\mu,\mu+1} = -(\mathcal{A}_{\mu,\mu+1} W^0 - W^0 \Lambda) \tag{7.23}$$

with $\Lambda = (W^0)^\mathsf{T} \mathcal{A}_{\mu,\mu+1} W^0$. This is the orthogonal projection of the negative gradient of the local DMRG problem (7.16) on the Stiefel manifold. Thus the addition of $\mathbf{R}_{\mu,\mu+1}$ to W^0 as implicitly performed in the augmentation step 3 of Algorithm 7.4 can be seen as a steepest descent step for the DMRG problem.

Furthermore, due to the μ-orthogonality, $\mathbf{X}_{\neq\mu,\mu+1}^\mathsf{T} \mathbf{X}_{\neq\mu,\mu+1}$ it holds

$$\mathcal{A}_{\mu,\mu+1} W^0 - W^0 \Lambda = \mathbf{X}_{\neq\mu,\mu+1}^\mathsf{T} (AX - X\Lambda),$$

and also $\Lambda = X^\mathsf{T} A X$. Thus, the choice (7.23) can at the same time be interpreted as the projection of the global negative residual onto the local subspace of the local DMRG problem. In this sense, global residual information is injected to the local problem.

Our experiments, see Section 7.4, demonstrate that it is beneficial to use a preconditioned DMRG residual

$$\mathbf{R}_{\mu,\mu+1} = -\mathcal{B}_{\mu,\mu+1}^{-1} (\mathcal{A}_{\mu,\mu+1} W^0 - W^0 \Lambda) \tag{7.24}$$

instead of (7.23). The derivation of such a local preconditioner $\mathcal{B}_{\mu,\mu+1}^{-1}$ for $\mathcal{A}_{\mu,\mu+1}$ will be discussed in Section 7.3.4. An alternative to (7.24) would be to use $\mathbf{R}_{\mu,\mu+1} = -\mathbf{X}_{\mu,\mu+1}^\mathsf{T} B^{-1} (AX - X\Lambda)$, where B^{-1} is a preconditioner for A. This corresponds to injecting information on the preconditioned global residual into the local DMRG problem. Although this choice is better than using no preconditioner information, our numerical experiments indicate that it is inferior to the preconditioned local DMRG residual (7.24).

Incorporating the residual increases the TT rank to $r_\mu \leftarrow r_\mu + s_\mu$, where s_μ is the rank of $\mathbf{R}_{\mu,\mu+1}$. The exact computation of $\mathbf{R}_{\mu,\mu+1}$ by (7.24) will typically lead to an unacceptably large value of s_μ. To avoid this, we use repeated low-rank truncation to limit this additional rank to $s_\mu \leq 2$. Such a truncation can be justified by the fact that the addition of $\mathbf{R}_{\mu,\mu+1}$ to W^0 only serves as a very rough subspace correction, with the fine tuning performed by ALS.

7.3.3. Convergence analysis

For alternating optimization schemes, only local convergence results are available [EHK15, RU13, Usc12]. To still be able to prove convergence of the AMEn algorithm for linear systems, Dolgov *et al.* [DS14] use the following reasoning:

- The enrichment step can be seen as a modified steepest descent step with guaranteed global convergence.
- The local core optimization steps do not worsen the current iterate.

They obtain global convergence but with a convergence rate that deteriorates exponentially fast with d. This result is very pessimistic, as it does not take into account the progress due to the local core optimization steps, which, in practice, contribute the most. Furthermore, the residual is assumed to be exact or well-approximated. In practice, the residual is heavily truncated to avoid excessive rank growth.

For the (block) eigenvalue case we face the added difficulty that the trace minimization problem is not a convex cost function. Thus, the iteration may converge to a subspace X spanned by p eigenvectors which are not necessarily the smallest.

Let $X = \mathbf{X}_{\neq\mu,\mu+1}W^0$ with W^0 as in (7.17) be our current iterate at the μth core in Algorithm 7.4. Adding the local DMRG-residual $\mathbf{R}_{\mu,\mu+1}$ as in (7.23) results in an enriched iterate $\widetilde{\mathbf{X}}$,

$$\widetilde{X} = X - \alpha \mathbf{X}_{\mu,\mu+1}\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda),$$

where $\alpha \in \mathbb{R}$ is a parameter controlling the strength of the enrichment. Alternatively, for the preconditioned DMRG residual (7.24), we obtain the enriched iterate

$$\widetilde{X} = X - \alpha \mathbf{X}_{\mu,\mu+1}\mathcal{B}_{\mu,\mu+1}^{-1}\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda).$$

This bears close resemblance to a steepest descent scheme with a modified search direction $\xi = -\mathbf{X}_{\mu,\mu+1}\mathcal{B}_{\mu,\mu+1}^{-1}\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda)$. Indeed, we can check that ξ is a descent direction. The Riemannian gradient at X for $f(X) = \operatorname{trace}(X^{\mathsf{T}}AX)$ is given by

$$\operatorname{grad} f(X) = AX - X\Lambda,$$

see (7.13). Thus, we compute

$$\begin{aligned}\langle \operatorname{grad} f(X), \xi\rangle &= \langle \operatorname{grad} f(X), -\mathbf{X}_{\mu,\mu+1}\mathcal{B}_{\mu,\mu+1}^{-1}\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda)\rangle \\ &= -\langle \mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda), \mathcal{B}_{\mu,\mu+1}^{-1}\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda)\rangle\end{aligned}$$

As the local preconditioner $\mathcal{B}_{\mu,\mu+1}^{-1}$ is symmetric positive definite, we can perform a Cholesky decomposition $B_{\mu,\mu+1}^{-1} = C^{\mathsf{T}}C$ and to obtain

$$-\langle C\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda), C\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda)\rangle = -\|C\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}(AX - X\Lambda)\|^2 \leq 0.$$

Note that we cannot guarantee that this choice of ξ also defines a gradient-related sequence in the sense of Def. 2.22: Even if $AX - X\Lambda$ is guaranteed to be non-zero at a non-critical point, it can be an element of the kernel of $\mathbf{X}_{\mu,\mu+1}^{\mathsf{T}}$. We can assume that $\|C\|$ is bounded by choosing the local preconditioner $\mathcal{B}_{\mu,\mu+1}^{-1}$ accordingly.

If we could guarantee the gradient-relatedness we would obtain global convergence to critical points of f, see Thm. 2.24 when choosing the parameter α according to the Armijo rule, Def. 2.23, We note that the step size α is formally not present in Algorithm 7.4. Instead, it is implicitly included in the local core optimization step (7.21), which performs an optimization over a subspace including *all* linear combinations from $\operatorname{range}(W^0) + \operatorname{range}(\mathbf{R}_{\mu,\mu+1})$, see Prop. 7.2.

7.3.4. Construction of local preconditioners

In our algorithms, preconditioning is important at two points:

- The reduced eigenvalue problems (7.10) are solved by LOBPCG [Kny01]. The convergence, and hence the time needed, by LOBPCG crucially depends on the availability of a good preconditioner for the reduced matrix $\mathcal{A}_\mu = \mathbf{X}_{\neq\mu}^\mathsf{T} A \mathbf{X}_{\neq\mu}$.
- The preconditioned DMRG residual (7.24) for the augmentation steps in EVAMEn requires a preconditioner for $\mathcal{A}_{\mu,\mu+1} = \mathbf{X}_{\neq\mu,\mu+1}^\mathsf{T} A \mathbf{X}_{\neq\mu,\mu+1}$.

Both problems consist of finding a preconditioner for a reduced matrix $Y^\mathsf{T} A Y$ where Y has orthonormal columns. As explained in [KT11b], a preconditioner B^{-1} for A leads to an at least equally good preconditioner

$$B^{-1} \approx (Y^\mathsf{T} B Y)^{-1} \tag{7.25}$$

for $Y^\mathsf{T} A Y$. It is a nontrivial task to efficiently construct and apply $(Y^\mathsf{T} B Y)^{-1}$.

A special case are Laplace-like operators of the form (6.3), which we have already examined in Chapter 6. In this case, we have seen in (6.35) that $\mathcal{A}_{\neq\mu}$ and $\mathcal{A}_{\neq\mu,\mu+1}$ themselves can be written as Laplace-like operators:

$$\mathcal{A}_{\neq\mu} = \mathcal{L}_{\geq\mu+1} \otimes I_{r_\mu} \otimes I_{n_1\cdots n_{\mu-1}} + I_{n_{\mu+1}\cdots n_d} \otimes L_\mu \otimes I_{n_1\cdots n_{\mu-1}} + I_{n_{\mu+1}\cdots n_d} \otimes I_{r_\mu} \otimes \mathcal{L}_{\leq\mu-1}.$$

Both $\mathcal{L}_{\leq\mu-1}$ and $\mathcal{L}_{\geq\mu+1}$ are assembled efficiently using low-dimensional contractions. As a consequence of this short Laplace-like representation, $\mathcal{A}_{\neq\mu}^{-1}$ and $\mathcal{A}_{\neq\mu,\mu+1}^{-1}$ can be well approximated by a sum of Kronecker products of exponentials [Gra04]:

$$\mathcal{A}_{\neq\mu}^{-1} \approx \sum_{k=1}^{R} \frac{\omega_k}{\lambda_{\min}} \exp\left(-\frac{\alpha_k}{\lambda_{\min}} \mathcal{L}_{\geq\mu+1}\right) \otimes \exp\left(-\frac{\alpha_k}{\lambda_{\min}} L_\mu\right) \otimes \exp\left(-\frac{\alpha_k}{\lambda_{\min}} \mathcal{L}_{\leq\mu-1}\right)$$

The approximation error in the spectral norm is governed by the maximal error of the approximation

$$\frac{1}{x} \approx \sum_{k=1}^{R} \omega_k \mathrm{e}^{-\alpha_k x}$$

on the interval $[1, \lambda_{\max}/\lambda_{\min}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and largest eigenvalue of $\mathcal{A}_{\neq\mu}$, see [Hac12, Sec. 9.7.2]. Usually, rather small values for R are sufficient to attain an acceptable error. Almost optimal values for R, ω_k, and α_k to achieve a specific accuracy have been reported in [Hac10].

In the examples considered in our numerical experiments, $\mathcal{A}$ is always composed of a Laplace-like operator and a potential. Our preconditioner is then based on the Laplace-like part alone.

7.4. Numerical experiments

To assess the performance of Algorithm 7.4 (EVAMEn), we calculate the p smallest eigenvalues of the PDE eigenvalue problem

$$\begin{aligned} -\Delta u(x) + V(x)u(x) &= \lambda u(x) \quad \text{for } x \in \Omega =]a,b[^d, \\ u(x) &= 0 \qquad\quad\ \text{for } x \in \partial\Omega, \end{aligned} \tag{7.26}$$

where Δ is the d-dimensional Laplace operator, and V is a potential. The finite difference discretization using a regular grid with n grid points in every dimension yields an n^d-dimensional matrix eigenvalue problem

$$Ax = \lambda x, \quad A \in \mathbb{R}^{n^d \times n^d},$$

where $A = L + V$ is composed of the discretized Laplace operator

$$L = \sum_{\mu=1}^{d} I_n \otimes \cdots \otimes L_\mu \otimes \cdots \otimes I_n$$

and the discretized potential V. Finding the p smallest eigenvalues of A then corresponds to the solution of the trace minimization problem (7.1).

Both algorithms, ALS and EVAMEn, have been implemented in MATLAB version 2013b. To solve the reduced eigenvalue problems (7.10), we use Knyazev's implementation of LOBPCG[3] with the stopping tolerance

$$\max\{\, 10^{-6},\ \min\{\, 10^{-2},\ 10^{-2}\|R_{\mu,\mu+1}\|\}\,\}.$$

The preconditioner for LOBPCG is a rank-three approximation of $(\mathbf{X}_{\neq\mu}^{\mathsf{T}} L \mathbf{X}_{\neq\mu})^{-1}$ by exponential sums, see Section 7.3.4). Similarly, the preconditioner for $\mathcal{A}_{\mu,\mu+1}$ needed in EVAMEn, see (7.24), is a rank-three approximation of $(\mathbf{X}_{\neq\mu,\mu+1}^{\mathsf{T}} L \mathbf{X}_{\neq\mu,\mu+1})^{-1}$ by exponential sums.

In our numerical implementation, we use repeatedly truncated singular value decompositions to prevent excessive rank growth. In particular, this concerns the following two points:

- The residual $\mathbf{R}_{\mu,\mu+1}$, see (7.23), needed for EVAMEn is truncated to rank two before and after the preconditioner $\mathcal{B}_{\mu,\mu+1}^{-1}$ is applied. This is only done when more than one eigenvalue is sought ($p > 1$).
- When shifting the index α (Algorithms 7.1 and 7.2) we discard, as suggested in [DKOS13, PV2012], all singular values below a threshold `tol_sv`. We choose a relative tolerance 10^{-8}, but do not allow the ranks to grow larger than 40 in both ALS and EVAMEn.

All computations and timings were performed on a 12-core Intel Xeon CPU X5675, 3.07GHz with 192 GB RAM running 64-Bit Linux version 2.6.32.

---

[3] Available at `http://www.mathworks.com/matlabcentral/fileexchange/48-lobpcg-m`.

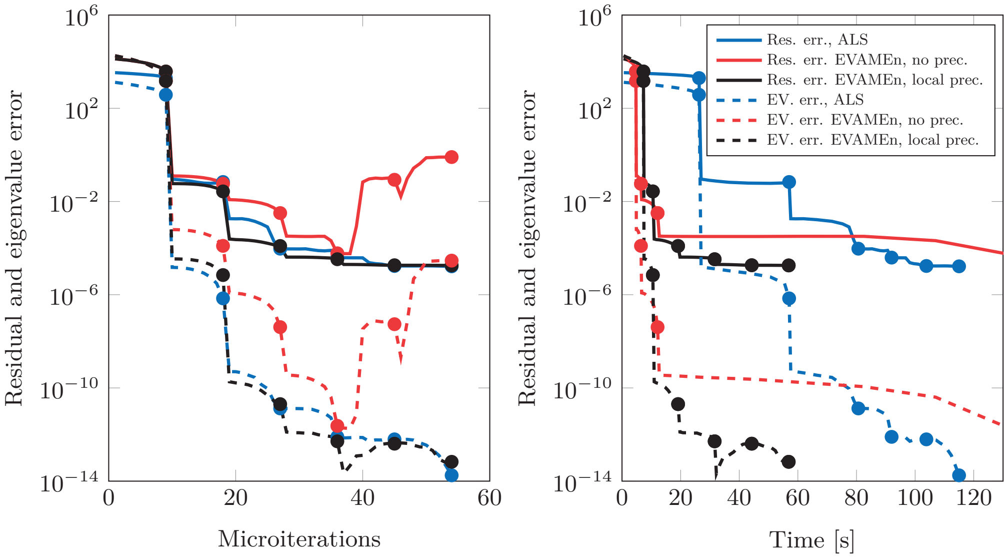


Figure 7.3: Comparison of ALS (blue lines) to EVAMEn without preconditioning (red lines) and with preconditioning (black lines) for calculating the smallest eigenvalue of (7.26) *using the Newton potential,* $\Omega =]-1, 1[^{10}$*, and* $n = 128$ *grid points in every dimension. Each marker indicates the completion of a half-sweep.*

7.4.1. Setting 1: One eigenvalue for the Newton potential

First, consider again the Newton potential, see also Section 6.3.1, defined by $V(x) = \frac{1}{\|x\|}$. We use an exponential sum with ten terms [Hac10, Hac12] to construct a rank-10 TT approximation of the discretized potential $\mathbf{V}$, with an accuracy of $3.6 \cdot 10^{-5}$. This approximate potential is applied to vectors in TT format using the Hadamard product, see Section 4.2.4. We use $d = 10$, $\Omega =]-1, 1[^{d}$ and $n = 128$ grid points in every dimension, yielding a discretized eigenvalue problem of size 128^{10}.

As a first experiment we calculate only the smallest eigenvalue, that is, $p = 1$. In this case, no shifting of the index α is performed and therefore ALS is unable to adapt the ranks. We initialize ALS with a random block-1 TT tensor having all ranks equal to $r_\mu = 8$, corresponding to the ranks obtained by EVAMEn, which is initialized with a random block-1 TT tensor. Figure 7.3 shows the obtained convergence in terms of the eigenvalue error (dashed lines) and the residual (solid lines), both with respect to the number of iterations and with respect to execution time.

Surprisingly, the preconditioned version (7.24) of EVAMEn makes almost identical progress per iteration as ALS, but the first few half-sweeps take significantly less time. The latter is explained by the fact that EVAMEn operates with smaller TT ranks for the first few half-sweeps.

It can be clearly observed that preconditioning the residual $\mathbf{R}_{\mu,\mu+1}$ is important, as the convergence of the unpreconditioned version (7.23) is significantly worse. The errors even start growing for later iterations, due to convergence failures of LOBPCG. This experiment allows us to conclude that the choice of the core augmentation in EVAMEn matters, and should not be taken randomly, which would already allow for rank adaptivity. Therefore, we only consider the preconditioned version of EVAMEn in the following experiments.

7.4.2. Setting 2: Multiple eigenvalues for the Newton potential

This setting is identical to Setting 1, except that we now seek for the $p > 1$ smallest eigenvalues. In this case, shifting the index α provides a mechanism for rank adaption in both ALS and EVAMEn. Therefore, ALS is now also initialized with a random rank-1 block-1 TT tensor.

We compare ALS with EVAMEn for $p = 3$ and $p = 11$ in Figures 7.4 and 7.5, respectively. Solid lines correspond to the scaled Frobenius norm of the residual, $p^{-1/2}\|AX - X\Lambda\|_F$, while dashed lines correspond to the trace error, $|\operatorname{trace}(X^\mathsf{T}AX) - \operatorname{trace}(X_{\text{final}}^\mathsf{T}AX_{\text{final}})|$, where X_{final} is a reference solution computed by the ALS procedure with high accuracy.

Since $d = 10$, the second largest eigenvalue of the discrete Laplace operator has multiplicity 10 and therefore the eigenvalues $\lambda_2, \ldots, \lambda_{11}$ of A form a cluster. This cluster is broken when using $p = 3$, resulting in very slow convergence of the LOBPCG iterations when no local preconditioner is used.

In both cases, EVAMEn outperforms ALS in terms of both required number of iterations and computational time. The final TT ranks in the block format of the solutions are bounded by 26 for $p = 3$, and attain the allowed limit of 40 for $p = 11$. Consequently, the obtained final accuracy for $p = 11$ is less than for $p = 3$.

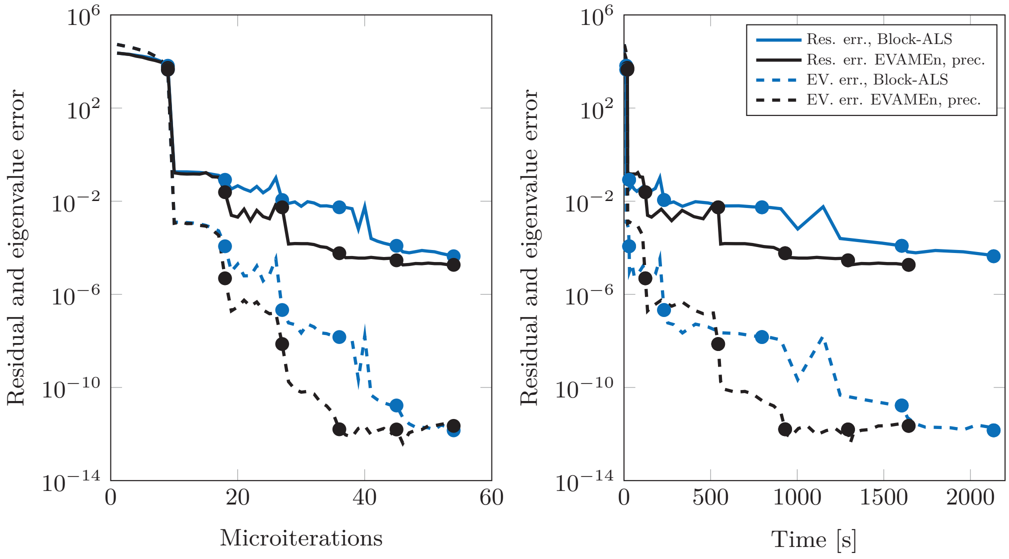


Figure 7.4: Comparison of block ALS (blue lines) and preconditioned EVAMEn (black lines) for calculating the three smallest eigenvalues of (7.26) *using the Newton potential,* $\Omega =]-1,1[^{10}$, *and* $n = 128$ *grid points in every dimension.*

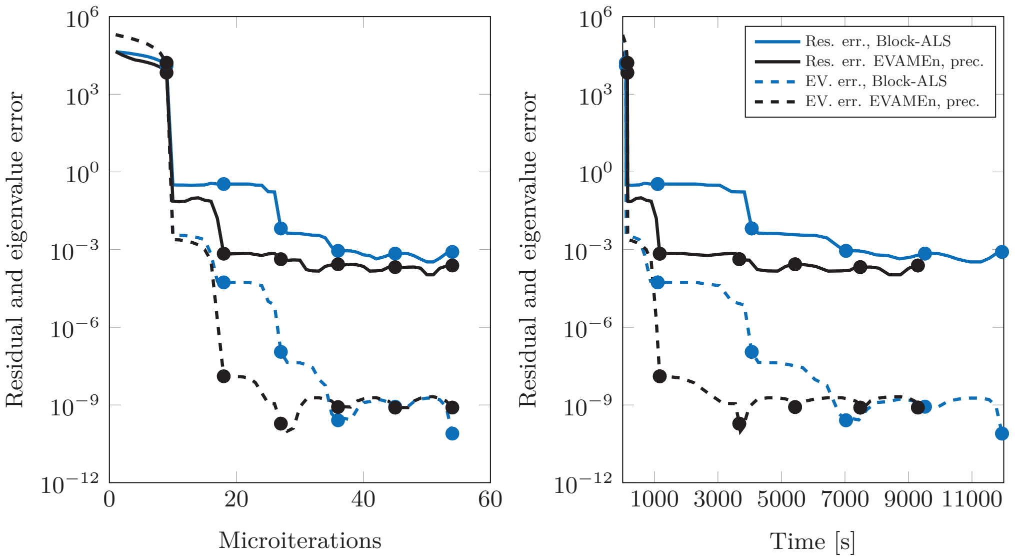


Figure 7.5: Comparison of block ALS (blue lines) and preconditioned EVAMEn (black lines) for calculating the eleven smallest eigenvalues of (7.26) *using the Newton potential,* $\Omega =]-1,1[^{10}$*, and* $n = 128$ *grid points in every dimension.*

7.4.3. Setting 3: Henon-Heiles potential

As a second test case, we take the modified Henon-Heiles potential [FGL09, MMC90, RM00]

$$V(x) = \frac{1}{2}\sum_{\mu=1}^{d} x_\mu^2 + \sum_{\mu=1}^{d-1} \left(\sigma_* \left(x_\mu x_{\mu+1}^2 - \frac{1}{3} x_\mu^3 \right) + \frac{\sigma_*^2}{16} \left(x_\mu^2 + x_{\mu+1}^2 \right)^2 \right),$$

modelling a coupled oscillator. This potential is usually defined on the entire real space. As in [DKOS13], we apply spectral collocation, using a tensor product grid based on the zeros $\xi_1, \ldots, \xi_n$ of the nth Hermite polynomial. The corresponding discrete one-dimensional Laplace operator is given by [BH86]

$$(L_H)_{ij} = \begin{cases} \frac{1}{6}(4n - 1 - 2\xi_i^2)\ , & i = j, \\ (-1)^{(i-j)} \left(\frac{1}{(\xi_i - \xi_j)^2} - \frac{1}{2} \right)\ , & i \neq j. \end{cases}$$

The discretized operator $\mathcal{A}$ then takes the matrix representation

$$\sum_{\mu=1}^{d} I_n \otimes \ldots I_n \otimes L_\mu \otimes I_n \otimes \cdots \otimes I_n + \sum_{\mu=1}^{d-1} I_n \otimes \cdots \otimes C_{\mu+1} \otimes B_\mu \otimes \cdots \otimes I_n, \quad (7.27)$$

with $B_\mu = \sigma_* D + \frac{\sigma_*^2}{8} D^2$, $C_{\mu+1} = D^2$, and

$$L_\mu = \begin{cases} L_H + \frac{1}{2}D^2 - \frac{\sigma_*}{3}D^3 + \frac{\sigma_*^2}{16}D^4 & \text{for } \mu = 1, \\ L_H + \frac{1}{2}D^2 - \frac{\sigma_*}{3}D^3 + \frac{\sigma_*^2}{8}D^4 & \text{for } 1 < \mu < d, \\ L_H + \frac{1}{2}D^2 + \frac{\sigma_*^2}{16}D^4 & \text{for } \mu = d, \end{cases}$$

where $D = \operatorname{diag}(\xi_1, \xi_2, \ldots, \xi_n)$ contains the grid points for one dimension. Hence $\mathcal{A}$ allows for an operator TT representation with all ranks equal to 3 using the cores described in Section 4.7.

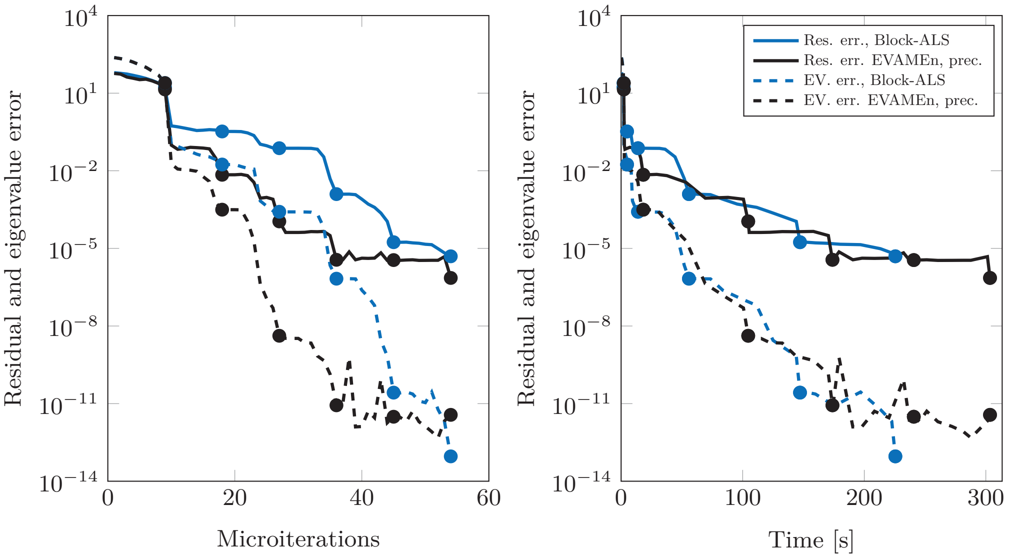


Figure 7.6: Comparison of block ALS (blue lines) and EVAMEn (black lines) for calculating the three smallest eigenvalues of (7.26) *using the Henon-Heiles potential with* $d = 10$ *and* $n = 28$ *collocation points in every dimension.*

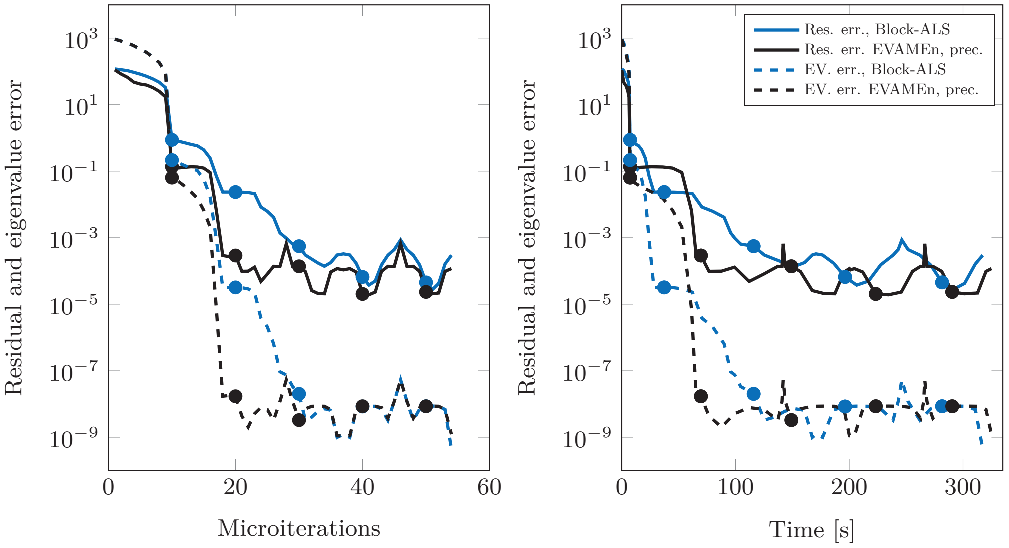


Figure 7.7: Comparison of block ALS (blue lines) and EVAMEn (black lines) for calculating the eleven smallest eigenvalues of (7.26) *using the Henon-Heiles potential with* $d = 10$ *and* $n = 28$ *collocation points in every dimension.*

The first term in (7.27), which is composed of the discrete Laplacian $\mathcal{L}$ *and* the non-interacting parts of the potential, is taken as an approximation $\mathcal{B}$ of $\mathcal{A}$ for the local preconditioner.

In our experiments we choose $\sigma_* = 0.11$, and consider (7.26) with $d = 10$ and $n = 28$ collocation points in every dimension. Thus, the discretized eigenvalue problem is of dimension 28^{10}.

Figure 7.6 shows the obtained results when calculating $p = 3$ eigenvalues. Similarly to the Newton potential, EVAMEn performs better than ALS in terms of the number of iterations. Since this is offset by the higher cost per iteration, both algorithms perform equally well with respect to time. For $p = 11$, EVAMEn converges only slightly better and the obtained timings are similar for both algorithms, see Figure 7.7. Possibly, the local nature of the potential benefits ALS to an extent that it can hardly be improved by injecting residual information.

7.5. Conclusion

We have developed a new algorithm called EVAMEn, which allows to incorporate preconditioned residual information into the ALS procedure for computing one or several eigenvectors in the (block) TT format. For the Newton potential, the numerical experiments clearly demonstrate the benefits obtained from incorporating this information and the importance of using local preconditioners. We expect that using such preconditioners would also be beneficial in the AMEn algorithm for solving linear systems.

The use of the TT format makes it expensive to work with large mode sizes n_μ, as n_μ enters directly into the size of the TT cores $\mathbf{X}_\mu \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$. Unless the ranks are very small, this imposes computational restrictions on n_μ and hence on the discretization. A simple way to avoid this is to use low-dimensional subspaces of $\mathbb{R}^{n_\mu}$ in every mode. This has been considered for the TT format in [Ose11c] and it is commonplace for the hierarchical Tucker (HT) format [HK09, Gra10]. Concerning the latter, it is certainly possible to extend the block TT format to the HT format, but the derivations and the resulting algorithms can be expected to become significantly more technical. We refer to [KT11b] for a discussion on the ALS and DMRG algorithms for computing a single eigenvector in the HT format.

Chapter A

▶ Appendix: Smooth decompositions of matrices

In this chapter, we derive a short corollary which deals with the question of how the factors of a decomposition of a matrix $A \in \mathbb{R}^{m\times n}$ change when A varies in a smooth way. This result is used in the proof of the manifold structure of the set of tensors with fixed multilinear rank, see Section 3.4.

Theorem A.1 (Chern/Dieci, [CD00, Thm. 2.4])**.** *Let $A(t) \in C^k(\mathbb{R}, \mathbb{R}^{m\times n})$, $m \geq n$ and $k \geq 0$, have constant rank:* $\operatorname{rank}(A(t)) = r$ *for all t, $1 \leq r \leq n$ fixed. Then there exist orthogonal $U(t) \in C^k(\mathbb{R}, \mathbb{R}^{m\times m})$ and $V(t) \in C^k(\mathbb{R}, \mathbb{R}^{n\times n})$ such that*

$$U^\mathsf{T}(t)A(t)V(t) = \begin{bmatrix} S_+(t) & 0 \\ 0 & 0 \end{bmatrix}$$

where $S_+(t) \in C^k(\mathbb{R}, \mathbb{R}^{m\times m})$ is symmetric positive definite.

From which we can deduct the following corollary for fat matrices with many more columns than rows.

Corollary A.2. *Consider the smooth (C^∞) matrix $A(t) \in \mathbb{R}^{m\times n}$, $n \geq m$ with constant rank* $\operatorname{rank}(A(t)) = r$ *for all t. Then there exists a smooth QR-like-decomposition*

$$A(t) = Q(t)R(t)$$

with smooth matrices $Q(t) \in \mathbb{R}^{m\times r}$ having orthonormal columns and $R(t) \in \mathbb{R}^{r\times n}$. The columns of $Q(t)$ are the first r left singular vectors of $A(t)$.

Proof. We look at the transpose $A^\mathsf{T}(t)$, a skinny matrix, and apply Theorem A.1 to it to obtain

$$A^\mathsf{T}(t) = U(t)\begin{bmatrix} S_+(t) & 0 \\ 0 & 0 \end{bmatrix} V^\mathsf{T}(t).$$

Transposing this result yields

$$A(t) = V^\mathsf{T}(t)\begin{bmatrix} S_+(t) & 0 \\ 0 & 0 \end{bmatrix} U(t) = Q(t)\begin{bmatrix} S_+(t) & 0 \end{bmatrix} U(t) = Q(t)R(t),$$

where $Q(t)$ consists of the first r columns of $V^\mathsf{T}(t)$ and $R(t) := \begin{bmatrix} S_+(t) & 0 \end{bmatrix} U(t)$. □

▶ Bibliography

[AKLT87] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, *Rigorous results on valence-bond ground states in antiferromagnets*, Phys. Rev. Lett. **59** (1987), no. 7, 799–802.

[AKLT88] ______, *Valence bond ground states in isotropic quantum antiferromagnets*, Comm. Math. Phys. **115** (1988), no. 3, 477–528.

[AM12] P.-A. Absil and J. Malick, *Projection-like retractions on matrix manifolds*, SIAM J. Control Optim. **22** (2012), no. 1, 135–158.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, 2008.

[AMSVD02] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren, *A Grassmann-Rayleigh quotient iteration for computing invariant subspaces*, SIAM Rev. **44** (2002), no. 1, 57–73.

[AMT13] P.-A. Absil, R. Mahony, and J. Trumpf, *An extrinsic look at the Riemannian Hessian*, Geometric Science of Information (F. Nielsen and F. Barbaresco, eds.), Lecture Notes in Computer Science, vol. 8085, Springer Berlin Heidelberg, 2013, pp. 361–368 (English).

[AO14] P.-A. Absil and I. Oseledets, *Low-rank retractions: a survey and new results*, Computational Optimization and Applications (2014), 5–29.

[BD15] M. Bachmayr and W. Dahmen, *Adaptive near-optimal rank tensor approximation for high-dimensional operator equations*, Found. Comput. Math. **15** (2015), no. 4, 839–898.

[Beb00] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math. **86** (2000), no. 4, 565–589.

[BG05] S. Börm and L. Grasedyck, *Hybrid cross approximation of integral operators*, Numer. Math. **101** (2005), no. 2, 221–249.

[BG13] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl. **20** (2013), no. 1, 27–43.

[BG14] J. Ballani and L. Grasedyck, *Hierarchical tensor approximation of output quantities of parameter-dependent PDEs*, Tech. report, ANCHP, EPF Lausanne, Switzerland, March 2014.

[BGBMN91] A. Bunse-Gerstner, R. Byers, V. Mehrmann, and N. K. Nichols, *Numerical computation of an analytic singular value decomposition of a matrix valued function*, Numer. Math. **60** (1991), no. 1, 1–39.

[BGK13] J. Ballani, L. Grasedyck, and M. Kluge, *Black box approximation of tensors in hierarchical Tucker format*, Linear Algebra Appl. **438** (2013), no. 2, 639–657.

[BGL05] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer. **14** (2005), 1–137.

[BH86] D. Baye and P.-H. Heenen, *Generalised meshes for quantum mechanical problems*, J. Phys. A **19** (1986), no. 11, 2041–2059.

[BH05] D. Braess and W. Hackbusch, *Approximation of $1/x$ by exponential sums in* $[1, \infty)$, IMA J. Numer. Anal. **25** (2005), no. 4, 685–697.

[BK+12] B. W. Bader, T. G. Kolda, et al., *Matlab tensor toolbox version 2.5*, Available from `http://www.sandia.gov/~tgkolda/TensorToolbox/`, January 2012.

[Boc87] H. G. Bock, *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, Bonner Math. Schriften, 1987.

[BPK96] J. H. Bramble, J. E. Pasciak, and A. V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math. **6** (1996), no. 2, 159–189.

[BR03] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing **70** (2003), no. 1, 1–24.

[CD00] J.-L. Chern and L. Dieci, *Smoothness and periodicity of some matrix decompositions*, SIAM J. Matrix Anal. Appl. **22** (2000), no. 3, 772–792.

[CEM12] E. Cancès, V. Ehrlacher, and Y. Maday, *Periodic Schrödinger operators with local defects and spectral pollution*, SIAM J. Numer. Anal. **50** (2012), no. 6, 3016–3035.

[CT09] E. J. Candès and T. Tao, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Trans. Inform. Theory **56** (2009), no. 5, 2053–2080.

[DKOS13] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Preprint 59/2013, MPI MIS Leipzig, 2013.

[DL97] L. De Lathauwer, *Signal processing based on multilinear algebra*, Ph.D. thesis, Katholike Universiteit Leuven, 1997.

[LMV00] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl. **21** (2000), no. 4, 1253–1278.

[DO12] S. V. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput. **34** (2012), no. 5, A2718–A2739.

[Dol13] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian J. Numer. Anal. Math. Modelling **28** (2013), no. 2, 149–172.

[DS13a] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions. Part I: SPD systems*, arXiv preprint arXiv:1301.6068 (2013).

[DS13b] ______, *Alternating minimal energy methods for linear systems in higher dimensions. Part II: Faster algorithm and application to nonsymmetric systems*, arXiv preprint arXiv:1304.1222 (2013).

[DS14] ______, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput. **36** (2014), no. 5, A2248–A2271.

[DS15] ______, *Corrected One-Site Density Matrix Renormalization Group and Alternating Minimal Energy Algorithm*, Numerical Mathematics and Advanced Applications - ENUMATH 2013 (A. Abdulle, S. Deparis, D. Kressner, F. Nobile, and M. Picasso, eds.), Lecture Notes in Computational Science and Engineering, vol. 103, Springer International Publishing, 2015, pp. 335–343.

[DSH13] C. Da Silva and F. J. Herrmann, *Hierarchical tucker tensor optimization – applications to tensor completion*, SAMPTA, 2013.

[DSH15] ———, *Hierarchical Tucker tensor optimization – applications to tensor completion*, Linear Algebra Appl. **481** (2015), 131–173.

[EAS99] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl. **20** (1999), no. 2, 303–353.

[EHK15] M. Espig, W. Hackbusch, and A. Khachatryan, *On the convergence of alternating least squares optimisation in tensor format representations*, arXiv preprint 1506.00062, 2015.

[FGL09] E. Faou, V. Gradinaru, and C. Lubich, *Computing semiclassical quantum dynamics with Hagedorn wavepackets*, SIAM J. Sci. Comput. **31** (2009), no. 4, 3027–3041.

[FNA04] D. H. Foster, S. M. C. Nascimento, and K. Amano, *Information limits on neural identification of colored surfaces in natural scenes*, Visual Neurosci. **21** (2004), 331–336.

[FNW92] M. Fannes, B. Nachtergaele, and R. F. Werner, *Finitely correlated states on quantum spin chains*, Comm. Math. Phys. **144** (1992), no. 3, 443–490.

[GKK13] L. Grasedyck, M. Kluge, and S. Krämer, *Alternating directions fitting (ADF) of hierarchical low rank tensors*, DFG SPP 1324 Preprint 149, 2013.

[GKK15] ———, Personal Communication, 2015.

[GKT13] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt. **36** (2013), no. 1, 53–78.

[Gra04] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing **72** (2004), no. 3–4, 247–265.

[Gra10] ———, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl. **31** (2010), no. 4, 2029–2054.

[GRY11] S. Gandy, B. Recht, and I. Yamada, *Tensor completion and low-n-rank tensor recovery via convex optimization*, Inverse Problems **27** (2011), no. 2, 025010.

[GTZ97] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl. **261** (1997), 1–21.

[GVL13] G. H. Golub and C. F. Van Loan, *Matrix computations*, fourth ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013.

[Hac10] W. Hackbusch, *Entwicklungen nach Exponentialsummen*, Technical Report 4/2005, MPI MIS Leipzig, 2010, Revised version September 2010.

[Hac12] ———, *Tensor spaces and numerical tensor calculus*, Springer, Heidelberg, 2012.

[Hen94] R. Henrion, *N-way principal component analysis theory, algorithms and applications*, Chemometrics and intelligent laboratory systems **25** (1994), no. 1, 1–23.

[HFY$^+$04] R. J. Harrison, G. I. Fann, T. Yanai, Z. Gan, and G. Beylkin, *Multiresolution quantum chemistry: Basic theory and initial applications*, J. Chem. Phys. **121** (2004), no. 23, 11587–11598.

[HK06] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions*, Computing **76** (2006), no. 3-4, 177–202.

[HK09] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl. **15** (2009), no. 5, 706–722.

[HKST12] W. Hackbusch, B. N. Khoromskij, S. A. Sauter, and E. E. Tyrtyshnikov, *Use of tensor formats in elliptic eigenvalue problems*, Numer. Linear Algebra Appl. **19** (2012), no. 1, 133–151.

[HRS12a] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput. **34** (2012), no. 2, A683–A713.

[HRS12b] ———, *On manifolds of tensors of fixed TT-rank*, Numer. Math. **120** (2012), no. 4, 701–731.

[HW12] T. Huckle and K. Waldherr, *Subspace iteration methods in terms of matrix product states*, PAMM **12** (2012), no. 1, 641–642.

[HWSH13] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen, *Computations in quantum tensor networks*, Linear Algebra Appl. **438** (2013), no. 2, 750–781.

[Kho14] V. Khoromskaia, *Black-box Hartree–Fock solver by tensor numerical methods*, Comput. Methods Appl. Math. **14** (2014), no. 1, 89–111.

[KKF11] B. N. Khoromskij, V. Khoromskaia, and H.-J. Flad, *Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format*, SIAM J. Sci. Comput. **33** (2011), no. 1, 45–65.

[KL10] O. Koch and C. Lubich, *Dynamical tensor approximation*, SIAM J. Matrix Anal. Appl. **31** (2010), no. 5, 2360–2375.

[KM15] H. Kasai and B. Mishra, *Riemannian preconditioning for tensor completion*, arXiv preprint 1506.02159, 2015.

[KMO10] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from noisy entries*, JMLR **11** (2010), 2057–2078.

[Kny01] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput. **23** (2001), no. 2, 517–541.

[KO10a] B. N. Khoromskij and I. V. Oseledets, *DMRG+QTT approach to computation of the ground state for the molecular Schrödinger operator*, Tech. Report 69/2010, MPI MIS Leipzig, 2010.

[KO10b] ———, *Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs*, Comput. Meth. Appl. Math. **10** (2010), no. 4, 376–394.

[KOS12] B. N. Khoromskij, I. V. Oseledets, and R. Schneider, *Efficient time-stepping scheme for dynamics on TT-manifolds*, Tech. Report 24, MPI MIS Leipzig, 2012.

[KPT14] D. Kressner, M. Plešinger, and C. Tobler, *A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations*, Numer. Linear Algebra Appl. **21** (2014), no. 5, 666–684.

[KRS13] V. Kazeev, O. Reichmann, and Ch. Schwab, *Low-rank tensor structure of linear diffusion operators in the TT and QTT formats*, Lin. Alg. Appl. **438** (2013), no. 11, 4204–4221.

[KS11] B. N. Khoromskij and Ch. Schwab, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput. **33** (2011), no. 1, 364–385.

[KSU14] D. Kressner, M. Steinlechner, and A. Uschmajew, *Low-rank tensor methods with subspace correction for symmetric eigenvalue problems*, SIAM J. Sci. Comput. **36** (2014), no. 5, A2346–A2368.

[KSV14] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT **54** (2014), no. 2, 447–468.

[KSV15] ———, *Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure*, Technical report MATHICSE 18.2015, EPF Lausanne, Switzerland, 2015.

[KT10] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl. **31** (2010), no. 4, 1688–1714.

[KT11a] ———, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl. **32** (2011), no. 4, 1288–1316.

[KT11b] ———, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Comput. Methods Appl. Math. **11** (2011), no. 3, 363–381.

[KT14] ———, *Algorithm 941:* `htucker`*—a Matlab toolbox for tensors in hierarchical tucker format*, ACM Trans. Math. Softw. **40** (2014), no. 3, 22:1–22:22.

[KVM01] H. A. L. Kiers and I. Van Mechelen, *Three-way component analysis: Principles and illustrative application.*, Psych. methods **6** (2001), no. 1, 84–110.

[Leb11] O. S. Lebedeva, *Tensor conjugate-gradient-type method for Rayleigh quotient minimization in block QTT-format*, Russian J. Numer. Anal. Math. Modelling **26** (2011), no. 5, 465–489.

[Lee03] J. M. Lee, *Introduction to smooth manifolds*, Graduate Texts in Mathematics, vol. 218, Springer-Verlag, New York, 2003.

[LMW12] A. Logg, K.-A. Mardal, and G. N. Wells, *Automated solution of differential equations by the finite element method*, Lecture Notes in Computational Science and Engineering, vol. 84, Springer-Verlag, Berlin, 2012.

[LMWY09] J. Liu, P. Musialski, P. Wonka, and J. Ye, *Tensor completion for estimating missing values in visual data*, Computer Vision, 2009 IEEE 12th International Conference on, 2009, pp. 2114–2121.

[LO14] C. Lubich and I. V. Oseledets, *A projector-splitting integrator for dynamical low-rank approximation*, BIT **54** (2014), no. 1, 171–188.

[LOV15] C. Lubich, I. V. Oseledets, and B. Vandereycken, *Time integration of tensor trains*, SIAM J. Numer. Anal. **53** (2015), no. 2, 917–941.

[LS13] Y. Liu and F. Shang, *An efficient matrix factorization method for tensor completion*, IEEE Signal Processing Letters **20** (2013), no. 4, 307–310.

[Lub08] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, Zurich Lectures in Advanced Mathematics, European Mathematical Society (EMS), Zürich, 2008.

[Lub15] ———, *Time Integration in the Multiconfiguration Time-Dependent Hartree Method of Molecular Quantum Dynamics*, Appl. Math. Res. Express **2015** (2015), no. 2, 311–328.

[Man02] J. H. Manton, *Optimization algorithms exploiting unitary constraints*, IEEE Trans. Signal Process. **50** (2002), no. 3, 635–650.

[MHWG14] C. Mu, B. Huang, J. Wright, and D. Goldfarb, *Square deal: Lower bounds and improved relaxations for tensor recovery*, Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 73–81.

[MMC90] H.-D. Meyer, U. Manthe, and L.-S. Cederbaum, *The multiconfigurational time-dependent Hartree approach*, Chemical Physics Letters **165** (1990), no. 1, 73–78.

[MMH94] J. B. Moore, R. E. Mahony, and U. Helmke, *Numerical gradient algorithms for eigenvalue and singular value calculations*, SIAM J. Matrix Anal. Appl. **15** (1994), no. 3, 881–902.

[MS14a] B. Mishra and R. Sepulchre, *R3MC: A Riemannian three-factor algorithm for low-rank matrix completion*, Decision and Control (CDC), 53rd Annual Conference on, IEEE, 2014, pp. 1137–1142.

[MS14b] ———, *Riemannian preconditioning*, arXiv preprint 1405.6055, 2014.

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

[Nes04] Y. Nesterov, *Introductory lectures on convex optimization*, Applied Optimization, vol. 87, Kluwer Academic Publishers, Boston, MA, 2004, A basic course.

[Ney02] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp. **71** (2002), no. 237, 197–216 (electronic).

[NS12] T. Ngo and Y. Saad, *Scaled gradients on Grassmann manifolds for matrix completion*, Advances in Neural Information Processing Systems 25 (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2012, pp. 1412–1420.

[NW06] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., Springer Series in Operations Research, Springer, 2006.

[ODSK14] I. V. Oseledets, S. Dolgov, D. Savostyanov, and V. Kazeev, *TT-Toolbox Version 2.3*, 2014, Available at `https://github.com/oseledets/TT-Toolbox`.

[OR95] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett. **75** (1995), 3537–3540.

[Ose11a] I. V. Oseledets, *DMRG approach to fast linear algebra in the TT–format*, Comput. Meth. Appl. Math **11** (2011), no. 3, 382–393.

[Ose11b] ———, *MATLAB TT-Toolbox Version 2.1*, May 2011, See `http://spring.inm.ras.ru/osel/?page_id=24`.

[Ose11c] ———, *Tensor-train decomposition*, SIAM J. Sci. Comput. **33** (2011), no. 5, 2295–2317.

[OT09] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput. **31** (2009), no. 5, 3744–3759.

[OT10] ———, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl. **432** (2010), no. 1, 70–88.

[PV2012] I. Pižorn and F. Verstraete, *Variational numerical renormalization group: Bridging the gap between NRG and Density Matrix Renormalization Group*, Phys. Rev. Lett. **108** (2012), 067202.

[RM00] A. Raab and H.-D. Meyer, *A numerical study on the performance of the multiconfigurational time-dependent Hartree method for density operators*, J. Chem. Phys. **112** (2000), 10718–10729.

[RS13] J. W. Robbin and D. A. Salamon, *Introduction to Differential Geometry*, September 2013, Lecture notes, available at `https://people.math.ethz.ch/~salamon/PREPRINTS/diffgeo.pdf`.

[RSS13] H. Rauhut, R. Schneider, and Z. Stojanac, *Low rank tensor tensor recovery via iterative hard thresholding*, Proc. SampTA 2013.

[RSS15] ———, *Tensor completion in hierarchical tensor representations*, Compressed Sensing and its Applications: MATHEON Workshop 2013 (H. Boche, R. Calderbank, G. Kutyniok, and J. Vybíral, eds.), Applied and Numerical Harmonic Analysis, Birkhäuser Basel, 2015, pp. 419–450.

[RU13] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Numer. Anal. **51** (2013), no. 2, 1134–1162.

[RW12] W. Ring and B. Wirth, *Optimization Methods on Riemannian Manifolds and Their Application to Shape Space*, SIAM J. Optim. **22** (2012), no. 2, 596–627.

[Sch11] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Physics **326** (2011), 96–192.

[SLS10] M. Signoretto, L. De Lathauwer, and J. A. K. Suykens, *Nuclear norms for tensors and their use for convex multilinear estimation*, Tech. Report 10-186, K. U. Leuven, 2010.

[SG11] C. Schwab and C. J. Gittelson, *Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs*, Acta Numerica **20** (2011), 291–467.

[Shu86] M. Shub, *Some remarks on dynamical systems and numerical analysis*, Proc. VII ELAM. (L. Lara-Carrero and J. Lewowicz, eds.), Equinoccio, U. Simón Bolívar, Caracas, 1986, pp. 69–92.

[Sim13] V. Simoncini, *Computational methods for linear matrix equations*, 2013, Preprint available from `http://www.dm.unibo.it/~simoncin/list.html`.

[Smi94] S. T. Smith, *Optimization techniques on Riemannian manifolds*, Hamiltonian and gradient flows, algorithms and control, Fields Inst. Commun., vol. 3, Amer. Math. Soc., 1994, pp. 113–136.

[SO11] D. V. Savostyanov and I. V. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, Proceedings of 7th International Workshop on Multidimensional Systems (nDS), IEEE, 2011.

[ST00] A. H. Sameh and Z. Tong, *The trace minimization method for the symmetric generalized eigenvalue problem*, J. Comput. Appl. Math. **123** (2000), no. 1-2, 155–175.

[STL$^+$14] M. Signoretto, Q. Tran Dinh, L. De Lathauwer, and J. A. K. Suykens, *Learning with tensors: a framework based on convex optimization and spectral regularization*, Machine Learning **94** (2014), no. 3, 303–351.

[Ste15] M. Steinlechner, *Riemannian Optimization for High-Dimensional Tensor Completion*, Technical report MATHICSE 5.2015, EPF Lausanne, Switzerland, 2015, Accepted for publication in SIAM J. Sci. Comput.

[SU15] R. Schneider and A. Uschmajew, *Convergence results for projected line-search methods on varieties of low-rank matrices via Łojasiewicz inequality*, SIAM J. Optim. **25** (2015), no. 1, 622–646.

[SPM$^+$11] M. Signoretto, R. Van de Plas, B. De Moor, and J. A. K. Suykens, *Tensor versus matrix completion: A comparison with application to spectral data*, IEEE Signal Processing Letters **18** (2011), no. 7, 403–406.

[SW82] A. H. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal. **19** (1982), no. 6, 1243–1259.

[Tob12] C. Tobler, *Low-rank tensor methods for linear systems and eigenvalue problems*, Ph.D. thesis, ETH Zurich, Switzerland, 2012.

[TTW$^+$14] M. Tan, I. Tsang, L. Wang, B. Vandereycken, and S. Pan, *Riemannian pursuit for big matrix recovery*, ICML 2014, vol. 32, 2014, pp. 1539–1547.

[Tuc66] L. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika **31** (1966), 279–311.

[Tyr00] E. E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing **64** (2000), no. 4, 367–380, International GAMM-Workshop on Multigrid Methods (Bonn, 1998).

[Usc12] A. Uschmajew, *Local convergence of the alternating least squares algorithm for canonical tensor approximation*, SIAM J. Matrix Anal. Appl. **33** (2012), no. 2, 639–652.

[Usc13] ———, *Zur Theorie der Niedrigrangapproximation in Tensorprodukten von Hilberträumen*, Ph.D. thesis, Technische Universität Berlin, 2013.

[UV13] A. Uschmajew and B. Vandereycken, *The geometry of algorithms using hierarchical tensors*, Linear Algebra Appl. **439** (2013), no. 1, 133–166.

[UV15] ———, *Greedy rank updates combined with Riemannian descent methods for low-rank optimization*, Sampling Theory and Applications (SampTA), 2015 International Conference on, IEEE, 2015, pp. 420–424.

[Van13] B. Vandereycken, *Low-rank matrix completion by Riemannian optimization*, SIAM J. Optim. **23** (2013), no. 2, 1214—1236.

[Vid03] G. Vidal, *Efficient Classical Simulation of Slightly Entangled Quantum Computations*, Phys. Rev. Lett. **91** (2003), 147902.

[VT02] M. A. O. Vasilescu and D. Terzopoulos, *Multilinear Analysis of Image Ensembles: Tensorfaces*, Computer Vision — ECCV 2002 (A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds.), Lecture Notes in Computer Science, vol. 2350, Springer, 2002, pp. 447–460.

[VV10] B. Vandereycken and S. Vandewalle, *A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations*, SIAM J. Matrix Anal. Appl. **31** (2010), no. 5, 2553–2579.

[Whi92] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett. **69** (1992), 2863–2866.

[Whi05] ———, *Density matrix renormalization group algorithms with a single center site*, Phys. Rev. B **72** (2005), 180403.

[WVS$^+$09] A. Weichselbaum, F. Verstraete, U. Schollwöck, J. I. Cirac, and J. von Delft, *Variational matrix-product-state approach to quantum impurity models*, Phys. Rev. B **80** (2009), 165117.

[Xu92] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev. **34** (1992), no. 4, 581–613.

[Xu01] ———, *The method of subspace corrections*, J. Comput. Appl. Math. **128** (2001), no. 1-2, 335–362, Numerical analysis 2000, Vol. VII, Partial differential equations.

Curriculum Vitae

Michael Maximilian Steinlechner
Born January 13th, 1987 in Marburg, Germany
Nationality: German

Education

since 2012	**École polytechnique fédérale de Lausanne, Switzerland** Doctoral studies in Applied Mathematics. Advisor: Prof. D. Kressner. Topic: *Riemannian optimization for high-dimensional problems with low-rank tensor structure.*
2014	**Princeton University, USA.** Visiting graduate student (VSRC). 6 months research stay with Prof. B. Vandereycken.
2010-2012	**ETH Zürich, Switzerland.** Master of Science ETH in Computational Science and Engineering (CSE). Field of specialization: computational physics and quantum chemistry. Master thesis with Prof. M. Reiher, *Locating Minimum-Energy Crossing points in Spin-Forbidden Reactions.*
2008-2011	**ETH Zürich, Switzerland.** Bachelor of Science ETH in Computational Science and Engineering (CSE). Bachelor thesis with Prof. D. Kressner, *A Boundary Element Method for Solving PDE Eigenvalue Problems.*
2007-2008	**ETH Zürich, Switzerland.** First year in Physics/Mathematics.
2006	**Gymnasium Sarstedt, Germany.** German *Abitur*.

Work experience

since 2012	**École polytechnique fédérale de Lausanne, Switzerland** Research assistant
2009-2011	**ETH Zürich, Switzerland** Student teaching assistant for a numerical mathematics course *Numerische Methoden* for mathematics and physics students
2006-2008	**Freelance graphic designer**
2006-2007	**Otterzentrum Hankensbüttel, Germany** Civilian service at an environmental non-profit organization.

Awards and scholarships

2014	**Mobility scholarship from the Swiss National Science Foundation** for a 6 months research stay at Princeton University, USA.
2008-2012	**German national academic foundation**

Publications

- D. Kressner, M. Steinlechner and B. Vandereycken *Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure. Technical report*, MATHICSE 18.2015. Submitted.
- M. Steinlechner, *Riemannian Optimization for High-Dimensional Tensor Completion. Technical report*, MATHICSE 5.2015. To appear in SIAM J. Sci. Comput.
- D. Kressner, M. Steinlechner and A. Uschmajew, *Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. SIAM J. Sci. Comput.*, 36(5):A2346–A2368, 2014.
- D. Kressner, M. Steinlechner and B. Vandereycken, *Low-Rank Tensor Completion by Riemannian Optimization. BIT Numerical Mathematics*, 54(2):447–468, 2014.

Conference contributions

- SIAM Conference on Linear Algebra, Atlanta, Georgia, October 26–30, 2015;
  Invited minisymposium speaker: *Riemannian Optimization for High-Dimensional Tensor Completion*
- MATHICSE Retreat, Leysin, Switzerland, July 13–15, 2015;
  Talk on *Riemannian Optimization for High-Dimensional Tensor Completion*
- Swiss Numerical Analysis Day 2015, Université de Genève, Geneva, Switzerland, April 17, 2015; Poster presentation: *Riemannian Optimization for High-Dimensional Tensor Completion*
- Pro*Doc Retreat, Disentis, Switzerland, August 14–16, 2014;
  Talk on *Low-Rank Tensor Methods for Symmetric Eigenvalue Problems*
- Oberwolfach Seminar: The Mathematics of Quantum Chemistry, Oberwolfach, Germany, November 24–30, 2013.
- Workshop on Matrix Equations and Tensor Techniques, EPFL, Lausanne, Switzerland, October 10–11, 2013;
  Poster presentation: *Low-Rank Tensor Completion by Riemannian Optimization*
- Dolomites Research Week on Approximation, Alba di Canazei, Italy, September 8–13, 2013; Invited talk on *Low-Rank Tensor Completion by Riemannian Optimization*
- CECAM Workshop on “Tensor Network Algorithms in Computational Physics and Numerical Analysis”, ETH Zürich, Zurich, Switzerland, May 15-17, 2013;
  Poster presentation: *Low-Rank Tensor Completion by Riemannian Optimization*

- Swiss Numerics Colloquium, EPF Lausanne, Lausanne, Switzerland, April 5, 2013; Contributed talk: *Low-Rank Tensor Completion by Riemannian Optimization*
- WONAPDE, Universidad de Concepción, Concepción, Chile, January 14–18, 2013; Invited minisymposium speaker: *Low-Rank Tensor Completion by Riemannian Optimization*